HAL open science

Domain Adaptation and Model Combination for the Annotation of Multi-source, Multi-domain Texts

Tian Tian

▶ **To cite this version:**

Tian Tian. Domain Adaptation and Model Combination for the Annotation of Multi-source, Multi-domain Texts. Document and Text Processing. Université Sorbonne Nouvelle - Paris 3, 2019. English. NNT : . tel-02473489v1

HAL Id: tel-02473489

https://hal.science/tel-02473489v1

Submitted on 10 Feb 2020 (v1), last revised 11 Jun 2020 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

UNIVERSITÉ SORBONNE NOUVELLE PARIS 3

Ecole doctorale Langage et langues : ED 268
Langues, Textes, Traitements informatiques et Cognition: Lattice (UMR8094)

THÈSE DE DOCTORAT CIFRE

SPÉCIALITÉ DU DOCTORAT : SCIENCES DU LANGAGE

Domain Adaptation and Model Combination for the Annotation of Multi-source, Multi-domain Texts

---

Par :
Tian TIAN

Sous la direction de :
Dr. Isabelle TELLIER†
Dr. Thierry POIBEAU
Dr. Marco DINARELLI

Composition du jury:
Iris ESHKOL, Professeure, Université Paris Nanterre, rapporteure
Anne-Laure LIGOZAT, MCF, HDR, ENSIIE, rapporteure
Sophie PREVOST, DR-CNRS, Université Paris 3, examinatrice
Thierry POIBEAU, DR-CNRS, Université Paris 3, directeur de thèse
Marco DINARELLI, CR-CNRS, Université Grenoble Alpes, co-directeur
Patrick MARTY, Ingénieur de recherche, FNAC, examinateur

Soutenue le 16 octobre 2019

Domain adaptation and model combination for the annotation of multi-source, multi-domain texts

Abstract

The increasing mass of User-Generated Content (UGC) on the Internet means that people are now willing to comment, edit or share their opinions on different topics. This content is now the main ressource for sentiment analysis on the Internet. Due to abbreviations, noise, spelling errors and all other problems with UGC, traditional Natural Language Processing (NLP) tools, including Named Entity Recognizers and part-of-speech (POS) taggers, perform poorly when compared to their usual results on canonical text (Ritter et al., 2011).

This thesis deals with Named Entity Recognition (NER) on some User-Generated Content (UGC). We have created an evaluation dataset including multi-domain and multi-sources texts. We then developed a Conditional Random Fields (CRFs) model trained on User-Generated Content (UGC).

In order to improve NER results in this context, we first developed a POS-tagger on UGC and used the predicted POS tags as a feature in the CRFs model. To turn UGC into canonical text, we also developed a normalization model using neural networks to propose a correct form for Non-Standard Words (NSW) in the UGC.

Keywords: domain adaptation, named entity recognition, machine learning, conditional random fields, neural networks

Adaptation au domaine et combinaison de modèles pour l'annotation de textes multi-sources et multi-domaines

Résumé

Internet propose aujourd'hui aux utilisateurs de services en ligne de commenter, d'éditer et de partager leurs points de vue sur différents sujets de discussion. Ce type de contenu est maintenant devenu la ressource principale pour les analyses d'opinions sur Internet. Néanmoins, à cause des abréviations, du bruit, des fautes d'orthographe et toutes autres sortes de problèmes, les outils de traitements automatiques des langues, y compris les reconnaisseurs d'entités nommées et les étiqueteurs automatiques morpho-syntaxiques, ont des performances plus faibles que sur les textes bien-formés (Ritter et al., 2011).

Cette thèse a pour objet la reconnaissance d'entités nommées sur les contenus générés par les utilisateurs sur Internet. Nous avons établi un corpus d'évaluation avec des textes multi-sources et multi-domaines. Ensuite, nous avons développé un modèle de champs conditionnels aléatoires, entrainé sur un corpus annoté provenant des contenus générés par les utilisateurs.

Dans le but d'améliorer les résultats de la reconnaissance d'entités nommées, nous avons d'abord développé un étiqueteur morpho-syntaxique sur les contenus générés par les utilisateurs et nous avons utilisé les étiquettes prédites comme un attribut du modèle des champs conditionels aléatoire. Enfin, pour transformer les contenus générés par les utilisateurs en textes bien-formés, nous avons développé un modèle de normalisation lexicale basé sur des réseaux de neurones pour proposer une forme correcte pour les mots non-standard.

Mots-clés : adaptation au domaine, reconnaissance des entités nommées, apprentissage automatique, champs aléatoires conditionnels, réseaux de neurones

Acknowledgements

I have been really lucky to have many people who helped me to accomplish this thesis, this thesis would never have been completed without them.

First, I would like to express my heartfelt gratitude to my thesis supervisors, Dr. Marco Dinarelli and Dr. Isabelle Tellier, with whom I had a meeting every two weeks in Synthesio's office along with M. Pedro Cardoso for my thesis progress. They were always by my side, gave me their advices and helped me writing, correcting and publishing papers. I also want to thank Dr. Thierry Poibeau, who took over the supervision of this thesis since Isabelle Tellier's death from a cancer, and who was always ready to read, comment and correct my writings.

I would also like to thank my two reporters, Dr. Iris Eshkol and Dr. Anne-Laure Ligozat for their helpful and complementary remarks during the presentation of this thesis. My examinator Dr. Patrick Marty also made suggestions for some experiments.

I also want to thank the director of the Lattice laboratory, Dr. Sophie Prevost, all the members of the laboratory who gave me a warm place and always encouraged me, Other phd students of Lattice, Mylène, Marine, Auphélie, Marie-Amelie, Loïc who kept me company and to whom I heartfully wish succes for their own phd soon.

I won't forget to thank my colleagues from the Paris-Sorbonne University Gaël Lejeune, Christian Vincent, Dhaou Ghoul, Vincent Lully, who helped me arranging my teaching schedule for my thesis.

At last, I want to thank my parents, my parents in-law and my husband for their encouragements, their company and their efforts to back me both emotionally and physically.

Contents

Acknowledgements **iv**

1 Introduction **1**
1.1 Background 1
1.2 Aim and Scope 2
1.2.1 About Synthesio 2
1.2.2 Internal organization 5
1.2.3 Natural language processing at Synthesio 5
1.2.4 Necessity of named entity recognition 6
1.3 Contributions 8
1.4 Thesis outline 9

2 Literature review **12**
2.1 History of Named Entity Recognition 13
2.2 Named entity definition 15
2.2.1 From the linguistic point of view 15
2.2.2 From a practical perspective 17
2.3 Domain and text genre 18
2.4 Taxonomy of named entities 20
2.5 Evaluation of Named Entity Recognition (NER) 21
2.5.1 MUC evaluation (partial match evaluation) 22
2.5.2 CoNLL evaluation (exact match evaluation) 23
2.5.3 SemEval shared task evaluation 24
2.6 Machine learning based NER methods 26
2.6.1 Supervised learning 26
Conditional Random Fields (CRFs) 28
An example of CRF training 30
Training 30
Applying the trained CRFs model 32
2.6.2 Unsupervised learning and Semi-supervised learning . 34
2.7 Text normalization for User Generated Content 35
2.7.1 Definition and scope of lexical normalization 36

2.7.2 Definition and Taxonomy of Non-Standard Words . . . 36
2.7.3 General procedure of lexical normalization 38
Neural Networks (NN) 45
Convolutional neural network (CNN) 48
Long short term memory (LSTM) 51
2.8 Conclusion 57

3 Raw data, basic normalization and Synthesio's dataset 58
3.1 Analysis of a Twitter subset data 59
3.2 Basic normalization on Synthesio's data 61
3.2.1 Use rewrite rules for tweet lexical normalization 62
3.2.2 Uppercase/lowercase normalization 62
3.2.3 Construction of Synthesio word2vec model 70
3.3 Use Google Request to find correct form for non-standard words 71
3.4 Creation of Synthesio Reference corpus 75
3.4.1 Data extraction 75
3.4.2 Entity type definition 78
3.4.3 Data annotation 79
Reference corpus statistics 80
3.5 Conclusion on Synthesio's data 81

4 Improve named entity recognition results by POS tagging 82
4.1 First NER experiments 83
4.1.1 Existing Named Entity Recognizers 83
4.1.2 Existing CRFs Named Entity Recognizer features and patterns 86
4.1.3 Annotated dataset for UGC in English 90
4.1.4 First NER results with CRFs on corpus from Ritter et al., 2011 91
4.1.5 CRFs NER models on Synthesio annotated data 93
4.1.6 Iterative training 94
4.1.7 Domain adaptation with reduced features 96
4.2 Aim and necessity to develop a POS tagger 97
4.3 Difficulties of developing a POS tagger for UGC 98
4.4 Annotated part-of-speech datasets 101
4.5 Synthesio tagset definition 102
4.6 Feature space for a POS tagger 104
4.6.1 Features related to character type 105
4.6.2 Features about uppercase/lowercase letters 105

4.6.3 Features about token value 106
4.6.4 Features using regular expression 107
4.6.5 Features using extern ressources 107
4.7 Experiments of CRFs POS tagger and results 109
4.7.1 Experiments uni-corpus 109
4.7.2 Experiments with multi-corpus 112
4.7.3 Results with artificial examples 113
4.7.4 Compare T-POS tagset and Synthesio tagset 116
Cross validation with only T-POS 116
Mixed model 116
4.8 Experiments using POS tagger results for NER 117

5 Lexical normalization of tweets with neural networks 119
5.1 Annotated datasets for the lexical normalization of tweets . . 120
5.1.1 Dataset from (Li and Liu, 2014) 120
5.1.2 Dataset from workshop ACL2015 120
5.1.3 Typology Analysis of labeled corpora 121
5.2 Use SVM for NSW and SW classification 121
5.3 Experiments of SW and NSW classification with neural networks 124
5.3.1 Context-free experiment 125
5.3.2 5-grams experiments 126
First experiment 127
Experiment by class 127
5 words embedding and characters model with sigmoid as output 128
Optimisation of configurations 129
Other Neural network structures 130
5.3.3 Experiments with pre-trained word2vec models 132
Word2vec models 132
Conclusion to word2vec model 135
5.3.4 Experiment with optimizers 135
Experiment with WNUT test 137
5.4 Word corrector 138
5.4.1 Context-free corrector 140
5.4.2 Corrector with context 142
5.5 Conclusion 149

6 Using normalized text as CRFs features for NER **150**
6.1 Using NSW/SW classification prediction as CRFs features for NER . . . 151
6.1.1 NSW/SW classification experiments on the Synthesio dataset . . . 152
6.1.2 NSW/SW classification experiments on dataset from (Ritter et al., 2011) . . . 156
6.1.3 Using NSW/SW prediction as features to NER model . 159
6.2 Using normalized text as feature for NER . . . 162
6.2.1 Word normalization based on NSW/SW classification prediction . . . 162
6.2.2 Normalization model for all tokens prediction . . . 168
6.2.3 Using normalized texts as features for NER . . . 172
6.3 Conclusion . . . 175

7 Conclusion and perspectives **176**
7.1 Summary of findings . . . 177
7.2 Limitations and Perspective . . . 184

Bibliography **186**

List of Figures

1.1 Examples of User-Generated Content applications 1
1.2 A "dashboard" with defined topics in timeline 2
1.3 Two dashboards with quality analysis 3
1.4 Example of a mention in Synthesio 3

2.1 Named Entity "Ohio" 17
2.2 Syntactic analysis by tree diagram 27
2.3 Linear CRFs 28
2.4 An unigram CRFs 29
2.5 A bigram CRFs 29
2.6 An annotated text 31
2.7 The pattern's movement 31
2.8 A text to label 32
2.9 The pattern's movement in prediction 33
2.10 NSW and definition of normalization from Han, 2014 37
2.11 Details of SW words 38
2.12 Extract of (Liu, Weng, and Jiang, 2012) dataset 39
2.13 First Classify then correct NSW 39
2.14 First propose candidates then sort 40
2.15 An example of tweet 40
2.16 The correction of the tweet example 41
2.17 normalization processing in (Leeman-Munk, Lester, and Cox, 2015) 43
2.18 Screen shot for Google's auto-correction 44
2.19 a simple neural network in general 46
2.20 A perceptron 47
2.21 CNN's window movement 50
2.22 CNN's output 50
2.23 Max pooling with CNN output 51
2.24 CNN from (Collobert and Weston, 2008) 51
2.25 Elman's RNN 52
2.26 LSTM States 53

2.27 LSTM gates calculate inspired by colah's blog (http://colah.github.io) . . . 55
2.28 A simple CBOW model from (Rong, 2014) . . . 56

3.1 Google page with the request "I 4got my password" . . . 72
3.2 Page descriptions for "posted by wfolarry view post" . . . 73
3.3 Examples of a dashboard with three widgets . . . 76
3.4 An example of an attached apostrophe . . . 78
3.5 An example of long text . . . 80
3.6 An example of short text . . . 80

4.1 An example of Stanford NER prediction . . . 86
4.2 Example of "North Yorkshire" . . . 95
4.3 Example of "witch doctor" . . . 96
4.4 An example of tweet . . . 98
4.5 Correct form of tweet 4.4 . . . 98
4.6 Number usage in tweets from (Ritter et al., 2011) . . . 99
4.7 Another exemple of tweet . . . 99
4.8 Hashtag as part of sentence . . . 100
4.9 Hashtag with multi-words . . . 100
4.10 At mention as part of sentence . . . 101
4.11 At mention at the end of sentence . . . 101
4.12 Cross validation adding Penn Treebank data . . . 112

5.1 NSW substitution . . . 122
5.2 An example of sentence with NSW . . . 139
5.3 CNN structure for corrector with context . . . 143
5.4 Context-free CNN model structure with ch2vec . . . 145
5.5 CNN 5-grams model structure with ch2vec . . . 145

6.1 Token "ca" in training data . . . 154
6.2 Example of named entities as standard words . . . 154
6.3 Example of noise in Deezer short text . . . 155
6.4 Example of named entity and a predicted NSW . . . 155
6.5 Example of a tweet in automobile demain . . . 156
6.6 Example of succeeded NSW predictions . . . 157
6.7 Example of Dutch tweet in dataset . . . 157
6.8 Example of missed NSW predictions . . . 157
6.9 Example of missed NSW predictions . . . 158
6.10 Example of noises . . . 158

6.11 Example of noises 158
6.12 Tweet full of spelling errors 159
6.13 Example of "BBC", predicted as NSW and detected as Company 160
6.14 Token "DD" detected as Company 161
6.15 Another occurrence of "DD" 161
6.16 Example of "DC" 161
6.17 Token "llol" in training data 170
6.18 Token "fraggle" in training data 170
6.19 Named entity distribution in two datasets 174

List of Tables

2.1 Examples of "McDonald's" 18
2.2 An example of MUC-6 annotation 21
2.3 An example of a NER system prediction 22
2.4 Same prediction example with SemEval evaluation 25
2.5 SemEval 2003 Evaluation results 26
2.6 Feature functions values 33
2.7 Feature functions summations for each Y position 33
2.8 All possible sequences and the best path 34
2.9 Taxonomy of NSW in (Sproat et al., 2001) 36
2.10 Methods of WNUT workshop 38
2.11 SW/NSW classification references and results 40
2.12 Example of CRF labels for correcting NSW word 41

3.1 Term count and number of OOV words 60
3.2 Forms and their percentages for "crystal" 64
3.3 Forms and their percentages for "jack" 64
3.4 Forms and their percentages for "share" 64
3.5 Forms and their percentages for "however" 64
3.6 Forms and their percentages for "nissan" 64
3.7 Forms and their percentages for "McDonald" 64
3.8 Forms and their percentages for "4Runner" 64
3.9 Examples of the most frequent two forms 65
3.10 Vocabulary filters 68
3.11 Final vocabulary filters 68
3.12 Most frequent 5-grams in automotive industry domain 69
3.13 The most frequent context in automotive industry domain . . 70
3.14 Nearest words of "Chinese river" 71
3.15 Contexts and term count for "wfolarry" 73
3.16 Context of "wfolarry" and matched words 74
3.17 Candidates and edit distance with original token 74
3.18 Other edit distance with original token 74
3.19 Synthesio Reference Corpus. 78

3.20 Synthesio Named Entity Definition 79
3.21 Synthesio corpora named entities statistics 80

4.1 NER Toolkits 83
4.2 CRFs pattern inspired by Nooralahzadeh, Brun, and Roux, 2014 89
4.3 Model Constant1 89
4.4 Model Constant2 90
4.5 Annotated named entities in (Ritter et al., 2011) 90
4.6 NER results with Constant1 pattern 91
4.7 NER results with Noor pattern 92
4.8 NER results with constant1 pattern 92
4.9 Synthesio corpus evaluation results 93
4.10 Evaluation of a mixed model trained with the Ritter corpus and predicted sequences with a confidence score more higher than 0.8. 95
4.11 Number of selected sequences in each domain 95
4.12 Evaluation of model with Ritter on reduced features with Deezer. 96
4.13 Universal tagset and Penn Treebank tagset mapping 103
4.14 PTB, Ritter and universal tagset correspondance 104
4.15 POS tags of token "a" in Penn Treebank 107
4.16 POS tags of token "down" in Penn Treebank 107
4.17 Penn Treebank POS tags feature values examples 108
4.18 Sample of Brown Clusters for Tweets 108
4.19 Results with model Nooralahzadeh2014 110
4.20 Model Constant1 111
4.21 Model Constant1 results 111
4.22 Model Constant2 and results 112
4.23 Results of mixed-corpora experiments 113
4.24 Examples to test POS tagger 114
4.25 Prediction results with first sentence "Yesterday I bought a new Renault. It was cool, better than my old Ford focus." 114
4.26 Prediction results with second sentence "got an iPhone6 for my birthday, better than my old BlackBerry, wonderful!" 115
4.27 Prediction results with third sentence "A paper by Maggie Simpson and Edna Krabappel from New York was accepted by two scientific journals urlhttp://vox.com/e/7103628?utm_campaign... ..." 115
4.28 Prediction results with fourth sentence "If I had one wish it would be to drive a bugatti for a day!" 115
4.29 cross validation with corpus T-POS 116

4.30 Results of mixed models with different tagsets 117
4.31 NER results using POS tagger as feature 118

5.1 Numbers of tokens in each class 121
5.2 NSW substitution in English 122
5.3 Two datasets for text normalization and statistics 123
5.4 Results of cross validation in 5 folds 124
5.5 Train a model with one dataset and test with the other 124
5.6 Results of experiments with uni convolution layer 126
5.7 Results of experiments with multi convolution layers 126
5.8 Neural Network structure of character and word embedding model 128
5.9 NSW and SW classification results by class 128
5.10 5 words embedding and characters Convolutional model structure . 128
5.11 5 words embedding and characters convolutional model with softmax functon 128
5.12 5 words embedding and characters convolutional model with sigmoid functon 129
5.13 Optimisation of character embedding and word embedding model 129
5.14 5 words embeddings convolutional model structure 130
5.15 optimisation of 5 word embedding model 130
5.16 5 words embeddings LSTM model structure 131
5.17 Compare convolution and LSTM model 131
5.18 LSTM model on 2577 tweets 131
5.19 LSTM experiments with 5-fold cross validation 132
5.20 Early Stopping with 100 epochs 132
5.21 CNN model with dataset in (Li and Liu, 2014) 133
5.22 CNN model with dataset (Baldwin et al., 2015) 134
5.23 CNN model with 2 datasets 134
5.24 Optimizers experiment using (Li and Liu, 2014) 135
5.25 Optimizers experiment using (Baldwin et al., 2015) 136
5.26 Optimizers experiment on mixed dataset 136
5.27 Experiment with WNUT test, model trained with WNUT training 137
5.28 Experiment with WNUT test, model trained with WNUT and 2577tweets 138
5.29 Token "txtd" in characters 139
5.30 Token "txtd" representation as vector 139

5.31 Correction "texted" in character 139
5.32 Correction "texted" vector 139
5.33 CNN structure for corrector 139
5.34 First results of Corrector using CNN model 140
5.35 Ambiguous token examples 140
5.36 Word list: corrector for all tokens 141
5.37 CNN structure for training NSW 141
5.38 Succeeded examples using edit distance 142
5.39 Failed examples using edit distance 142
5.40 Vocabulary sizes 143
5.41 Corrector with context results on all words 143
5.42 Time needed for each vocabulary 143
5.43 Training and testing with only NSW 144
5.44 Results analysis by category by model 146
5.45 Some result of formes IV in training data 146
5.46 Some result of formes IV not in training data 147
5.47 Predict an IV forme by an IV but correction is OOV 148
5.48 Correction is IV but predicted with OOV 148
5.49 OOV with OOV correction predicted as IV 148

6.1 NSW/SW Classification on the Synthesio Dataset 152
6.2 NSW/SW Classification by domain and text type 153
6.3 Classification cases with the token "ca" 154
6.4 Example of a predicted NSW in Deezer short text 155
6.5 NSW/SW Classification on (Ritter et al., 2011) 157
6.6 NER results using NSW/SW classification feature 160
6.7 Annotated named entities in the text of figure 6.13 160
6.8 Succeeded normalization examples on (Ritter et al., 2011) . . . 163
6.9 Wrong candidates for predicted NSW 164
6.10 Wrong candidate for predicted NSW 165
6.11 candidate for predicted NSW on the Synthesio dataset 166
6.12 Correctly predicted tweet in (Ritter et al., 2011) 169
6.13 Miss-corrected tokens in (Ritter et al., 2011) 170
6.14 Miss-corrected tokens in (Ritter et al., 2011) using Aspell dictionary 172
6.15 NER results using POS tagger as feature 173
6.16 Clients names and their concurrents 174

List of Abbreviations

NER	**N**amed **E**ntity **R**ecognition
NLP	**N**atural **L**anguage **P**rocessing
CRF	**C**onditional **R**andom **F**ields
NN	**N**eural **N**etwork
CNN	**C**onvolutional **N**eural **N**etwork
LSTM	**L**ong **S**hort-**T**erm **M**emory
POS	**P**art **O**f **S**peech
NSW	**N**on **S**tandard **W**ords
SW	**S**tandard **W**ords

Chapter 1

Introduction

User-Generated Content (UGC) is all form of content (photographs, videos, podcasts, articles and blogs) created by Internet users and posted publicly (George and Scerri, 2018).

A growing number of User-Generated Content (UGC) can be found on the Internet everyday. Users increasingly express their opinions, concerns and other sentiments on online platforms. YouTube[1], Facebook[2], Wikipedia[3] and Twitter[4] are some of the most popular social media which enable people to comment, like, edit or share their thoughts on different subjects.

This thesis deals with Named Entity Recognition (NER) processing application and text normalization on non-canonical words through textual User-Generated Content (UGC). We will use the term UGC, for User-Generated Content in the rest of this document.

1.1 Background

The huge amount of User-Generated Content (UGC) drives many new applications as shown in figure 1.1.

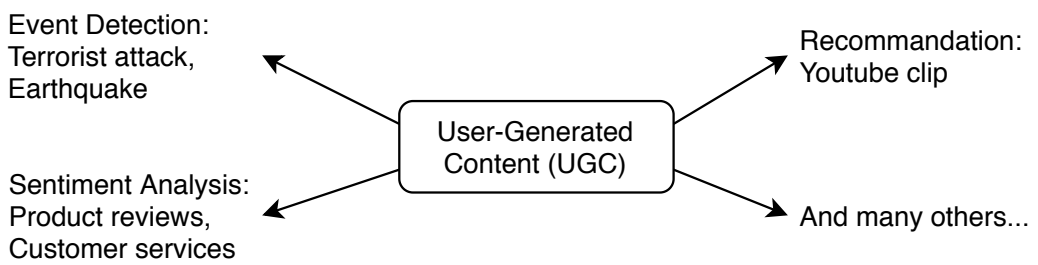

FIGURE 1.1: Examples of User-Generated Content applications

[1] https://www.youtube.com
[2] https://www.facebook.com
[3] https://www.wikipedia.org
[4] https://www.twitter.com

Beyond these examples, UGC has attracted attention from the research community, government and industry: researchers dig from UGC to make predictions on stock trends (Bollen, Mao, and Zeng, 2010); governments may track public opinion in order to predict possible presidential outcomes (Hu et al., 2012); companies use UGC to analyze user sentiment on particular products (Jiang et al., 2011).

1.2 Aim and Scope

This thesis was funded by Synthesio (https://www.synthesio.com) in the context of CIFRE (Conventions Industrielles de Formation par la REcherche) in France.

1.2.1 About Synthesio

Synthesio, a global social intelligence and information monitoring company, provides E-reputation services. Brands use the Synthesio Social Listening Platform to monitor their online presence:

Synthesio customers belong to various domains and include global brands such as L'Oréal (cosmetics), Deezer (entertainment) and Nissan (automobile) etc. Synthesio provides different services such as social listening, alerting and audience insights.

Social listening

Synthesio follows real-time conversations on the Internet around the world with its industry-leading dashboards and data coverage.

Its clients define their "dashboards" with key-words which interest them, time periode, geo-location (of posters), languages and ressources as their command.

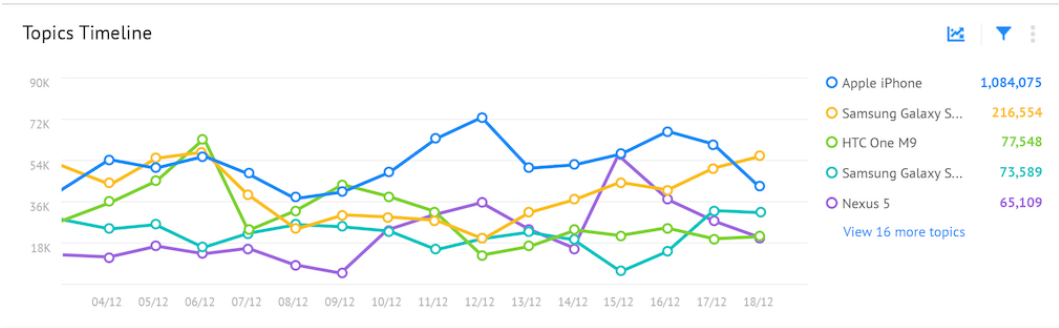

FIGURE 1.2: A "dashboard" with defined topics in timeline

Figure 1.2 shows a dashboard defined by a time period (from December 4th to Decembrer 18th 2013) and key-words "iPhone", "Samsung Galaxy S10" (in yellow), "HTC One M9", "Samsung Galaxy S8" (in dark green) and "Nexus S". This dashboard shows during a fixed time period, the proportion of different topics (subject brand and competitor brands) that have been mentioned in all ressources (mainstream and social media) in the mobile phone domain.

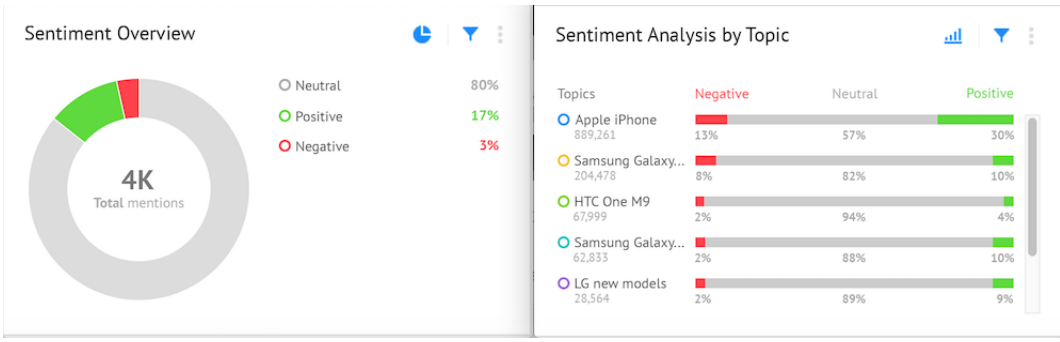

FIGURE 1.3: Two dashboards with quality analysis

Figure 1.3 shows two other dashboards with a sentiment overview for the target object (on the left) and a sentiment analysis with concurrent topics (on the right).

Synthesio processes data which covers over 80 languages from 195 countries all over the world. The data is collected from mainstream resources like corporate websites, online newspapers, information websites, specialized forums, etc, and social media like Facebook, Twitter and Instagram. Most of this content is generated by Internet users.

The minimal unit of data in Synthesio is called a "mention". All mentions are enriched with metadata which may include the title (if there is one), content, author, time, source (often Internet address) and tone (positive, negative or neutral). The data can then be filtered and visualized by the customer.

mention ID: xxxxxxxxxx24795962548
title:
content: dnce 2 day at skool 2 dau wuz kinda corny evn doe i dnce a lil bit ugh bored lke always whn is ya gurl neva nt trin 2 fin sunthn 2
sources: Twitter
url: `https://twitter.com/sweetangel4000/status/24795962548`
topic:
tone:

FIGURE 1.4: Example of a mention in Synthesio

Figure 1.4 shows an example of a mention from Twitter in Synthesio. The field "mention ID" is the unique identification. For tweets, the field "title" is empty but for other ressources like newspaper articles, the value of the field "title" is the title of the article. The field "topic" is annotated by humans with key-words like "mobile phone", product name etc. The key-words are also indexes used to find corresponding mentions according to clients' research. The "tone" field has three possible values: positive, negative and neutral, annotated by sentiment analyzer.

Analysis of global social situation can help Synthesio customers create or modify their communication strategies and surveil the impact of their social activities.

Synthesio data also contain interaction and engagement metrics. Synthesio tracks views, likes, favorites, replies, retweets and shares from social media like Facebook, Twitter, Instagram and YouTube directly in the social media listening dashboards.

Synthesio can also generate interaction (likes, edits or share) timelines to determine how long a post was influent or to analyze the brand's share of voice as part of a competitive dataset.

Alerting

Synthesio constantly scans mainstream news sites (such as CNN[5]) specialized discussion forums and social media (like Facebook or Instagram). Synthesio has also signed a special contract with Twitter which allows it to receive 10% of Twitter data in real-time. Some analysis on this constantly arriving stream of data is done in real-time. Particularly, if not originally present, geolocation information is added when possible. This allows Synthesio to detect new trending keywords and topics very quickly, to locate potential events on the map and to alert specific customers.

For a concrete example, one of Synthesio's customers subscribing to this real-time alerting service is a security company. It has subscribed to danger and violence-related keywords associated to public concert venues in the United States. Its goal is to respond to public attacks and shootings as fast as possible. Social media alerting allows to receive this kind of information faster than traditional systems based on phone calls.

[5] https://edition.cnn.com/

Audience insights

Synthesio runs demographic analysis on more than 10,000 data categories related to target audiences. Its clients can both visualize the popularity of some of their products on sale and analyze what their own clients are talking about. They can feed their marketing automation programs with deep demographic audience analysis to deliver customized messaging.

Synthesio can also profile the audience from email, web visits and social media. Its clients can analyze any target group based on age, location, interests and competitive affinities. They will have surface audience insights from fans, followers or even consumers who visit the client's website.

When demographic information is missing in the original text, Synthesio uses a supervised machine learning method to complete the fields gender, age, location etc.

1.2.2 Internal organization

The Synthesio group has agencies and marketing teams in London, Singapore (for APAC), Paris and New York. Besides, there is a Research and Development (R&D) group of 20-30 people, and a 5-person sourcing group in France. This R&D group has a back-end team, a front-end team, an infrastructure team and a Natural Language Processing (NLP) team.

The sourcing group collects URLs of websites and forums according to clients' domains and requirements. The back-end team develops and maintains a crawler to fetch documents from addresses obtained by the sourcing group and stores the mentions on Synthesio servers.

The infrastructure team ensures the smooth operation of all Synthesio services. Synthesio uses more than 500 physical servers and virtual machines located all over the world in order to store and process data up to three years. As for client's visualisation, the front-end team keeps improving the visual experience of the dashboards by adding animation effects, changing color combinations etc.

1.2.3 Natural language processing at Synthesio

The Synthesio Natural Language Processing (NLP) team is doing more research-driven work. Its members have developed a language identifier, tokenizers for different languages and a sentiment analyzer.

For each of these tasks, the NLP team first tests open-source solutions and runs them on existing Synthesio data in order to improve internal models. If the results are unsatisfying, the team either develops its own software or considers licensing proprietary pieces of software.

The present research work has been realized between November 2014 and October 2017 for the Synthesio NLP team and was based on the language identifier and the English tokenizer. Their functions are described later. At that time, the team was working on demographic detection for the Synthesio audience insights service, language identifiers for Arabic transliteration (Dias Cardoso and Roy, 2016) and tokenizers for Chinese, Japanese and Korean.

The language identifier is based on a 5-gram language model. It takes a text (a word sequence) for input and its output is the most probable language of this text associated with its probability.

The tokenizers work with 80 languages supported by Synthesio. The English tokenizer uses a neural network (NN) and exception lists. Texts written in some Asian languages such as Chinese, Japanese, Korean or Thai are the most difficult to tokenize because there is no blank between words.

The English sentiment analyzer is working with a hybrid method with Neural networks (NN) and word lists (Cardoso and Roy, 2016).

The present research work has been realized between November 2014 and October 2017 for the Synthesio NLP team and was based on the language identifier and the English tokenizer. Their functions are described later. The team was working on demographic detection for Synthesio audience insights service, language identifier for Arabic transliteration (Dias-Cardoso2016), Asian languages tokenizers (Chinese, Korean and Japanese) and improvement of its existing modules.

1.2.4 Necessity of named entity recognition

In Synthesio, each document to be analyzed is considered as a "mention". A mention contains a unique ID, the original text, a time stamp, source information (url), maybe a title (depends on the ressource type) and a "topic". The topic is supposed to be the subject discussed in the text and is associated with a tone (positive, negative or neutral) as shown in figure 1.4.

Sometimes, one text can be repeated several times. These repetitions are considered as different mentions even though their original texts are the same. That is the reason why Synthesio has sometimes different mentions

with the same content. The duplicated data mean that the subject is mentioned several times at different moments or by different users. It is important to keep all occurrences for Synthesio to show quantity analysis information (like shown in figure 1.2) and to track "likes", replies and retweets. However, from the Natural Language Processing (NLP) point of view, it would be a waste of time to process all these duplicated documents: the same document more than once, and it will also change evaluation results. To avoid this, removing duplicated texts in an extracted dataset is necessary.

The pipeline of Natural Language Processing of Synthesio in the NLP team for the original text contains a natural language identifier, a tokenizer, a sentiment analyser and a search engine. The tokenizer and the sentiment analyser are different for each natural language. For each document, the language identifier begins by identifying the natural language of the text. In a second time, the corresponding tokenizer turns the text into an ordered token list. After applying a filter of stopwords, the tokens remained in the list are then indexed for the search engine.

As mentioned before, the clients of Synthesio can create their proper dashboard with a selected time period, chosen ressources and most importantly, with the keywords they entered. The search engine works with indexed keywords. The customers have to enter their requests using Structured Query Language (SQL).

These keywords are often the name of their company, their brands, the name of competitors' brands, the name of their products, similar products in the market, etc. To analyze the texts, they sometimes try to get statistics for a specific location, for example in Europe. These keywords contain more useful information than a single token, a string with an assigned and identified meaning. They also have different types (location, company, brand) and they are probably the "topic" that one client could search for.

Some brand names are common words like "Apple", "Orange" (telecommunication company), "President" (French cheese brand). Clients searching for "Apple" have to mention excluded words like "fruit" or "pie" in their SQL requests to ignore mentions with "apple" and "fruit" or "pie" in the same text or mentions related to apple pie recipes.

If we can add these types of information as metadata to these tokens of company, brand and location names, there would be more information in the research engine to distinguish the ambiguous names and to improve its performance. The procedure would add a pre-treatement for each document, to mark for example which token is a brand name, which token is a person

name, etc. That is to say, to add some metadata into these names.

1.3 Contributions

- Creation of Synthesio Corpus

  Synthesio processes large amount of User-Generated Content (UGC) everyday in 80 languages. 80% of textual data are in English (from all users all over the world). When we want to extract named entities from English text, we first need to create a dataset for evaluation. This dataset represents multi-domain, multi-source textual User-Generated Content (UGC) and is annotated with nine types of Named Entity Recognition (NER) and twelve Part-of-Speech (POS) tags.

- Part-of-Speech (POS) tagger on User-Generated Content (UGC)

  According to Synthesio's needs, we defined a universal Part-of-Speech (POS) tagset with twelve possible POS tag (for all the 80 languages that Synthesio processes) and a mapping rule from other tagsets (of Penn Tree Bank from Marcus, Marcinkiewicz, and Santorini, 1993 and T-POS dataset from Ritter et al., 2011).

  If we consider User-Generated Content (UGC) and well-formed text as two different domains, to develop a Part-of-Speech (POS) tagger for User-Generated Content (UGC), we tried two domains adaptation methods: For the first one, we added well-formed text from Penn Tree Bank (Marcus, Marcinkiewicz, and Santorini, 1993) into the training data (Ritter et al., 2011) to adapt well-formed text of standard English with specificities of User-Generated Content (UGC). We tested different proportions of well-formed text and different numbers of features for different texts.

  Our second domain adaptation method is a semi-supervised training method: the bootstraping method. The goal is to get use of Synthesio's large amount of unannotated texts to improve the results of our POS tagger.

- Using Part-of-Speech (POS) tagger prediction as CRFs model on NER

  We tagged the Synthesio dataset with our Part-of-Speech (POS) tagger and we added these tags into features of the CRFs model for Named Entity Recognition (NER).

We also showed that the POS tagger improved the NER results on the Synthesio NER evaluation dataset.

- Text normalization

  Since User-Generated Content (UGC) often has a lot of Non-Standard Words (NSW) compared to well-formed texts, the goal of text normalization is to turn non-canonical texts into well-formed texts. Natural Language Processing (NLP) tools which perform better with well-formed texts could be useful for normalized texts.

  We developed a Non-Standard Words (NSW) and Standard Words (SW) classifier with a Convolutional Neural Network (CNN) model trained on datasets from Baldwin et al., 2015 and Li and Liu, 2014.

  We also developed another Convolutional Neural Network (CNN) model to normalize Non-Standard Words (NSW) in User-Generated Content (UGC) (from Baldwin et al., 2015 and Li and Liu, 2014).

- Using normalized text as a feature in CRFs model for NER

  We used Non-Standard Words (NSW) and Standard Words (SW) classes, and then normalized text as features in previous CRFs model in order to improve Named Entity Recognition (NER) performance on the Synthesio dataset. This feature has helped to slightly improve these NER results.

1.4 Thesis outline

The main goal of this thesis is to label defined named entities automatically from textual User-Generated Content (UGC) for Synthesio. This work aims to improve the performance of Synthesio's existing search engine and to help reduce ambiguity by adding named entities types as metadata into the text collection.

Chapter 2

Chapter two begins with a literature review of the task of Named Entity Recognition (NER) : the definition of a named entity from different points of view, Named Entity Recognition (NER) with different text domains and different text genre, taxonomy of named entities, how we evaluate a Named

Entity Recognition (NER) system, and general Named Entity Recognition (NER) methods on canonical text: methods by rules and machine learning.

Since extracting named entities for linear text turns to a sequence labelling task, we then explain how the most efficient sequence labelling method the Conditional Random Fields (CRFs) works with an example.

Then some concrete Named Entity Recognition (NER) methods with user-generated content (social media text) are presented.

Chapter 3

Chapter three is about how we created an annotated dataset for Named Entity Recognition (NER) evaluation and another unannotated data collection as a vocabulary.

We will explain how we extracted data from different domains and different ressources stored in the Synthesio dataset, how we defined named entity types and how we annotated the corpus. Statistics obtained from this reference corpus are shown later in this chapter.

We will then explain the process of collecting another (non-annotated) dataset from the Synthesio data collection. We will describe how we extracted and normalized texts using regular expression for some types of entities like quantities, volumes etc...

Chapter 4

In chapter four, we will show the first developed CRFs model for Named Entity Recognition (NER) on the Synthesio dataset.

For this first CRFs model, we have employed part-of-speech (POS) tag in the Penn Tree Bank dataset from Marcus, Marcinkiewicz, and Santorini, 1993 as a feature. Since one token can have more than one Part-of-Speech (POS) tag according to different contexts, this feature contains four (because it is the maximum number of possible POS tags for one token in the dataset Penn Tree Bank) sub-features, sorted by frequency. If we want the POS tag for the token in its context, we need a Part-of-Speech (POS) tagger.

We will then present some experiments of developing a POS tagger with CRFs model on User-Generated Content (UGC). To improve the POS tagging results, we also used domain adaptation and bootstrapping for POS tagging.

We will then uses the developed Part-Of-Speech (POS) tagger in order to improve the NER results on the Synthesio dataset.

Chapter 5

In chapter five, we will first present existing text normalization datasets and typology of Non-Standard Words (NSW). We will then show first experiments of Non-Standard Words (NSW) classification with Support Vector Machine (SVM).

Then we will show the classification results of Non-Standard Words (NSW) with Convolutional Neural Network (CNN), and then try to normalize Non-Standard Words (NSW) with another Convolutional Neural Network (CNN) model.

Lastly, we will try to combine the NSW classifier and the text normalization model.

Chapter 6

In chapter six, we will present results of Non-Standard Words (NSW) and Standard Words (SW) classifier on the Synthesio dataset and the annotated Named Entity Recognition (NER) dataset (Ritter et al., 2011) training data.

Then we will employ the prediction of NSW/SW classifier as a feature in the CRFs model for Named Entity Recognition (NER) on the Synthesio dataset.

We will also present the results of text normalization models on the Synthesio dataset and the dataset from Ritter et al., 2011 (for NER training). We will add the normalized text as a feature in the CRFs model to try to improve the Named Entity Recognization (NER).

Chapter 7

This chapter concludes the research outcomes of our proposed methods for Named Entity Recognition (NER) and text normalization on User-Generated Content (UGC).

We summarise our contributions and give a perspective for improving Named Entity Recognition (NER), for developing a Part-of-Speech (POS) tagger for Non-Standard Words (NSW) and Standard Words (SW classification and for text normalization on Non-Standard Words (NSW) on User-Generated Content (UGC).

Chapter 2

Literature review

This chapter reviews the literature on Named Entity Recognition (NER) and text normalization.

First, the definition of named entity is presented from both linguistic and Synthesio's point of views.

Secondly, previous works on Named Entity Recognition (NER) in different domains and with different types of text genres (journalistics texts, medical notes or tweets) are discussed in section 2.3.

Thirdly, the taxonomy of named entity is illustrated and the evaluation methods for NER are examined.

Machine learning based methods are then explained in section 2.6.

Written text can be considered as linear or structural. Linear as a sequence of words (or tokens), structural as constituent analysis on syntactic structures (Chomsky, 1957). However, named entities inside the text are often local contiguous sequences in case of multi-words named entity expressions (or only one word) in the text. Therefore, Named Entity Recognition (NER) tasks can be solved by sequence labelling. Each label corresponds to a word (or a token) in the text and these labels indicate if the word (or the token) begins (B), or is inside (I) or out (O) of a named entity. The most efficient sequence labelling model, the Conditional Random Fields (CRFs) is then explained with a concrete example.

Most of Synthesio's texts are from User Generated Content (UGC) found on the Internet: product reviews, discussions from forums, and tweets from Twitter. This User Generated Content (UGC) is full of abbreviations, spelling errors and other non-canonical words. Specificities of User Generated Content (UGC) are then presented along with procedures of lexical normalization for UGC.

Some machine learning methods: Convolutional Neural Networks (CNN) and Long-Short Term Memories (LSTM) are proved as efficient text normalization methods as in Yolchuyeva, Németh, and Gyires-Tóth, 2018; Kim et al.,

2015; Min and Mott, 2015. They are then explained with diagrams. Word embeddings, a word representation by vector, is often used as neural network input. Word2vec (Mikolov et al., 2013), developed by Google and Glove in Pennington, Socher, and Manning, 2014 is largely used as word embeddings. This word2vec training method, which is used by Synthesio, will be explained in the end of this chapter.

Contents

1.1 Background **1**
1.2 Aim and Scope **2**
1.2.1 About Synthesio 2
1.2.2 Internal organization 5
1.2.3 Natural language processing at Synthesio 5
1.2.4 Necessity of named entity recognition 6
1.3 Contributions **8**
1.4 Thesis outline **9**

2.1 History of Named Entity Recognition

In 1991, for the first time, "extracting company names" is mentioned in the information extraction task on the Conference on Artificial Intelligence Applications of Institute of Electrical and Electronics Engineers (IEEE) (Rau, 1991).

Later, in the Sixth Message Understanding Conference (MUC-6) on 1995 (Sundheim, 1995), the "named entity" task aimed at tagging named entities using the Standard Generalized Markup Language (SGML) to label each character string representing:

- ENAMEX: a person, organization, location name
- TIMEX: a date, a time stamp
- NUMEX: a currency or percentage figure

The tag elements were ENAMEX, TIMEX and NUMEX respectively.

However, they ignored product names like "Macintosh", other miscellaneous names like "Wall Street Journal" (in reference to the periodical as a physical object), names of groups of people like "Republicans" and addresses.

It was also the first time the term "named entity" had been employed. The task "coreference", related to named entity recognition, was also proposed in MUC-6.

Later, in the Multi-lingual Entity Task (MET) of the Association for Computational Linguistics (ACL) conference in 1996, there was a proposed task "locate and tag with SGML: named entity expressions (person, organizations and location); time expressions (time and date) and numeric expressions (percentage and money)".

At that moment, the notion of "named entity" contained only person names, organizations names and location names.

Two years later, on the Hub-4 workshop, Jonathan et al., 1999 talked about the Information Extraction Named-Entity Evaluation (IE-NE). From that point, named entity recognition (NER) has been considered as a subtask of information extraction.

Similar to the previous MUC-6, MUC-7 task (Chinchor and Robinson, 1998) was concerned with the extraction of three types of information: named entities, included `organisation`, `person` and `location`; temporal expressions and number expressions.

A very popular NER shared task was the Conference on Computational Natural Language Learning (CoNLL) 2003 (Tjong Kim Sang and De Meulder, 2003). This shared task was called language-independent named entity recognition and both English and German corpus were evaluated.

There were four types of named entities to be extracted: `Organization`, `Person`, `Location` and `MISC` (miscellaneous names). There exists recursive or overlapping named entities in natural language, but the authors chose to annotate only the top level entity (the most complete entity) when a named entity was embedded in another named entity.

Right after CoNLL2003 in 2004, the Automatic Content Extraction (ACE) evaluation framework was proposed (Doddington et al., 2004). The tasks included "recognition of entities", "recognition of relations", "event extraction" and "event extraction on speech and on optical character recognition (OCR) input". The "recognition of entities" part required not only named entities as defined above, but also all referential expressions representing an entity, a name that could be a description, a pronoun or a noun phrase. The goal was to reuse these extracted mentions to extract relationships among all reference of entities in the part of "recognition of relations" task.

Instead of recognizing instantly all types of defined named entities, we can also first segment the beginning and the end of named entities and then

classify extracted named entities into different types. That is how we talk about named entity recognition and classification (NERC) (Poibeau and Kosseim, 2001).

Nowadays, Named Entity Recognition (NER) is considered as a subtask of information extraction, and the tasks related to NER vary from "coreference" (Denis and Baldridge, 2009; Hajishirzi et al., 2013; Durrett and Klein, 2014), relation extraction (Bunescu and Mooney, 2005), entity linking (Doddington et al., 2004), knowledge base creation (Surdeanu, 2013) to automatic summarization (Gupta and Lehal, 2011) and question answering (QA) system (Han et al., 2017).

2.2 Named entity definition

As we can see through the task of Information Extraction (IE), depending on Natural Language Processing (NLP) tasks, named entities are often person names, organisation names, location names and sometimes other time, date or currency expressions. Apart from these examples of named entities, there is not a clear, exact definition of what a named entity is.

According to linguists, a named entity is a linguistic unit that makes reference to a named entity in the world (Ehrmann, 2008; Fort, Ehrmann, and Nazarenko, 2009; Dupont, 2017). This reference could refer to a specific object with "a proper name in every possible world in which the object exists" (by the American philosopher Saul Kripke in Kripke, 1972).

Therefore, a named entity refers to a concrete, existing object of a certain domain. It has a proper name with a general agreement in the community of the domain (Nadeau and Sekine, 2007; Nouvel, Ehrmann, and Rosset, 2016).

A named entity can be an organisation (a company, a facility), a person, location (a city name, a country), and also a movie name, a song, a character name (in a book) or a holiday name (like "Christmas"). It can also be a protein name (Dupont, 2017) or a disease name (Zhao et al., 2017) in the biomedical domain.

2.2.1 From the linguistic point of view

In general, Named Entities (NE) are "mentions" refering to entities of a specific domain, those mentions relating to different linguistic categories. A Named Entity (NE) can be composed of only one proper noun like "Rabelais",

"Reuter" and "Orange". It can also be a noun phrase with common nouns like "Assurance Maladie" (French for "health insurance").

If we follow the definition that all named object is a named entity, there are film names, musician names, book names, fictional characters, sport-steam, holidays, road names, airplane names (such as "Concorde") etc. From the linguistic point of view, the original forms of named entities could be common nouns (as "Orange"), noun phrases (as "Long Island") or complete sentences (as a book name "Who Moved My Cheese?"), etc..

As a convention, in English, French and most Indo-European languages, proper names begin with the first letter in uppercase like "Reuter", "Apple" (as a company name) like person names (like "Obama").

Multi-words expressions are more complicated: in English, full words and stop words are distinguished. Full words are words which contain meanings in a sentence: nouns, pronouns, verbs, adjectives, and adverbs. Stop words are grammatically necessary and are filtered out before or after processing of natural language data (Rajaraman and Ullman, 2011): articles, prepositions, or coordinating conjunctions. In English, all words in a location name begin with an uppercase letter as in "New York"; But for book titles, only full words begin with uppercases (and stop words begin with lowercase letters) as in "Game of Thrones". There are also abbreviations with all letters in uppercase like "CNN" (Cable News Network) and sometimes there can be dots that separate these letters as in "S.N.C.F" (French for "Société Nationale des Chemins de Fer"). Some brand names also mix uppercase characters and symbols as in "H&M" and "C&A".

The uppercase is often a mark of proper names and named entities. However, the convention that the first letter of a sentence be in uppercase makes the first word ambiguous. "Twins" with "T" in uppercase can be a common noun in the beginning of a sentence, or it can also refer to an all-female band, a film name, a book name etc[1]. "Kiss" with "K" in uppercase is also a rock band name [2].

Without the first letter in uppercase, one word could be ambiguous: it could be a named entity or not. As mentioned before, "orange" is a commun noun or an adjectif, the orange color with "o" in lowercase. However, "Orange" with "O" in uppercase could be various types of named entities: a city (in France), a telecommunication company or a noble family name.

[1] https://en.wikipedia.org/wiki/Twins_(disambiguation)
[2] https://www.kissonline.com/end-of-the-road

For coreference tasks, all referring expressions that refer to all named entities are first extracted, including pronouns like "he" for a person that was mentioned before. In a larger sense, descriptions like "the father of Gargantua" is also considered as a referential expression.

Entities which refer to the same object are then categorized as referring to the same object thanks to a unique identifier

Sometimes identical word sequences do not refer to the same "entity". For example, the referential expression "the president of America" referred to Barack Obama from 2009 to 2017 and since 2017, the same character string refers to Donald Trump. And even though we can imagine that there are other persons named Donald Trump, who are not the one of the president of America. That means, the same character sequence may refer to different entities (person).

Coreference will not be addressed in this thesis, so pronouns will not be considered, neither do noun phrases as referential expression.

THE FOOTBALL FEVER: Ohio head coach Frank Solich says Ohio State knows they have a special team and season underway

FIGURE 2.1: Named Entity "Ohio"

"Ohio" is often considered as a location name (a state name in the USA).

However, figure 2.1 shows an ambiguous case where the named entity "Ohio" in blue is a sport team name and the "Ohio" in red refers to the state inhabitants.

In conclusion, named entities are mentions that refer to an object of some defined types like "person", "organisation" or "location", etc. They are often proper nouns or noun phrases and multi-words named entities often have each word beginning with an uppercase letter.

2.2.2 From a practical perspective

Synthesio works on E-reputation for clients like brands, local government (like cities), music group or sportsteam etc. These brand names, music group names and location names are considered as types named entities.

Therefore, in a larger point of view, information that we want to extract can be from any category, not only named entity. For example, the goal of Marty, Tian, and Tellier, 2014 is to find `color`, `material` and `brand` in products' description texts.

Sometimes the same name can refer to different aspects of the named entity.

"McDonald's" is often considered as a company. Table 2.1 shows a typical example in Tian et al., 2016 and different aspects of this named entity "McDonald's".

My McDonald's was still hot when it was served and it tasted delicious.	product
The room was too hot when we ate at McDonald's yesterday.	location

TABLE 2.1: Examples of "McDonald's"

In the first sentence, "McDonald's" refers to the products (food). But in the second sentence, "McDonald's" refers to a restaurant, a location.

Synthesio's requirement was to add metadata into character sequences which represent company names, product names etc in order to add medadata into index for Synthesio's search engine. However, the distinction between brands, companies and products is often far from clear.

A brand can be a company name or a product name. For example, "L'Oréal" can refer to the "L'Oréal" group name (of type `Company`) or their products under the brand of the same name, just like "BMW", which can be a company name or a brand name.

This context entails specific problems like "J'adore" (French for "I love"), being the name of parfum of brand Dior (a product name). Products are often associated with brands like "Citroën C6". "Citroën" is the brand name and a company name (a subsidiary of the PSA Peugeot Citroën group) ."C6" is the car model name (of type product) of the company.

Therefore, it is difficult for Synthesio to extract company names and product names. The Synthesio definition of named entities types is shown in chapter 3.

2.3 Domain and text genre

A text genre is a type of written or spoken discourse, characterized by the way it was created and by the register of language it uses, its audience, etc (Kessler, Numberg, and Schütze, 1997). Therefore, we distinguish poetries, journalistic texts, discourse transcriptions and User-Generated Content (UGC) as different text genres.

As most of NLP tasks (Part-Of-Speech tagging, chunking etc), Named Entity Recognition (NER) first processed on standard, well-formed texts as journalistic texts.

Texts of different genres like oral conversation transcription and User-Generated Content (UGC) on the Internet are then processed. There are also Named Entity Recognition (NER) works on texts in other specialised domains as medical notes, historical archive text and automobile product description.

For CoNLL 2003 (Tjong Kim Sang and De Meulder, 2003), the NER was based on well-formed texts (news) from Reuters (for English) and from German newspaper Frankfurter Rundshau (for German). The ACE 2004 corpus was extracted from broadcast news, newspaper and newswire data (Doddington et al., 2004).

In other domains, Quimbaya et al., 2016 tried to extract named entities (diagnosis, treatment and etc) from electronic health records (EHR) using a dictionary-based Fuzzy Gazetteer approach (by edit distance). Kumar et al., 2015 tested a CRFs-based NER system with clinical notes. More specifically, Wang et al., 2016 also employed a CRFs-based system to recognize entities (disease, disease type, complaint-symptom, test-result, test and treatment) from Chinese Electronic Medical Records and they achieved a F1 measure of 89.07%. Besides these medical text processing, Byrne, 2007 tested a NER task with historical archive text. Their definition of named entities are hierarchical, for example, the named entity type `Location` were divided into `PLACE` (cities, towns, etc). Their best result was up to about 77% on F1 measure.

With other text genre, in 2001, Poibeau and Kosseim, 2001 compared the results obtained using a standard NER system when applied to news (MUC-6 data) and to speech transcriptions. They reported a drop in F1 measure. Proper names extraction systems' (without adaptation) f1 measures drop between 0.25 (0.69 to 0.44 for Exibum system) and 0.40 (0.90 to 0.50 for Lexis system). Nowadays, more and more user-generated content (UGC) appears on Internet. With platforms such as forums, Facebook, Twitter, more and more topics are being discussed everyday. On 2011, Ritter et al., 2011 created an annotated NER corpus from Twitter and tested their CRFs-based NER system. Liu et al., 2012; Cano Basave et al., 2013; Derczynski, Yang, and Jensen, 2013; Erp, Rizzo, and Troncy, 2013 also developed NER systems with Twitter data.

As we can imagine, a NER system performs differently with texts in different domains and different text genres. Domain adaptation (DA) aims to learn a model from a source data and then to apply this model (often well preforming model) on a different (but related) target data. The target data is often from different domain or different text genre (Ben-David et al., 2010).

Guo et al., 2009 tested their DA model in English and Chinese with four domains: economics, entertainment, politics and sports. Alvarado, Verspoor, and Baldwin, 2015 tried NER domain adaptation (DA) with risk assessment. Finally, Tian et al., 2016 applied NER system trained from tweets with other long and short texts.

2.4 Taxonomy of named entities

As discusted previously, NER tasks often extract person names, company names and location names. Some systems also work with time, data, currency, number and percentages.

From 1996, in the Multi-lingual Entity Task (MET), the conference of Association for Computational Linguistics (ACL) (Merchant, Okurowski, and Chinchor, 1996) defined, as in MUC-6, three types of named entity: `person`, `organization` and `location`.

Later in 2002 and 2003, the CoNLL shared task added a `MISC` type for names of miscellaneous entities that do not belong to the previous three groups (Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003). We can find named entities of `MISC` type like: "Christmas" (holiday name), "A6 Quattro" (car model name of Audi), "Africa Cup 2006" (rugby union tournament in Africa or football championship of Africa), "Chicken salad", "Alien" (movie name) etc.

In French, the French Treebank (Abeillé, Clément, and Toussenel, 2003) is annotated with eight types of named entities: `Company`, `Organization`, `Location`, `Fiction-Character`, `POI` (Point Of Interest) and `Product`.

If we look at the User Generated Content (UGC), Ritter et al., 2011 created a corpus from Twitter which talked about various subjects. The developer of the corpus annotated 2,400 tweets with 10 types which are both popular on Twitter, and have good coverage in Freebase: `Person`, `Company`, `Geo-location`, `Product`, `Facility`, `Tv-Show`, `Movie`, `Music-artist`, `Sports-team` and `Other`.

In the medical domain, the named entities that we want to extract are very different. (Wang et al., 2016) used a CRFs-based system to recognize six entities: `disease`, `disease-type`, `complaint-symptom`, `test-result`, `medical-test` and `medical-treatment`.

Sometimes in biomedical texts, one type of entity can include another type of entity. For example in the character sequence "CIITA mRNA", the

string “CIITA” is a DNA name and the entire string “CIITA mRNA” refers to an RNA (Ribonucleic acid).

Finkel and Manning, 2009 presented a nested named entity recognition system for parsing and they tried to extract protein, DNA, RNA, cell line and cell type as named entities.

Byrne, 2007 worked with historical archive text and tried to extract 11 classes: `ORG`, `PERSNAME`, `ROLE`, `SITETYPE`, `ARTEFACT`, `PLACE`, `SITENAME`, `ADDRESS`, `PERIOD`, `DATE` and `EVENT`. Three of these, `SITETYPE`, `ARTEFACT` and `EVENT`, are further divided into subclasses.

In this thesis, we only talk about simple named entities. We will extract the most complete named entity in case of nested named entities. Synthesio’s definition of named entity types is shown in chapter 3.

2.5 Evaluation of Named Entity Recognition (NER)

To evaluate a Named Entity Recognition (NER) method, we first need a reference annotation (a benchmark). The problems with human annotation as benchmark will be discussed in chapter 3. The named entities extracted by the NER method are then compared with the reference annotations.

Criteria of a correctly extracted named entity are: the named entity’s type and boundaries: the beginning and end of the named entities.

For example, there is a cocktail name "Long Island Ice Tea" in a text and we want the Named Entity Recognition (NER) system to recognize this named entity. If the NER system only recognizes "Long Island" (a populated island off the East Coast of the United States), that will not be considered as a correct extraction because the end of the named entity is not correct.

Mr. <ENAMEX TYPE="PERSON">Dooner</ENAMEX> met with <ENAMEX TYPE="PERSON">Martin Puris</ENAMEX>, president and chief executive officer of <ENAMEX TYPE="ORGANIZATION">Ammirati & Puris</ENAMEX>, about <ENAMEX TYPE="ORGANIZATION">McCann</ENAMEX>’s acquiring the agency with billings of <NUMEX TYPE="MONEY">$400 million</NUMEX>, but nothing has materialized.

TABLE 2.2: An example of MUC-6 annotation

Figure 2.2 is an annotated example of MUC-6 from (Grishman and Sundheim, 1996). Each named entity is annotated with its type: `PERSON`, `ORGANIZATION` for `ENAMEX` and `MONEY` for `NUMEX` and their boundaries (the beginning and the end of each entity). The goal of MUC-6 is to find all the named entities (of type person, organization and location), time, plus currency and percentage

expressions using SGML. The tag ENAMEX is used by person, organization and location entities while the tag NUMEX is used for currency and percentage expressions.

Table 2.3 shows all possible errors that a NER system could make. We will use this example to explain how to evaluate a NER system with different evaluations.

Human annotation		System prediction		Comment
Type	Entity value	Type	Entity value	
PERSON	Dooner	PERSON	Dooner	Correct
PERSON	Martin Puris	PERSON	with Martin Puris	Boundaries error
ORG	Ammirati & Puris	PERSON	Ammirati	Boundaries and type errors
ORG	McCann	PERSON	McCann	Type error
MONEY	$400 million			Missing
		ORG	nothing	Spurious
		ORG	materialized	Spurious

TABLE 2.3: An example of a NER system prediction

2.5.1 MUC evaluation (partial match evaluation)

The MUC evaluation metrics is used in MUC-6 (Grishman and Sundheim, 1996) and MUC-7 (Chinchor, 1999). This evaluation metrics took into account partially correct extraction which means: an entity correctly recognized but with a wrong type and an entity with correct type but wrong boundarie(s) are considered as 1/2 correct answer. The second case with wrong boundaries concerns for example "Dooner" in the text and "Mr. Dooner" in the golden annotation or vice-versa. In the example of the table 2.3, the prediction "with Martin Puris" is considered as a partial correct answer for "Martin Puris" in the golden annotation.

With this evaluation metric, the true positive (number of named entities which are correctly extracted) is calculated as:

- 1: Dooner (PERSON)
- 0.5: Martin (PERSON) for Martin Puris (PERSON)
- 0.5: McCann (PERSON) for McCann (ORGANIZATION)

The global precision is calculated as:

$$\frac{1 + 0.5 + 0.5}{\text{number of prediction}} = \frac{2}{6} \approx 0.33$$

The global recall is calculated as: 9

$$\frac{1 + 0.5 + 0.5}{\text{number of golden annotation}} = \frac{2}{5} = 0.4$$

Then the global F1-measure is calculated as mean average of precision and recall.

$$F1 = 2 * \frac{precision * recall}{precision + recall} \approx 0.362$$

This MUC evaluation is not so strict that they tolerate one error on either type or boundary. Named entities extracted with correct boundaries but with a wrong entity type are considered as a half of a true positive (0.5); named entities extracted with correct entity type but with a wrong boundary is also considered as a half of a true positive (0.5). Therefore, scores are higher than other more strict metrics.

2.5.2 CoNLL evaluation (exact match evaluation)

On IREX (in Japanese), CoNLL 2002, CoNLL 2003 and WNUT 2015, the evaluation method employed is exact match evaluation. That means, only named entities with correct boundaries and type that the system extracted are considered as positive answers. Either boundaries or type error is considered as wrong prediction.

The global precision is calculated as:

$$\frac{\text{true positive}}{\text{number of prediction}} = \frac{1}{6} \approx 0.17$$

The recall is calculated as:

$$\frac{\text{true positive}}{\text{number of golden annotation}} = \frac{1}{5} = 0.2$$

Then the F1-measure is calculated as mean average of precision and recall.

$$F1 = 2 * \frac{precision * recall}{precision + recall} \approx 0.184$$

2.5.3 SemEval shared task evaluation

In the International Workshop on Semantic Evaluation (SemEval) in 2003, a task of extracting Drug-Drug Interaction from BioMedical Texts (DDIExtraction) was proposed. It was divided into two parts: recognition and classification of drug names and Extraction of drug-drug interactions. The first part correspond to a NER task and four different evaluation measures were proposed, each requires different level of strictness.

- Strict: requires correct entity type and boundaries
- Exact: requires exact entity boundary matching regardless to type
- Partial: requires at least one boundary matching regardless to type
- Type: requires correct entity type when there is some overlap

We can see that the Strict evaluation correspond to the CoNLL 2003 evaluation: only extracted entities with correct type and two boundaries are considered as a true positive result. The Exact and Partial evaluations treat wrong types of entities as correct results.

The partial evaluation also tolerates entities with only one correct boundary. Lastly, type evaluation only insists the entity type as well as there is an overlap between the human annotation and the system prediction.

Table 2.4 shows the same exemple of a NER system prediction as table 2.3 but with more measures on the right. In each row, in the cases of scores, we show the result of the four different metrics.

For all metrics:

- COR: correct answer
- INC: the system's output and the gold-standard annotation disagree
- MIS: missing
- SPU: spurious

For partial evaluation, PAR means that the system's output and the gold-standard annotation are not identical but have some overlapping text. This category makes sense only if the partial match is allowed.

With the indications of the table 2.4, we can calculate that:

$$\text{POSSIBLE(POS)} = \text{COR} + \text{INC} + \text{PAR} + \text{MIS} = \text{TP} + \text{FN}$$
$$\text{ACTUAL(ACT)} = \text{COR} + \text{INC} + \text{PAR} + \text{SPU} = \text{TP} + \text{FP}$$

Human annotation		System prediction		Evaluation measures			
Type	Entity value	Type	Entity value	Strict	Exact	Partial	Type
PERSON	Dooner	PERSON	Dooner	COR	COR	COR	COR
PERSON	Martin Puris	PERSON	with Martin Puris	INC	INC	PAR	COR
ORG	Ammirati & Puris	PERSON	Ammirati	INC	INC	PAR	INC
ORG	McCann	PERSON	McCann	INC	COR	COR	INC
MONEY	$400 million			MIS	MIS	MIS	MIS
		ORG	nothing	SPU	SPU	SPU	SPU
		ORG	materialized	SPU	SPU	SPU	SPU

TABLE 2.4: Same prediction example with SemEval evaluation

where:

TP: True positive

FN: False negative

FP: False positive

Then evaluation results will be reported using the standard precision/recall like :

$$Precision = \frac{COR}{ACT}$$

$$Recall = \frac{COR}{POS}$$

Except that for Partial evaluation, the precision and recall will be calculated as:

$$Precision = \frac{COR + 0.5 * PAR}{ACT}$$

$$Recall = \frac{COR + 0.5 * PAR}{POS}$$

At last, the F1-measure is calculated as:

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

We then get the evaluation results in the table 2.5.

As we can see, Strict is the strictest evaluation and Partial is the least strict evaluation. Exact and Type evaluations are between these two.

The SemEval 2003 evaluation offered four evaluation measures with different strictness level. We can then evaluate different systems with more flexibility. We can distinguish systems that are more precise on entity boundaries (Exact) or more suitable for entity types (type). With partial evaluation, partly matched named entity (errors of boundaries) are considered as partly

Measure	Strict	Exact	Partial	Type
COR	1	2	2	2
INC	3	2	0	2
PAR	x	x	2	x
MIS	1	1	1	1
SPU	2	2	2	2
POS	5	5	5	5
ACT	6	6	6	6
Precision	0.17	0.33	0.5	0.33
Recall	0.2	0.4	0.6	0.4
F1	0.184	0.362	0.66	0.362

TABLE 2.5: SemEval 2003 Evaluation results

correct (for example,extracted "blue" from "dark blue"), not as wrong predictions as missing named entities. Different applications may require different strategies concerning accuracy and precision of named entity recognition.

In this thesis, we are going to use the CoNLL evaluation because it is the strictest evaluation for both entity boundaries and entity type.

2.6 Machine learning based NER methods

First NER systems were often based on rules. Regular expressions are defined to detect date, currency or time types. Person names, organisation names and location names were often extracted from a list of all typed named entities.

Machine learning is a study of algorithms and statistical models used in different NLP tasks including the Named Entity Recognition (NER).

There are three families of machine learning methods: supervised learning, semi-supervised learning and unsupervised learning.

Since annotating data is expensive and never exhaustive compared to real-world data we have to deal with, people may try to get use of large amount of unannotated, relevant data as a semi-supervised method or an unsupervised method.

2.6.1 Supervised learning

Supervised learning methods use annotated examples to learn a statistical model.

Simple supervised learning methods often used are: Maximum Entropy (MaxEnt) like in Bender, Och, and Ney, 2003; Chieu and Ng, 2003; Curran

and Clark, 2003, Hidden Markov Models (Florian et al., 2003; Klein et al., 2003; Mayfield, McNamee, and Piatko, 2003; Whitelaw and Patrick, 2003), Conditional Random Fields like in McCallum and Li, 2003, Neural Networks like LSTM in Hammerton, 2003; Murthy and Bhattacharyya, 2016.

Although sentences can be analyzed by syntactic trees as shown in figure 2.2 of Yule, 2010, in machine learning, texts are often treated as linear sequences of tokens. Named Entity Recognition (NER) is then considered as a sequence labelling problem.

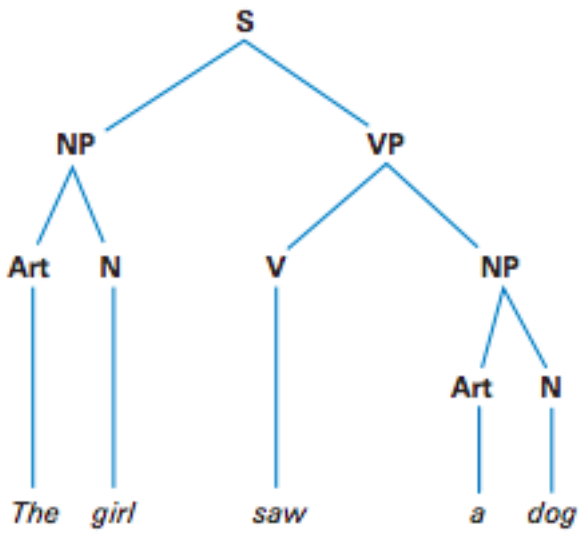

FIGURE 2.2: Syntactic analysis by tree diagram

Raymond and Fayolle, 2010 tested two methods with two corpora. They found that the CRFs performed better than Support Vector Machines (SVM) (Cortes and Vapnik, 1995), which was itself already more efficient than a finite state automaton.

Putthividhya and Hu, 2011 also reported that the CRFs worked better than Decision Trees (Quinlan, 1986), SVM, Maximum Entropy (MaxEnt) and HMM (Hidden Markov Model) with the task of extracting product attributes in the texts of description of cloth and shoe in eBay (`http://www.ebay.com/`).

Conditional Random Fields (CRFs) (Lafferty, McCallum, and Pereira, 2001; McCallum and Li, 2003) is considered as the best supervised model in resolving sequence labelling problems including Named Entity Recognition (NER) tasks (McCallum and Li, 2003; Ritter et al., 2011).

Conditional Random Fields (CRFs)

CRFs is a type of discriminative undirected probabilistic graphical model that can be used to segment and label sequence data (Lafferty, McCallum, and Pereira, 2001; Tellier and Tommasi, 2011).

For the NER task, CRFs that we employed are linear chain as in Sha and Pereira, 2003. This is a supervised training method; that means, as training data, we already have a set of label sequences Y corresponding to text sequences X (in the form of tokens).

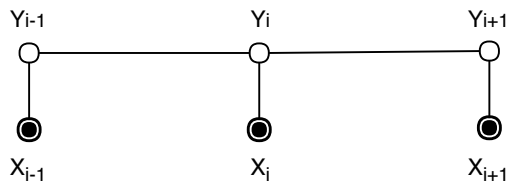

FIGURE 2.3: Linear CRFs

In graph theory, a graph is a set of vertices (also called nodes) which are connected by edges nodes (Bondy and Murty, 2008).

Figure 2.3 shows a linear CRFs model from Lafferty, McCallum, and Pereira, 2001. In this graph, X_{i-1}, X_i, X_{i+1}, Y_{i-1}, Y_i and Y_{i+1} are vertices (nodes) and lines which connect vertices are edges.

X are (considered as) independent observations and correspond to the sequence of tokens of the text. The order of tokens is from left to right, that is from $i-1$ to $i+1$ ($i > 0$) in the figure.

Y is a sequence of labels. Each label corresponds to one observation of X in the sequence (for example Y_i correspond to X_i, etc).

The prediction of a current Y_i may depend on the whole X sequence and the previous label Y_{i-1}.

The prediction is based on combinations of feature functions. The feature functions modelize properties local to the tokens sequence (observations).

The labels prediction for an X sequence, is to find the best sequence of labels Y given the sequence of observations X, that is, to find a sequence of labels Y which maximizes the probability $P(Y|X)$.

This probability, depending on combinations of feature functions, is expressed with the formula:

$$P(Y|X) = \frac{1}{Z(X)} \prod_{c \in C} exp(\sum_k \lambda_k f_k(c, Y_c, X)) \tag{2.1}$$

Z is a normalization factor over all output values. It is the same for all possible Y sequences so it can be ignored when we calculate the most probable Y sequence. "C" is for cliques, and "c" is one clique in the set of cliques "C". A clique in an undirected graph is a subset of its vertices such that every two vertices in the subset are connected by an edge (Luce and Perry, 1949).

$f_k(c, Y_c, X)$ represents a feature function (for one clique "c"), which returns a value of 0 or 1 according to the conditions in the definition. λ_k is the weight associated to each feature function $f_k(c, Y_c, X)$.

There are two types of feature functions that correspond to two types of cliques in the figure 2.3: uni-gram and bi-gram. The uni-gram feature functions return a value which depends on part of the entire set of observations and its current label.

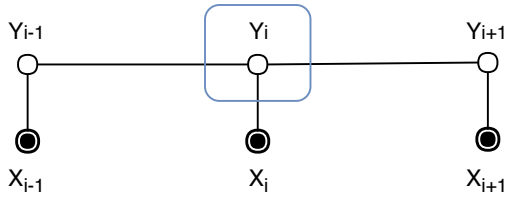

FIGURE 2.4: An unigram CRFs

The blue square in the figure 2.4 shows an unigram CRF, that is one type of cliques). The feature functions for an unigram model will be:

$$f(i, Y_i, X) = \begin{cases} 1 & \text{if } Y_i = \text{label value,} \\ & \text{and } X = \text{token value} \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

The probability of a label sequence Y for this unigram CRF is calculated like this:

$$P(Y|X) = \frac{1}{Z(X)} \prod_{i=1} exp(\sum_k \lambda_k f_k(i, Y_i, X)) \tag{2.3}$$

The bi-gram feature functions take in input the observations sequence X, the current label Y_i and the label of the previous observation: Y_{i-1}.

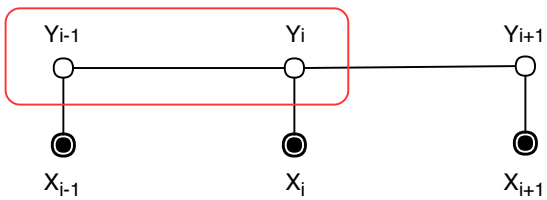

FIGURE 2.5: A bigram CRFs

The red square of figure 2.5 shows a bigram CRF (that is the other type of clique in CRFs sequence) and feature functions for a bi-gram model will be:

$$f'(i, Y_i, Y_{i-1}, X) = \begin{cases} 1 & \begin{array}{l} \text{if } Y_i = \text{label value,} \\ Y_{i-1} = \text{the previous label value,} \\ \text{and } X = \text{some values} \end{array} \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

If we consider the two types of cliques (which are expressed by feature functions) at the same time for the same model, the probability of the sequence of labels Y is calculated like this:

$$P(Y|X) = \frac{1}{Z(X)} \prod_{i=1} exp(\sum_k \lambda_k f_k(i, Y_i, X)) \prod_{i=2} exp(\sum_{k'} \lambda'_{k'} f'_{k'}(i, Y_i, Y_{i-1}, X)) \tag{2.5}$$

The objective is to find the best label sequence Y which maximizes this probability $P(Y|X)$.

An example of CRF training

Linear CRFs are designed to label text sequences. A tokenized text is often considered as a linear sequence ignoring syntactic dependencies.

The supervised training of linear CRFs is based on a pair of two sequences: the sequence of tokens and the corresponding sequence of labels, and a pattern which defines feature functions.

The generated model then fixes weights for each feature function by maximum likelihood estimation (MLE) and stochastic gradient descent (SGD). Calculation of the most probable labels sequence for a sequence of tokens is done with the Viterbi algorithm (Forney, 1973).

Training

In this example, we have a text "Welcome to Paris". We want to recognize the named entity of the city "Paris". We will have an annotation for this text like the figure 2.6 which follows the pattern of the CRF definition of figure 2.4: the text (tokens) "Welcome to Paris" in the bottom, annotation on the top.

Each label corresponds to a token of the text just under it. One of the possible label for annotation system is BIO (Ramshaw and Marcus, 1999): "B" for "Beginning", "I" for "Inside" and "O" for "Outside". In what follows, we use this BIO label system. In our Named Entity Recognition (NER) task,

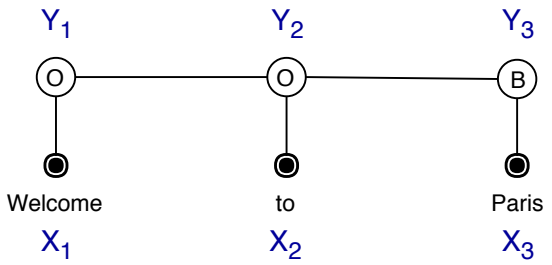

FIGURE 2.6: An annotated text

possible labels are "O" and combinations of "B" and "I" with named entity types. "B" followed by a hyphen "-" and the entity type for the token is the beginning of a named entity of the entity type that we want to recognize.

"I" followed by a hyphen "-" and the entity type for the token is inside of a named entity.

For example, if we have "New York" as a named entity of type `Location` to annotate, we will have label "B-location" for the token "New" and the label "I-location" for the token "York".

In the example of figure 2.6, "Paris" is a location name that we want to extract, so it is labelled with "B-Location", but we use "B" for short. "O" is for "outside", which means the corresponding token is not interesting (not a named entity to be extracted).

We then define a simple unigram CRF pattern for illustration: the current label depends only on the previous token value. In real CRF patterns, we can define bigram patterns using the previous label and the current label. The pattern about observations X can also be features about the token.

In our example, if the previous token is "to", the current label will be "B-Location". Other labels are all "O". That is to define a rule that the token that follows the token "to" is a location name.

The defined pattern (in red) will move from the beginning of the text until the last token of the text like in figure 2.7. The "$" is the beginning of a sentence.

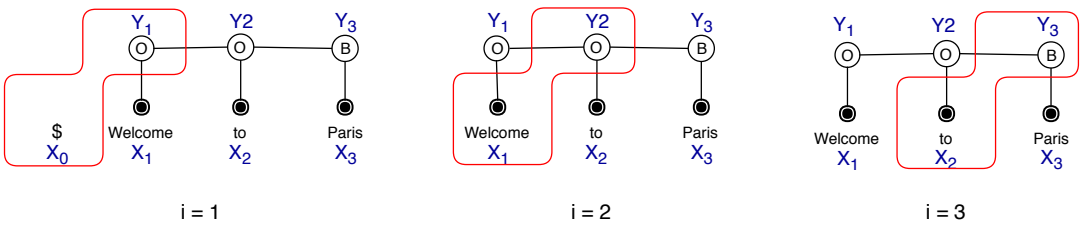

FIGURE 2.7: The pattern's movement

With the annotated text and the defined pattern, the CRFs then creates three corresponding feature functions:

$$f_1(i, Y_i, X) = \begin{cases} 1 & \text{if } Y_i = \text{"O" and } X_{i-1} = \$ \\ 0 & \text{otherwise} \end{cases} \tag{2.6}$$

$$f_2(i, Y_i, X) = \begin{cases} 1 & \text{if } Y_i = \text{"O" and } X_{i-1} = \text{"Welcome"} \\ 0 & \text{otherwise} \end{cases} \tag{2.7}$$

$$f_3(i, Y_i, X) = \begin{cases} 1 & \text{if } Y_i = \text{"B" and } X_{i-1} = \text{"to"} \\ 0 & \text{otherwise} \end{cases} \tag{2.8}$$

We can see that CRFs are based on three feature functions, that is the number of tokens in the training phase if there is no repetition. For each feature function, two conditions (defined with the pattern) are required. If the case satisfies the defined conditions (from annotated examples), feature functions return the value 1 otherwise they return the value 0.

In this example, we can define all feature functions weights (λ_k with k being from 1 to 3) as 1 (the real weights are first initialized and then adjusted using maximum likelihood estimation (MLE)).

The set of feature functions and their weights forms a CRFs model. The defined pattern and annotated examples are already included in the definition of feature functions.

Applying the trained CRFs model

We now have a CRFs model trained by our example "Welcome to Paris" and we want to apply it to other texts like in the figure 2.8.

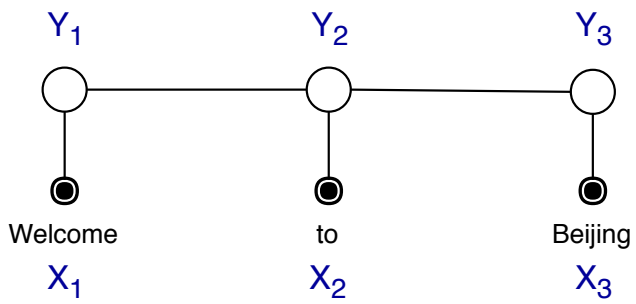

FIGURE 2.8: A text to label

When we apply feature functions with the new text "Welcome to Beijing", we will test all possible labels values. Here we have only two possible labels: "B-Loc" (B) and "O".

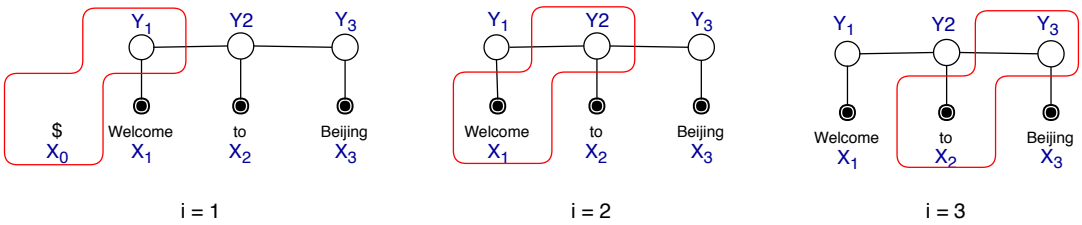

FIGURE 2.9: The pattern's movement in prediction

As shown in figure 2.9, when our defined pattern (in red) moves from the beginning to the end of the text, we can obtain results like table 2.6 (we use "B" for "B-Loc").

Y_0	Y_1	Y_2
i=0 X_{i-1} = '$'	i=1 X_{i-1} = 'Welcome'	i=2 X_{i-1} = 'to'
$f_1(i, Y_i =' O', X_{i-1})$ = 1 $f_1(i, Y_i =' B', X_{i-1})$= 0	$f_1(i, Y_i =' O', X_{i-1})$ = 0 $f_1(i, Y_i =' B', X_{i-1})$= 0	$f_1(i, Y_i =' O', X_{i-1})$ = 0 $f_1(i, Y_i =' B', X_{i-1})$= 0
$f_2(i, Y_i =' O', X_{i-1})$ = 0 $f_2(i, Y_i =' B', X_{i-1})$= 0	$f_2(i, Y_i =' O', X_{i-1})$ = 1 $f_2(i, Y_i =' B', X_{i-1})$= 0	$f_2(i, Y_i =' O', X_{i-1})$ = 0 $f_2(i, Y_i =' B', X_{i-1})$= 0
$f_3(i, Y_i =' O', X_{i-1})$ = 0 $f_3(i, Y_i =' B', X_{i-1})$= 0	$f_3(i, Y_i =' O', X_{i-1})$ = 0 $f_3(i, Y_i =' B', X_{i-1})$= 0	$f_3(i, Y_i =' O', X_{i-1})$ = 0 $f_3(i, Y_i =' B', X_{i-1})$= 1

TABLE 2.6: Feature functions values

Until now, we have all values of feature functions. The next stage is to compute the most probable Y sequence depending on X: $P(Y|X)$. The table 2.7 shows all combination of Y_i values and feature functions values f_i for i from 0 to 2 and the sum of the values.

	f_1	f_2	f_3	$\sum_k \lambda_k f_k(i, Y_i, X)$
Y_0 = 'O'	1	0	0	1
Y_0 = 'B'	0	0	0	0
Y_1 = 'O'	0	1	0	1
Y_1 = 'B'	0	0	0	0
Y_2 = 'O'	0	0	0	0
Y_2 = 'B'	0	0	1	1

TABLE 2.7: Feature functions summations for each Y position

Finally, the table 2.8 shows all the possible Y sequences and the best path which maximizes the value of $P(Y|X)$.

We can see from the table 2.8 that the label sequence "OOB" gives the maximum value of $P(Y|X)$, which means the CRFs extracted the location name "Beijing".

Y_0	Y_1	Y_2	$p(Y\|X) = \prod_{i=1} exp(\sum_k f_k(i, Y_i, X))$
B	B	B	$e^{0+0+1} = e^1$
B	B	O	$e^{0+0+0} = e^0$
B	O	B	$e^{0+1+1} = e^2$
B	O	O	$e^{0+1+0} = e^1$
O	B	B	$e^{1+0+1} = e^2$
O	B	O	$e^{1+0+0} = e^1$
O	**O**	**B**	$\mathbf{e^{1+1+1} = e^3}$
O	O	O	$e^{1+1+0} = e^2$

TABLE 2.8: All possible sequences and the best path

This is a CRFs model only trained with the example "Welcome to Paris". The trained model will extract the token just after the token "to". In the text "Ready to go", the model will extract "go" as a location name. That means more features about the token should be defined. For example, location names often begin with a capital letter, words before a location names could be "to", "from", "in" and so on.

The strongpoint of linear CRFs is that we can use any observation X, because feature functions have access to the whole observation sequence. Instead of defining token value equals to some value, the features could also be whether the token begins with a capital letter, whether the token is in a word list, the value of its part-of-speech tag, etc.

For real application, feature functions are often used to model local properties of the current token. These functions are defined according to the pattern, which defines a conjunction of basic tests about the tokens sequence properties. In addition, an observation window controls the locality of feature functions.

2.6.2 Unsupervised learning and Semi-supervised learning

Unsupervised learning

Evans, 2003 employed first a hyponyms extraction method. For each word "X" in the text that begin with uppercase (potential named entity, or head of a phrase), launch a Google (`www.google.com`) query "such as X" and collect the 1000 words appearing just before the query "such as X". These words are considered as X's hyponyms (Hearst, 1992). These hyponyms are then clustered and the clusters are labelled.

Many other works consist of named entity clustering. That is, the extracted named entities (with boundaries strings) are classifies into different

clusters and then the cluster types are named. Algorithms often used in this sort of clustering are KNNs (Altman, 1992) or more recently LDA (Blei, Ng, and Jordan, 2003).

Semi-supervised learning

Semi-supervised learning methods are often based on supervised learning methods and the objective is to complete the supervised learning models. As introduced in Putthividhya and Hu, 2011, we followed the procedure to first learn a model with annotated examples, then we can modify or complete the model with unannotated data. Since annotating data is costly, the tendancy is to employ unannotated data to improve supervised models.

Boostrapping learning

In this thesis, we will talk about a semi-supervised method with boostrapping (Putthividhya and Hu, 2011). That means, we first learn a model with annotated data. Then we use this model to predict other unannotated data, and use the predicted data as annotated data to learn another model and repeat the same procedure.

2.7 Text normalization for User Generated Content

User Generated Content (UGC) is becoming more and more important in opinion analysis through the Internet. Because of the growing quantity of the mass of information, more and more studies concentrate on how to process the data automatically and efficiently.

However, almost for all Natural Language Processing (NLP) tools, the performances declined on User Generated Content (UGC).

For example, the Standford POS tagger, trained by the Penn TreeBank (Toutanova and Manning, 2000), reached 96.86% in accuracy on well-formed texts but only 81.3% on tweet texts (Ritter et al., 2011).

Similar results are found on syntactic analysis (Plank et al., 2014; Kong et al., 2014). and also on Named Entity Recognition (NER) as in Ritter et al., 2011.

alpha	EXPN	abbreviation
	LSEQ	letter sequence
	ASWD	read as word
	MSPL	misspelling
NUMBERS	NUM	cardinal number
	NORD	ordinal number
	NTEL	telephone (or part of)
	NDG	number as digits
	NIDE	identifier
	NADDR	number as street address

TABLE 2.9: Taxonomy of NSW in (Sproat et al., 2001)

2.7.1 Definition and scope of lexical normalization

The definition of text normalization is first proposed in Sproat et al., 2001. The problem that they try to resolve is to recognize all token forms from written texts in systems like Text-to-speech synthesis systems. So their definitions of text normalization is to turn all texts into words which can be found in a dictionary and can be pronounced correctly by a human that speaks that language. All tokens which need to be transformed are called "Non-Standard Words". These "Non-Standard Words" contain numbers, abbreviations, dates, currency amounts and acronyms.

Later, Han, 2014 distinguished Non-Standard Words and Out-Of-Vocabulary words. Some Out-Of-Vocabulary words could be named entities like "Obama" which is absent of the standard dictionary. These Out-Of-Vocabulary words could not be replaced by other words. At the meantime, some In-Vocabulary words could be incorrect according to their context like "wit" in "I will go wit you". That is a Non-Standard Word which should be corrected.

The 2015 WNUT workshop focused on named entity recognition and normalization of noisy text. They followed the definition of Non-Standard Words of Han, 2014. In addition, they also kept unchanged noisy Out-Of-Vocabulary in tweets, like "hahahahaha".

2.7.2 Definition and Taxonomy of Non-Standard Words

According to taxonomy defined in Sproat et al., 2001, Non-Standard Words (NSW) contain numbers, abbreviations, dates, currency amounts, acronyms etc.

Further more, in thesis Han, 2014, the author defined Non-Standard Words (NSW), compared them to standard words (SW) and showed relationships with Out-Of-Vocabulary words (OOV) and In-Vocabulary words (IV) in figure 2.10.

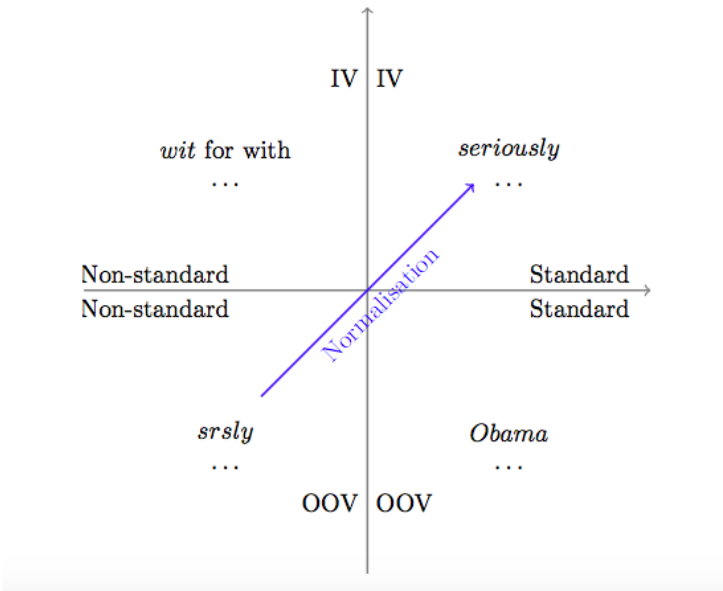

FIGURE 2.10: NSW and definition of normalization from Han, 2014

Figure 2.10 shows some examples of each class and the notion of normalization. All words are either In-Vocabulary (IV) or Out-Of-Vocabulary (OOV) according a reference standard English dictionary (here we use the union of the Aspell American and British English dictionary). Words are also divided into standard words (SW) and Non-Standard Words (NSW). However, Non-Standard Word could be In-Vocabulary words as in the example "wit" for "with you" shown in figure 2.10. On the other hand, some Out-Of-Vocabulary words are standard words, that do not need to be normalized. The example is a named entity, a proper noun "Obama", which does not exist in the standard English dictionary. But this is already the standard form that should not be normalized. We also find in this class tokens like "hahaha".

The objective of lexical normalization is to project Non-Standard Words (in vocabulary or Out-Of-Vocabulary words) into standard and In-Vocabulary words class. Here we only correct Non-Standard Words into In-Vocabulary words, but in reality, we should be able to correct them into Out-Of-Vocabulary words, like proposing "Obama" for the Non-Standard Word "Obamma".

They are often considered as standard words because we will not correct their surface form. But there are also exceptions for example sometimes "2" means "to" or "too" and "4" sometimes means "for".

In consequence, it is difficult to classify tokens into the four classes of the figure 2.10.

If we use one classifier to separate all tokens in a text, we can get tokens that we need to correct (Non-Standard Words, NSW) and tokens that we do not have to correct (standard words, SW).

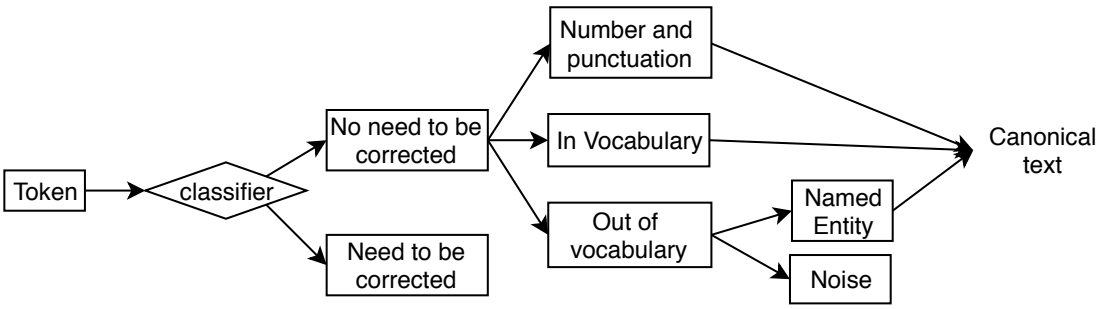

FIGURE 2.11: Details of SW words

Figure 2.11 describes this procedure. Standard words are tokens that do not need to be change. In this SW class, there are punctuations and numbers, there are also In-Vocabulary (IV) words and Out-Of-Vocabulary (OOV) words. In-Vocabulary (IV) words are comprehensive standard English words and Out-Of-Vocabulary (OOV) could be named entity (like "Obama" in figure 2.10), or simply some noise text (like "hahaha") for which we can not find a standard form. The variety of this SW class bring difficulties to the task of NSW/SW classification.

2.7.3 General procedure of lexical normalization

The ACL2015 workshop on noisy user-generated text (WNUT) proposed a text normalization challenge with an annotated dataset (Baldwin et al., 2015).

For this workshop, there were two distinct competitions: a constrained part, where participants develop their model without any external ressources and an unconstrained part, where participants were allowed to use external ressources and tools. The evaluation is done with F1 measure.

Reference	part	F1	Method
(Jin, 2015)	C	84.21	generate candidate with Jaccard similarity, random forest
(Supranovich and Patsepnia, 2015)	N	82.72	CRF for NSW/SW then lexicon
(Min and Mott, 2015)	C	81.75	LSTM + lexicon (labelled by edit distance)
(Leeman-Munk, Lester, and Cox, 2015)	C	81.49	char-level ANN for classification then ANN for training
(Berend and Tasnádi, 2015)	N	80.52	CRF to predict edit correction
(Akhtar, Sikdar, and Ekbal, 2015)	C	80.05	CRF for NSW/SW then rules
(Beckley, 2015)	N	75.71	Lexicon+rules+ranker

TABLE 2.10: Methods of WNUT workshop

Table 2.10 showed the four first teams in constrained part and the three first teams in unconstrained part, their results in F1 measure and their major methods.

(Liu, Weng, and Jiang, 2012) proposed a dataset of text normalization for tweets. They are generated from unsupervised tweets with context.

```
923 u | you
231 n | and
206 ur | your
199 dont | don't
158 b | be
138 w | with
131 ya | you
122 da | the
105 wit | with
100 r | are
92  thats | that's
88  ppl | people
86  bout | about
82  cuz | because
80  jus | just
77  dat | that
64  dnt | don't
63  c | see
59  wat | what
55  y | why
```

FIGURE 2.12: Extract of (Liu, Weng, and Jiang, 2012) dataset

Figure 2.12 showed an extract of this dataset. In this dataset, not only the NSW word and its corresponding correct form are shown, but also the number of occurrence for each NSW word.

But this dataset also shows that some NSW words have more than one correct forms and this depends on the context like "2" can be "too" or "to".

From these articles of WNUT workshop, we can see the general procedure of lexical normalization. First, the classification of SW/NSW words finds words to be corrected. In this step, we can use a dictionary to help finding OOV/IV words. Then the classification is often done with supervised learning methods. The corrector will only be executed on NSW words, as shown in figure 2.13.

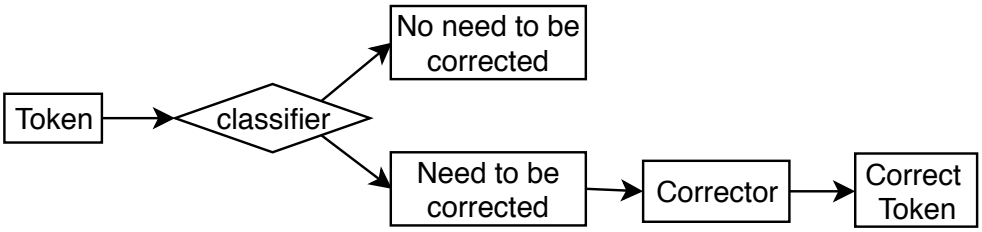

FIGURE 2.13: First Classify then correct NSW

Figure 2.14 showed another procedure: the corrector propose for all token in the text, a list of candidates to replace token, including the token itself, in the case that the token is SW. This model will be trained by all NSW and SW words.

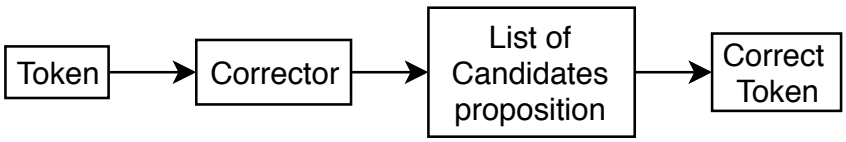

FIGURE 2.14: First propose candidates then sort

Reference	Algorithm	F1 (cv train | test for WNUT)	
Han and Baldwin, 2011	SVM	71.2%	
Supranovich and Patsepnia, 2015	CRF	86%	82%
Akhtar, Sikdar, and Ekbal, 2015	CRF	92%	86%
Leeman-Munk, Lester, and Cox, 2015	character-level ANN	98%	98%

TABLE 2.11: SW/NSW classification references and results

Table 2.11 shows references and results of SW/NSW classification. We followed Leeman-Munk, Lester, and Cox, 2015 to use neural networks for text normalization. The neural network will be explained later in this chapter.

This classification step is not always necessary, as shown in the general procedure 2.14.

Secondly, for each NSW words, some candidates are proposed. In this step, supervised learning methods are used to train a model and then to predict a candidate word from a vocabulary set. As for creating a training corpus, Gouws, Hovy, and Metzler, 2011 used vectors' similarity from 2 corpus. Liu et al., 2011a used Google request with OOV words' context. Then Liu, Weng, and Jiang, 2012 added more filter based on previous method, and generated candidates by trained model form 3802 pairs of NSW and its correction. Li and Liu, 2014 used a word embedding model (word2vec) with some more complexe formulas. Training algorithm also varise from machine translation in Pennell and Liu, 2011, Maxent as in Li and Liu, 2014, Character-level Neural Network as in Chrupała, 2014 and CRF (labelled by correction propositions) as in Berend and Tasnádi, 2015.

Table 2.12 showed two NSW words and corresponding CRFs labels. The original phrase is shown in 2.15. It correction is in 2.16.

I miss u my bie! Where u wanna out wif me?

FIGURE 2.15: An example of tweet

I miss you my bie! Where you wanna out with me?

FIGURE 2.16: The correction of the tweet example

letter	Feature examples			label
	v/c	prononciation	position	
^	N	N	B	correct
w	C	w	0	correct
i	V	i	1	correct
f	C	f	2	replace_by_th
$	N	$	E	correct
^	N	N	B	insert_yo
u	V	u	0	correct
$	N	$	E	correct

TABLE 2.12: Example of CRF labels for correcting NSW word

Features for each character that they used are: vowel or consonant (v/c), prononciation (represented by defined letters) and character position (B for begin, E for end and the number of character position in the word).

We can see that they defined manually rules to correct a NSW word. For "wif", they changed the position of "f" to "th" and for "u" they added "yo" at the position of "u".

The downside is that only defined correction rules could be predicted. If the correcting rule is not in the training data, the CRFs model could not predict the right label. For example, in the training data, we have "ASAP" rewritten as "as soon as possible", the CRFs model could not propose "as quickly as possible" for "AQAP". On the other hand, this method considers only the word without context, so they can not resolve the ambiguity problem.

Some unsupervised learning methods are also used, to generate a word using alphabet character. Han and Baldwin, 2011 used edit distance, double metaphone etc as supplement rules. If there was not the classification step, candidates are proposed for all tokens, including the surface form in the original text.

The last step is a ranker which determines the most probable candidate for a word to correct. The criterion of ranking could be strings similarities (word without context notion) like edit distance and distance between word vector representations, sequence probability (with Viterbi), or by trained language model Han and Baldwin, 2011. If there was not the classification step, the most probable candidate could be its surface form in the original text.

Jin, 2015 proposed a candidate generation method and a ranking by features. First, they generated candidates including the form of the token, all corrections for this form in the training data and the top m forms the most similar according to Jaccard Index Jaccard, 1912 similarity. Then they employed a random forest classifier with context-free features like:

- support: number of times that the token appeared
- confidence: the confidence of the form being normalised to a form c
- the similarity Jaccard Index with the original form
- string length

and features with context:

- pos-tagging confident: if the probability of the tagger is more with c than the original form
- look up of tag-1 and tag 0 for a local context

Their conclusion is that the most important features to evaluate a candidate are support and confidence, then the pos tagging.

Min and Mott, 2015 employed a contextual long-short term memory (LSTM) recurrent neural network. They also made use of the CMU Twitter POS tagger. They trained methods for normalization from Levenshtein distance algorithm, that is, before each character, possibilities of operations "insert", "replace" and "delete".

First, according to annotation principle, they changed all tokens into lowercase, and they kept usernames(@username), hashtags (tokens begin with #) and URLs unchanged. Then they created according to training data, pairs of word to correct and its non ambiguous correction.

And they trained their LSTM model with the first three characters of its POS tag plus all characters of the word. As context, they took one word before and one word after. As input, the had the representation of binary characters (in total 67 different characters), and they trained a character embedding using a linear projection layer while training the text normalization LSTM model. As output, they had one label for the current word, for example, they had a label as "insert_t_replace_h" for normalising "dese" into its correction "these".

Their conclusion is that using the context of one previous word and one next word is better than context-free model, but they did not exclude larger contexts.

Leeman-Munk, Lester, and Cox, 2015 employed a context-free, feed forward neural network without POS tag informations.

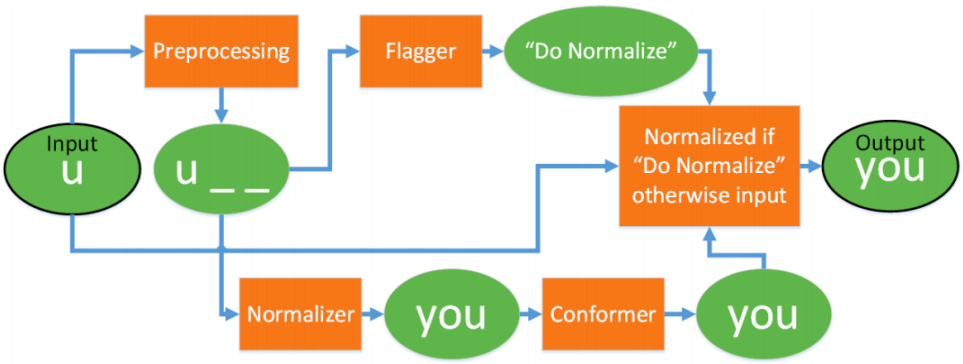

FIGURE 2.17: normalization processing in (Leeman-Munk, Lester, and Cox, 2015)

In the figure 2.17, they used an example to normalise "u" into "you". They first supposed that all words need to be normalised. Their "Normalizer", a neural network takes input as characters presented by vectors (as in Collobert et al., 2011) and through one hidden layer and proposes for each character position, the most probable character to reach the candidate "you". Then the "Conformer", uses Levenshtein distance (Levenshtein, 1966) with words in dictionary (generated from correct tokens of training data) to choose the most probable candidate "you". The independant "Flagger", another feed forward neural network, decides if we normalize or not. Their conclusion is, more dimensions in character presentation vector is more important than more layers in the neural network, and more iterations give always more satisfying result.

Liu et al., 2011a proposed a normalization method trained by auto-correction of google search as shown in figure 2.18. They collected the Out-Of-Vocabulary word "4got" in the request, and got the word in blod-italic "forgot" from the text "Including results for i ***forgot*** my password" in the web page as a pair of Non-Standard Word and its correction ("4got", "forgot").

They first applied a LangID model to filter English Twitter messages from 97 million Twitter messages from Edinburgh Twitter corpus (Petrović, Osborne, and Lavrenko, 2010).

Comparing to the GNU Aspell English dictionary [3], they found all Out-Of-Vocabulary (OOV) words. Then for OOV consisting of letters and apostrophe, they typed 6 possible "w1 w2 w3 OOV" or "OOV w1 w2 w3" sequences in google (w1, w2, w3 are context of OOV), kept the first 32 returned

[3] http://aspell.net/

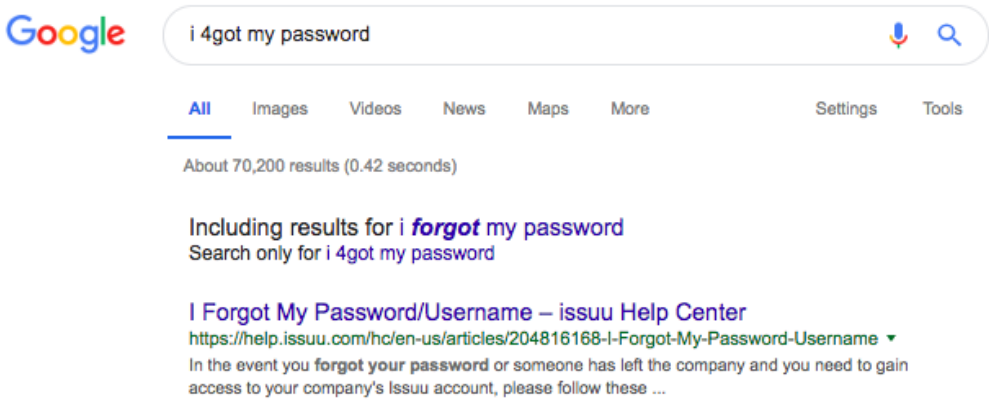

FIGURE 2.18: Screen shot for Google's auto-correction

snippets where words in boldface as candidates. For OOVs consisting of both letters and digits : they used rules like "1" rewritten as "one", "2" rewritten as "to" before using google query.

After some post-treatement, they obtained (NSW, SW) pairs and they created a character-level alignment for each (NSW, SW) pair by longest substring. Then they used alignment to train a CRFs model in order to generate NSW words for each SW word : P(NSW | SW) and they tested their method using Twitter and SMS data.

They evaluated their model by the best candidate and the first 3 best candidates comparing to simple dictionary method and spelling check system.

Liu, Weng, and Jiang, 2012 employed enhanced letter transformation to calculate P(NSW | SW), then generated a list of candidates from SW in the vocabulary using "Edit Distance". Then they selected pairs by context-aware training pair for candidates' ranking (using TF-IDF) and a character-level sequence labelling using CRFs as in (Liu et al., 2011a). They used visual priming approach ("déjà vu") by adding more weights for NSW in training data. They also employed the Jazzy spell checker[4] When combing candidates, they tested a word-level only method and a message-level method (by Viterbi algorithm (Forney, 1973)).

Chrupała, 2014 used a CRF (wapiti, LBFG) model for Levenshtein edit labelling at character level. His CRF labels included NIL (no edits), DEL (delete character at this position), INS(insert specified characters) etc.

This CRF model took usage of a character-level text embeddings. A model was first trained by simple recurrent networks (RNN) from 414 million bytes of UTF-8 text encoded in different languages at character-level for one tweet

[4] http://jazzy.sourceforge.net/

(a character sequence), the feature used in CRFs is the probability of the character at current position. They evaluated their model by word error rates (WER) and the result is outstanding from all combined dictionary method.

Park et al., 2016 published their results of classifying OOV terms (without their contexts) on a domain-specific social media corpus en automobile. They tries to classify into 9 categories: AUTO, DRUG, FOREIGN, MEASUREMENT, NE-AUTO, NE-OTHER, NOISE, SLANG, SPELLING-ERROR, or only 2 categories : NE-AUTO and OTHER (all the 8 rest).

Their OOV definition is:

1. frequency greater than 1000 in a automative corpus of 150 million posts
2. out of English dictionary Aspell
3. from 2 to 10 characters

Their CRF features contain:

- Character Ngrams (N from 1 to 3),
- language model character level (bigram and trigram models from English German and Spanish corpora)
- word frequency

They employed a word embedding by training a proper word2vec model with 8 billion tokens from Forum, Wikipedia and Twitter. The absent OOVs use average of all other OOV representations. They obtained a dataset of 665 OOVs, and they evaluated their model with 10-folds cross validation. They tested random forest and found that logistic regression (maximum entropy) works better. Their conclusion is that the proper pre-trained word embedding is useful for classification.

Neural Networks (NN)

Since Leeman-Munk, Lester, and Cox, 2015 has obtained a 98% on F1-measure for Non-Standard Words detection as in 2.11,

We followed this approch of neural networks for text normalization.

As its name suggests, neural networks were inspired by the brain's computation mechanism in neuroscience, which consists of computation units called neurons.

As shown in figure 2.19, neurons are connected with each layer (input layer and output layer in the figure) and neurons are not connected between

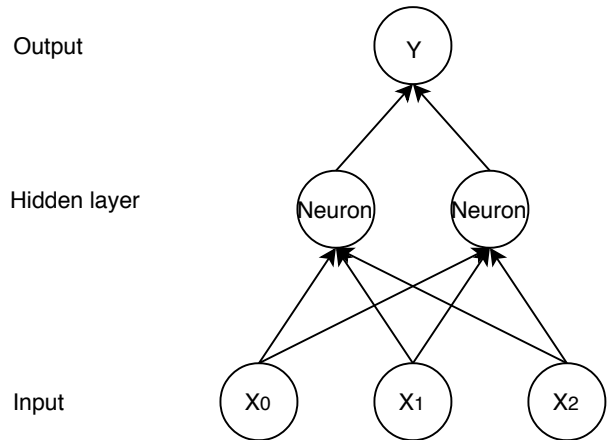

FIGURE 2.19: a simple neural network in general

them. In figure 2.19, there are two neurons in the middle level (hidden layer). They are not connected one to another but only with the input layer X sequence and the output layer Y, according to the definition of a simple multi-layer neural network.

The input and output layer could be defined by any numbers of dimensions, here we use only one hidden layer for an example. In the figure, the input layer X has three dimensions: X_1, X_2 and X_3. The output layer Y has only one dimension: Y. Each neuron represents a transformation from the lower layer to the higher layer.

With a lot of annotated data (pair of X sequence and Y) and a defined neural network like in figure 2.19, we can get a model capable to calculate a Y for a given sequence X.

The simplest neural network: The Perceptron

The perceptron is a binary classifier which decides, whether a vector of numbers (input) belongs to one class or not (Rosenblatt, 1957).

The perceptron function is presented as only one neuron, its input is a vector of numbers (of any dimensions), its output is binary: 0 or 1 here but it could also be 1 and -1.

The figure 2.20 shows a simple perceptron where m is the number of input vector dimension (or size). w is a vector of real-valued weights, each dimension w_i corresponding to x_i. $w = w_0, w_1, w_2, ..., w_m$. "SUM" means the function of dot product: $W^T X = \sum_{i=0}^{m} w_i \cdot x_i$. Here $x_0 = 1$, w_0 being the bias, does not depend on any input value. f is a function that receives the value

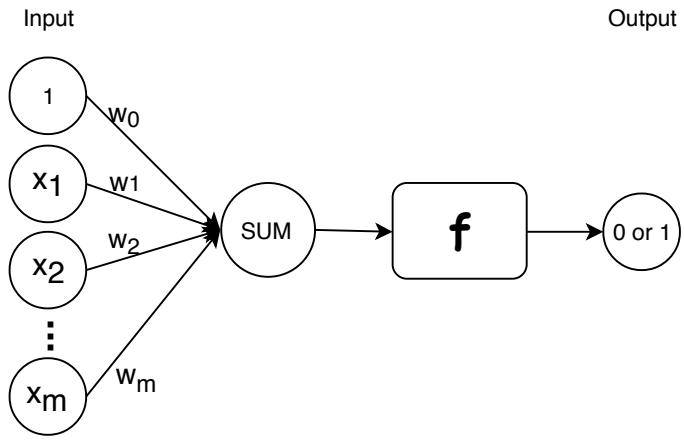

FIGURE 2.20: A perceptron

of "SUM" and returns the value 0 or 1. Here we define

$$f(v) = \begin{cases} 1 & \text{if } v > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.9}$$

We can see then the perceptron model is just the weights vector w. So the training step is to find the weights vector w with annotated examples.

Let vector $O = O_1, O_2, ..., O_n$ being n annotated examples and $A = A_1, A_2, ..., A_n$ the corresponding labels. The labels values are either 0 or 1. (A_j=0 or 1 for all j from 1 to n). Here each O_j represents a vector of $m + 1$ dimensions, which corresponds to the vector X in figure 2.20 ($O_j = x_0, x_1, x_2, ..., x_m$ with $x_0 = 1$).

To obtain the best w matrix, we first need to initialize the vector w. We can initialize it with all 0, or a small value superior to 0.

Let $w(t)$ being the actual (of the moment t) weights vector, it is a vector of $m + 1$ dimensions: $w(t) = w_0(t), w_1(t), w_2(t), ..., w_m(t)$ (initial weights w_0 for the first time). Then for each annotated example $O_j = o_{j,0}, o_{j,1}, o_{j,2}, ..., o_{j,m}$ where $o_{j,0} = 1$ for all j in n. We calculate the actual output $y_j(t)$:

$$y_j(t) = f(w(t) \cdot O_j) = f[w_0(t) \cdot o_{j,0} + w_1(t) \cdot o_{j,1} + ... + w_m(t) \cdot o_{j,m}]$$

Then we compare to the real value A_j (that $y_j(t)$ should be). We also define a $0 =< r <= 1$ called "learning rate", that means, with which probability we take each example into account. Then we can update the weights value. The next weights vector w is $w(t+1) = w_0(t+1), w_1(t+1), w_2(t+1), ..., w_m(t+1)$. We calculate each value $w_i(t + 1)$ of $w(t + 1)$ like this:

$$w_i(t + 1) = w_i(t) + r \cdot (A_j - y_j(t))o_{j,i}$$

And then we define an acceptable tolerance, the sum of errors threshold γ. The training procedure will stop when

$$\frac{1}{S}\sum_{j=1}^{S} \mid A_j - y_j(t) \mid <= \gamma$$

In the real training, we can browse more than one time all annotated examples and each time, a random order with all annotated examples to obtain the best result. So here S is the entire browsed examples set, it is the product of the number of iteration and n (number of annotated examples). That explains that t can also be superior than n.

We can see from the training procedure that there are configurations to define: initial weights vector, learning rate, error threshold and loss function (to calculate the difference of actual prediction value and the annotated value).

Convolutional neural network (CNN)

Convolutional neural networks use a variation of multilayer perceptrons designed to require minimal preprocessing (LeCun, 1989). The CNNs have been showing good performance for image classification (Krizhevsky, Sutskever, and Hinton, 2012). Compared to other algorithms, CNNs need less preprocessing because the network learns the filters and in traditional algorithms,the filters are often hand-engineered.

Then CNNs were used for natural language processing (Collobert and Weston, 2008), especially achieving excellent results in semantic parsing (Grefenstette et al., 2014), search query retrieval (Shen et al., 2014), sentence modeling (Kalchbrenner, Grefenstette, and Blunsom, 2014), classification (Kim, 2014) and other traditional NLP tasks (Collobert et al., 2011).

For sequence labelling task, CNNs are often used as the first layer of a more complete network, and represent local features moving from the head to the tail of a sequence. We first choose a window size representing the local context. For NLP, we often use a context of 3 words, that means, the current token, its previous token and its next token in the text. Sometimes, this context can be up to 5 words: the current token, two tokens before and two tokens after. When we study datasets from Baldwin et al., 2015 and Li and Liu, 2014 for token normalization, we find that the window of five words (current token, two token before and two token after) is capable of disambiguation of standard word and non-standard word. That is the raison

why we choose five as window size for token normalization. Here we use a window of size 3 for an example.

We use figure 2.21 to show how the window moves and how to calculate the output of the CNN layer from an input text of three words (phrase length $s = 3$) U = "Ready to go". In neural networks, all data is represented by vectors of numbers. Imagine that each word of the text is represented with a n dimension vector. We will have

$$U_1 = "Ready" = [u_{1,1}, u_{1,2}, ..., u_{1,n}]$$

$$U_2 = "to" = [u_{2,1}, u_{2,2}, ..., u_{2,n}]$$

$$U_3 = "go" = [u_{3,1}, u_{3,2}, ..., u_{3,n}]$$

Since we move the window of size $l = 3$ from the head to the tail of the 3-word-sentence. To consider each word to be the current word, we need representations for the head to the tail as follows:

$$HEAD = [h_1, h_2, ..., h_n]$$

$$TAIL = [t_1, t_2, ..., t_n]$$

Like in figure 2.21.

After the transformation by the window's movement, we obtain for this CNN layer the output of a matrix of $n * l$ rows and s colomns: $CNN(U)_{n*l,s}$.

Like mentioned before, the output of one neural network's layer is calculated as $output = w \cdot input$, where w is the weights matrix, $input$ is $CNN(U)_{n*l,s}$. So if the desired output dimension of this layer is o, the dimensions of the weights w will be $o * (n * l)$.

We often use other layers after the CNN's output, so we will need a vector (with only one colomn) but not a matrix. It means that we need an operation on $CNN(U)_{n*l,s}$ to reduce s to 1. Pooling operation is the method to reduce vector dimensions. Maxi Pooling operation is one of the pooling operation methods. For each row of $CNN(U)$, we keep only the highest value (meaning the most representative) of the s values as shown in the figure 2.23.

Then the output of this CNN and maxpooling layer will be another vector of $n * l$ dimensions, Which will be used as the input of the next layer.

As we can see, the principle of CNN is to locally observe each input (in its context) in order to determine the corresponding output. So this observation is enriched by the context of the input.

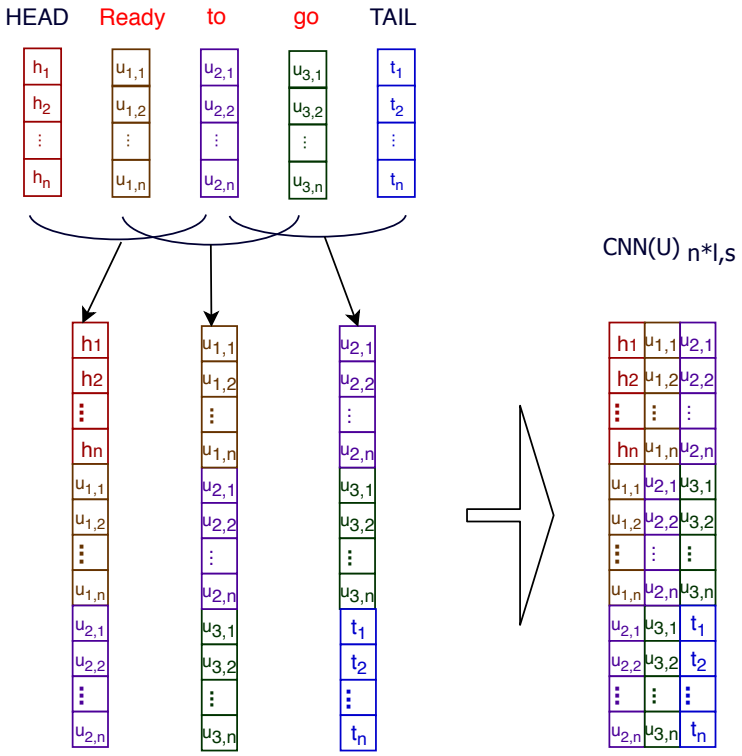

FIGURE 2.21: CNN's window movement

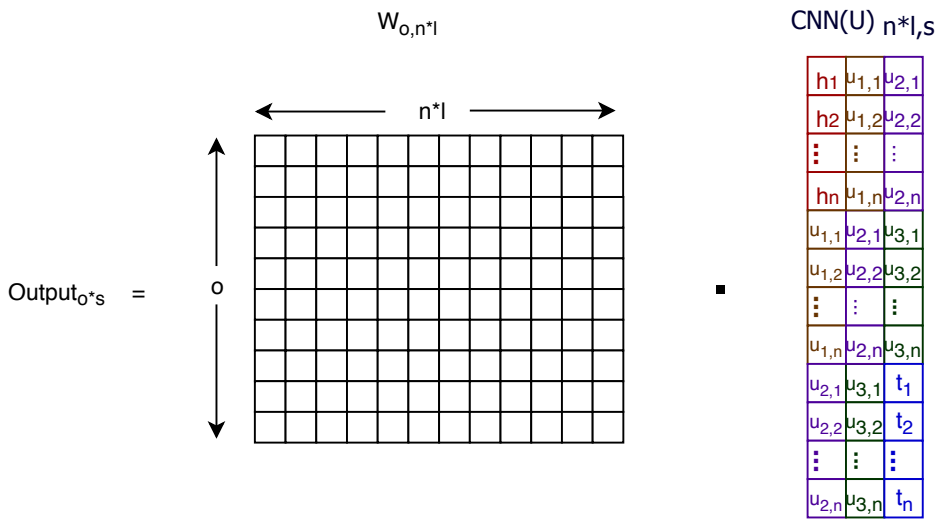

FIGURE 2.22: CNN's output

(Collobert and Weston, 2008) proposed a CNNs model for sequence labelling and phrase classification. As shown in figure 2.24, they first used a lookup table to represent words and features, and they then employed a convolution layer befor a max over time (pooling method). Finally, a softmax activation function gives the result of classification.

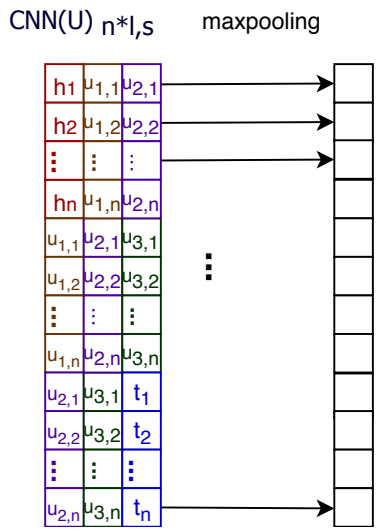

FIGURE 2.23: Max pooling with CNN output

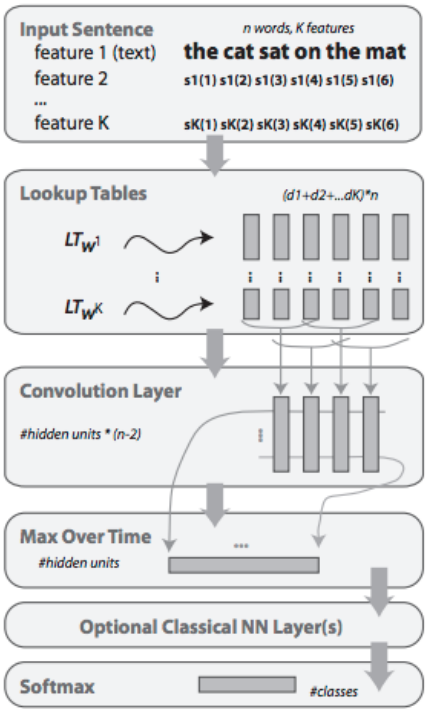

FIGURE 2.24: CNN from (Collobert and Weston, 2008)

Long short term memory (LSTM)

The perceptron and CNNs only consider the actual input. They do not keep previous information. That is why the long short term memory (LSTM) was invented later in (Hochreiter and Schmidhuber, 1997). The LSTM is a kind of recurrent neural network (RNN). The simplest RNN is Elman's network (Elman, 1990).

The figure 2.25 on the left shows this simple RNN.

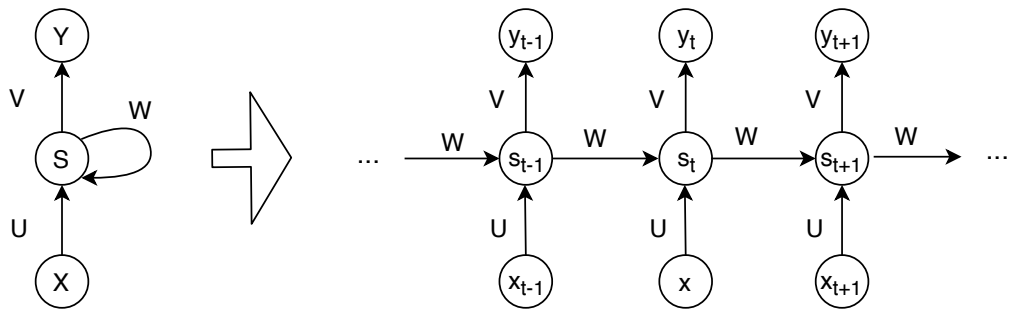

FIGURE 2.25: Elman's RNN

The RNN is just a simple hidden layer of a neural network plus a loop W. X is the input vector, Y is the output vector. U is the weights matrix for X to transform to S, which is the statement value. V is another weights matrix for statement S to transform to Y.

On the right of figure 2.25, we decompose the network structure in function of time: X is represented by $[...x_{t-1}, x_t, x_{t+1}, ...]$. Similarly S is $[...s_{t-1}, s_t, s_{t+1}, ...]$ and Y is $[...y_{t-1}, y_t, y_{t+1}, ...]$.

The recurrent loop W means, each statement s_t depends on the input x_t and the previous statement s_{t-1} at the moment t.

We suppose that the activation functions are f for X to S and g for S to Y. Since for all neural networks, the output of a hidden layer is $output = function(weights \cdot input)$, we can get for a RNN,

$$s_t = f(U \cdot x_t + W \cdot s_{t-1}) \tag{2.10}$$

and the output of a RNN will be

$$o_t = g(V \cdot s_t) \tag{2.11}$$

if we rewrite the equation 2.11 with the equation 2.10, we will get:

$$\begin{aligned} o_t &= g(V \cdot s_t) = g(V \cdot f(U \cdot x_t + W \cdot s_{t-1})) \\ &= g(V \cdot f(U \cdot x_t + W \cdot f(U \cdot x_{t-1} + W \cdot s_{t-2}))) \\ &= g(V \cdot f(U \cdot x_t + W \cdot f(U \cdot x_{t-1} + W \cdot f(U \cdot x_{t-2} + W \cdot s_{t-3})... \end{aligned} \tag{2.12}$$

The equation 2.12 means that the output of the time o_t depends on all previous input x_{t-1}, x_{t-2}.... That is long term memory.

Long short-term memory is a little different from the simple RNN. At the moment t, the statement s_t also depends on the previous output y_{t-1}, which is considered as short term memory, as shown in figure 2.26.

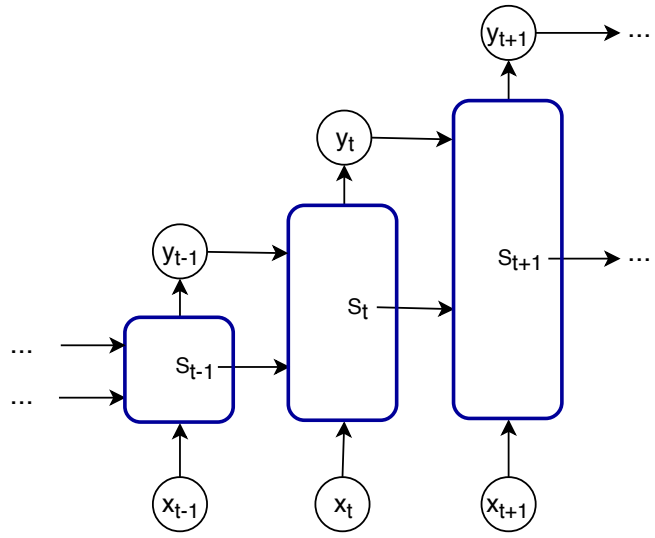

FIGURE 2.26: LSTM States

In this figure 2.26, blue squares replace the simple neuron of RNN. The cell statement c_{t-1} is considered as long term memory for the moment t.

We can see that at the moment t, we have three entries: the input x_t, long term statement s_{t-1} and short term output y_{t-1}. There are two outputs at this moment t: y_t as output and s_t as the input of the next statements (the next blue square).

LSTM also introduced the notion of "gate". A "gate" will control how much information passes through. It is just like a hidden layer, represented by a linear function

$$g(x) = \Phi(W \cdot X) + b \tag{2.13}$$

In formula 2.13, X is the input vector, W is the weights matrix of the gate and b is the bias. g(x) returns a real number between 0 and 1. When g(x)=0, that means the gate letting nothing pass through, the gate is completely close. When g(x)=1, that means the gate letting all pass through, the gate is completely open as if the gate does not exist. So in general, the gate is always between 0 and 1, that means between completely open and completely close. To use the gate, we multiply all elements of a vector (or matrix) by the gate value. The gate function often use the sigmoid function(Φ in equation 2.13).

LSTM defines three gates: The "input gate" decides how much informations of input x_t previous output y_{t-1} pass to cell state s_t. The "forget gate"

decides how much memory of previous output y_{t-1} and the input x_t we keep in the moment t. At last, the "output gate" controls how much of s_t to output to y_t.

All the three gates are sigmoid functions of the concatenation of input x_t vector and previous cell state y_{t-1} (expressed by [,] symbol) Only weights matrix (W) and bias (b) change.

The input gate:

$$i_t = \Phi(W_i \cdot [y_{t-1}, x_t] + b_i) \tag{2.14}$$

The forget gate:

$$f_t = \Phi(W_f \cdot [y_{t-1}, x_t] + b_f) \tag{2.15}$$

The output gate:

$$o_t = \Phi(W_o \cdot [y_{t-1}, x_t] + b_o) \tag{2.16}$$

Now we try to compute the output values: c_t and y_t. The output of the LSTM layer y_t only depends on the cell statement s_t and the output gate's value o_t.

$$y_t = o_t \cdot \Phi(s_t) \tag{2.17}$$

To compute the current cell statement s_t, we need an intermediate statement value $\tilde{s}$, which also depends on input x_t and previous output y_{t-1}. That is the memory of the moment t.

$$\tilde{s} = \Gamma(W_c \cdot [y_{t-1}, x_t] + b_c) \tag{2.18}$$

In the equation 2.18, Γ represents the $tanh$ function, which returns a real value from -1 to 1.

We can then compute the value of cell state s_t as follows:

$$s_t = f_t \cdot s_{t-1} + i_t \cdot \tilde{s} \tag{2.19}$$

With the control of the forget gate, the cell state s_t can keep information from long ago (before the moment t) or avoid unimportant information entering in the memory. Then the long term memory s_t will be the input of the next moment $t + 1$.

Figure 2.27 shows the calculates of gates and of outputs s_t and y_t.

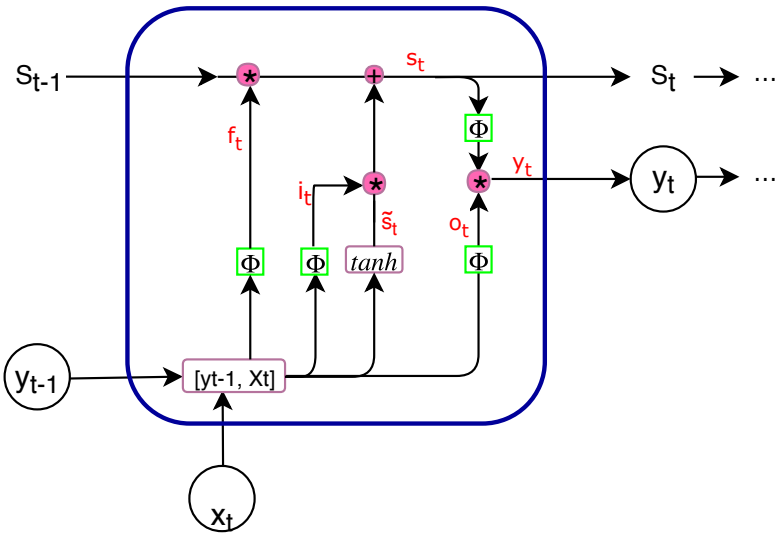

FIGURE 2.27: LSTM gates calculate inspired by colah's blog (http://colah.github.io)

Word representation with word2vec model

When we want to represent words in a form that our machine can read, we define a function to project words into a defined space.

The word2vec model presented in Mikolov et al., 2013 is first an additional representation obtained from a neural network training procedure. First, this NN model was in the objectif of training a language model, to predict the next word according to previous word(s).

We have a vocabulary of V words and each word has an number as its index in this vocabulary. With this index, we can represent this word by a V-dimensions vector with only one value of 1 (the index of the word) and other values of 0, this vector is called one-hot word representation.

This simple NN model takes the one-hot word representation of previous words as input layer, then the input layer is projected into one hidden layer of N dimensions with a linear activation function and then connects to a softmax output layer of V dimensions. The output is the next word according to the context (previous words), each dimension of V is one word in the vocabulary. In the hidden layer, the number of dimensions N is often much smaller than V, so the representation dimensions is fixed for all words and is much smaller than the vocabulary size V.

Figure 2.28 from Rong, 2014 shows a simple CBOW (Continuous Bag-of-Word) model with only one previous word as input. The input is a one-hot representation of size V. Only the previous word (for example x_k) has the value 1 and other dimensions have the value 0 for the input vector. With the

linear transformation using the vector W_{V*N}, we get a vector of N dimensions for the hidden layer. Then with another vector W'_{V*N}, we get into the output layer with softmax as the activation function. In the output layer, we also have a vector with only one value of 1 (for example, the word y_j from the vocabulary of size V) and other values are all 0.

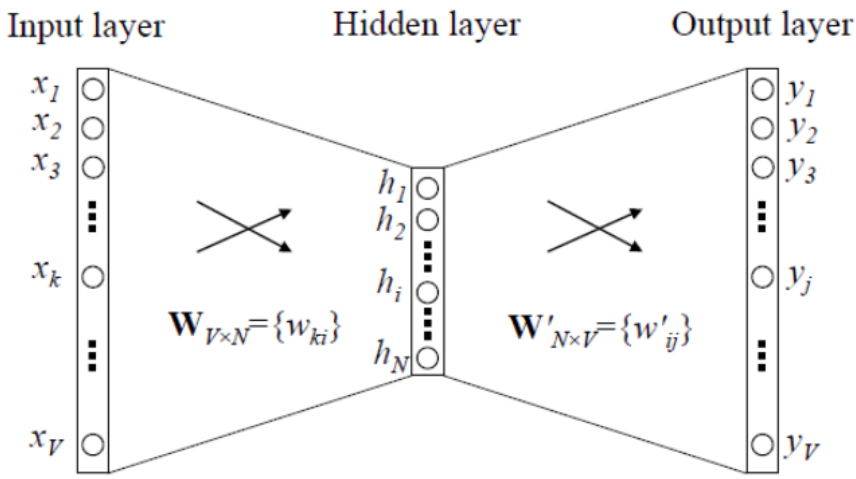

FIGURE 2.28: A simple CBOW model from (Rong, 2014)

Initially, the CBOW model was used to train a language model which predict the next word according to the context with the previous word. But in the end, with the trained model, the hidden layer of N dimensions (the vector $h_1, h_2, ..., h_i, ...h_N$) becomes a vector representation of the input word (x_k in the figure), and Mikolov, Yih, and Zweig, 2013 discovered that this representation encode efficiently word meaning. Words in similar context are considered as near to each other using cosinus distance. For example, "Paris", "Rome" and other capital cities often apear in similar context so these words are near to each other in the vector space. Since words are represented as vectors, we can do some addition and multiplication operations and calculate distance of two vectors. Mikolov, Yih, and Zweig, 2013 showed that "King - Man + Woman" results in a vector very close to "Queen".

The vector representation trained by this neural network for language model is called word2vec model. It is considered as an efficient representation for words in many nlp tasks since: for named entity recognition in Godin et al., 2015; Cherry and Guo, 2015; Sienčnik, 2015 and Toh, Chen, and Su, 2015, for text classifications in Lilleberg, Zhu, and Zhang, 2015, for syntax problems as in Ling et al., 2015, for semantic analysis in Yao et al., 2017 and etc...

There are also other similar word embeddings like GloVe (Pennington, Socher, and Manning, 2014), also used on named entity recognition in Toh, Chen, and Su, 2015.

2.8 Conclusion

In this section, named entity is defined from both linguistic and application point of views. Then major machine learning methods: Conditional Random Fields (CRFs) and Neural Networks (NN) are introduced. Some previous works about Named Entity Recognition (NER) and text normalization are examined.

Chapter 3

Raw data, basic normalization and Synthesio's dataset

In this chapter, we will first analyze a subset of Twitter, the most noisy social media. We will then try to analyze more finely a large unlabelled dataset from three Synthesio's classified domains: luxury, music and automotive industry.

We will talk about a basic normalization of an unannotated raw dataset by regular expressions. This normalization reduced the size of vocabulary in the dataset by using the same expression for similar tokens of time, quantity or volume etc. Meanwhile, the meaning of sentences and words sequences remain the same. The normalized large dataset is then used to create a word representation by vector (a word2vec model). The word meaning and semantic relations are preserved in the vector representation. This word2vec model (word representation by vector) is then used as input layer in the neural networks model in chapter 5.

In chapter 1, we explained that Synthesio needed to develop a proper named entity recognizer with a proper definition of named entity types. Furthermore, to evaluate the named entity recognizer with Synthesio's data, we first needed an annotated dataset. In this chapter, we will first explain the creation of the Synthesio reference dataset, including data extraction from different domains, definition of named entity types and manual annotation procedures.

Contents

2.1 History of Named Entity Recognition **13**

2.2 Named entity definition **15**

2.2.1 From the linguistic point of view 15

2.2.2 From a practical perspective 17

2.3 Domain and text genre **18**

2.4 Taxonomy of named entities **20**
2.5 Evaluation of Named Entity Recognition (NER) **21**
2.5.1 MUC evaluation (partial match evaluation) 22
2.5.2 CoNLL evaluation (exact match evaluation) 23
2.5.3 SemEval shared task evaluation 24
2.6 Machine learning based NER methods **26**
2.6.1 Supervised learning 26
2.6.2 Unsupervised learning and Semi-supervised learning 34
2.7 Text normalization for User Generated Content **35**
2.7.1 Definition and scope of lexical normalization 36
2.7.2 Definition and Taxonomy of Non-Standard Words . . 36
2.7.3 General procedure of lexical normalization 38
2.8 Conclusion **57**

3.1 Analysis of a Twitter subset data

Synthesio processes a huge amount of data everyday and subscribes to a Twitter service providing a randomized subset consisting of 10% of all given tweets in a defined time period. This service provided by Twitter makes it possible to trace world-wide trends. Useful information first needs to be filtered out.

In this section, we try to study a collection of Tweets and try to filter some forms of tokens. We then want to see the OOV words and their term count.

We collected 10% of random data from one month of tweets from the first two weeks of September 2015. We then filtered them with language identifier in Synthesio and kept only tweets with more than 90% probability of being written in English. This data collection contains all tokens (all letters in lowercase) and term counts for each token. There are 375,718 different tokens in total after removing:

- Tokens containing non-ASCII characters (numbers and punctuation symbols are kept)
- Extra apostrophes due to Synthesio tokenizer (tokens beginning with apostrophes like 'the)
- html code (like > for <)

- Tokens only containing punctuations (emoticons included)

We then extracted out-of-vocabulary words using the Aspell English dictionary (British English and American English) and we studied the number of OOV words with different term count.

Table 3.1 shows the statistics of this data collection. From this table, we can see that the term count and the percentage of tokens which are in the interval of term counts follows Zipf's law (Powers, 1998). That means, in this data collection, 37% tokens appear less than 10 times and 70% tokens appear less than 20 times.

Term count	percentage	number of OOV words
-10	37%	
-20	70%	
1k+	0.03%	about 2500
5k+	0.01%	about 500
10k+	0.005%	about 200
total		85%

TABLE 3.1: Term count and number of OOV words

We then studied the OOVs with more than 5000 term counts, there are almost 500.

They are:

- Named Entity (186)
  - Person: 79
  - Location: 48
  - Other: 14
  - Brand/company/product/Organization: 45
- Internet language words (164)
  - Invented words like "lol": 62
  - Extremely long words (such as "looooooove"): 5
  - Extremely short words (with only one letter, such as "n" for "and", "u" for "you"): 20
  - New words like "wifi": 10
  - Attached words: 7 (tokenizer error)
  - Without apostrophe (as "Im"): 14

  - Oral words: 10 (like "wanna")
  - Interjection: 16
  - Twitter language like "retweet": 14
  - Emoji: 1
  - Prononciation like "dat" for "that": 3
- Abbreviation: 6
- Standard words like $10: 16
- Unknown words (70)
  - Other natural languages (5)
  - Unknown words with only one letter (16)
  - Other unknown words (49)

As we can see, standard words of quantities like "$10" or "10$" are considered as out-of-vocabulary words (OOV). Since we want to automatically analyze texts, the token of the real quantity may not be so important for parsing the whole sentence. That is why we try to rewrite some quantity expressions with defined rules.

3.2 Basic normalization on Synthesio's data

As mentioned before, Synthesio processes data from all kinds of ressources: main stream articles, official web sites and User-Generated Content (UGC) as forums, tweets etc. Texts are labelled by domains and labels are then used as indexes. Synthesio's clients choose labels which interest them in order to create a dashboard to visualise data analysis results. We want to collect a subset of labelled Synthesio's data to simulate real-world data. We chose three distinct domains where Synthesio has most clients: luxury, music and automotive industry, and we tried to create a special vocabulary.

This data collection contains:

- 312,418,051 sentences
- 152, 260, 070 unique sentences
- 3,009026,341 tokens (in average 19.76 tokens per sentences)
- 11,562,105 unique tokens

The Aspell English dictionary (standard American and British English) contains 100,904 different words. This data collection has 115 times more tokens.

This data collection is extremely raw. Even if we only want a token form and the term count, we have to filter a lot of tokens.

3.2.1 Use rewrite rules for tweet lexical normalization

As showed in the first section, we can find other classes of quantity (with different unit), volume and currency expressions and we can regroup similar tokens like:

- Quantity: length (m, km), number with "k", volume (ml, cl), weight (kg), currency
- Date, month, hour
- Expression of equation, age
- Game score like "1-0"
- Telephone number

We used regular expression to regroup some of these expressions into sub-classes and changed them into normalized form. For example, we regrouped "1cm", "2m", "3inches" into the length class and we changed them into "__LENGTH__". We are only interested in this token which expresses a length. The exact length value of the expression is not so important, that is why we changed them into one unique token for all length expressions.

Similarly, we processed listed numbers for volume (10ml, 1L) , telephone numbers (from eight to ten consecutive numbers or separated by '-') , age expressions like "6-year-old" and "7 years" and etc...

These changes allowed us to limit the size of the vocabulary to train a word2vec model using Synthesio text.

3.2.2 Uppercase/lowercase normalization

In the previous study, we changed all tokens into lowercase form. This is not useful to distinguish named entities. Lots of tokens appear in the text with different forms in uppercase/lowercase.

We wanted to limit our vocabulary size and tried to normalize tokens with uppercase/lowercase and we wanted to ignore the uppercase letter in the beginning of a sentence.

For example, the Google word2vec model did not normalise different form of tokens. So there are different vectorial representation for words "Nissan", "nissan" and "NISSAN". If the word's vector represents the meaning of the word, it should not change when the word changes its form: first letter in uppercase, all letters in uppercase, etc. As we know, the character sequence "nissan" should be the automotive brand, there is not any ambiguity. So we want to normalize this kind of problems, to keep only the form "Nissan" for "NISSAN" and "nissan". There will be only one form "Nissan" replacing other forms "NISSAN" and "nissan", even eventually "nIssan", "niSsan" or "niSSan", these will be considered as errors for "Nissan" and all would be changed into one unique form "Nissan".

It is a delicate operation to ignore uppercase letters. To study the possibility to normalize tokens with this idea, we first tried to study different category of words and get statistics of all forms.

We extracted data from 3 special domains of Synthesio data: luxury, music and automotive industry. Data is extracted from Twitter and other sources (websites, forums, etc).

This time we kept all forms of tokens (uppercase and lowercase) to see if there was a huge impact of this normalization. We then filtered the data and only kept tokens when their term count was more than 5 (as did word2vec model), then we also filtered all classes like quantity, currency mentioned in the previous section.

Common nouns and proper nouns

We chose some words from different part-of-speech categories, to see their possible forms (letters in uppercase or lowercase) and for each form, the percentage in the whole large data collection. In order to limit vocabulary size, we decided to change some possible forms of tokens into their most frequent forms.

From tables 3.2to 3.8, we can see that "crystal" and "jack" are common nouns and proper nouns, "share" is a verb, "however" is an adverb (often begins a sentence), "Nissan" and "McDonald" are proper nouns and "4Runner" is a proper noun which mixed number and letters.

We can also see that even very rare, special forms like "JaCk", "SHare" and "MCDonald" exist in the large data collection. Two first letters in uppercase

like "SHare" may be due to people inadvertently keeping a "shift" key longer than necessary. "JaCk" is a classical style: there is one uppercase letter every two letters.

In either case, we can replace these special forms into their form the most frequent form.

crystal	Crystal	CRYSTAL
58.8%	39.5%	1.6%

TABLE 3.2: Forms and their percentages for "crystal"

jack	Jack	JACK	JAck	JaCk	jACK
59%	39.8%	1.06%	0.01%	5.11e-05	4.3e-05 (16 occurrences)

TABLE 3.3: Forms and their percentages for "jack"

Share	share	SHARE	SHare
64.1%	35.7%	0.21%	5.4e-06

TABLE 3.4: Forms and their percentages for "share"

However	however	HOWEVER	HOwever
53.7%	45.3%	0.88%	9.5e-05

TABLE 3.5: Forms and their percentages for "however"

Nissan	nissan	NISSAN	NIssan
89%	7.8%	2.6%	0.19%

TABLE 3.6: Forms and their percentages for "nissan"

McDonald	Mcdonald	mcdonald	MCDONALD	McDONALD	MCDonald	McDOnald
90.3%	4.4%	2.87%	1.144%	0.427%	0.414%	0.4136%

TABLE 3.7: Forms and their percentages for "McDonald"

4Runner	4runner	4RUNNER	4RuNnEr	4RUnner
57%	41.96%	0.79%	0.01%	9.4e-05

TABLE 3.8: Forms and their percentages for "4Runner"

However, for the second most frequent form, it is difficult to decide if we keep this form as the standard one or replace it by its most frequent form.

Table 3.9 showed examples of tokens and its most frequent two forms. We can see that for the proper noun "McDonald" (either familly name or part of "McDonald's" for the fastfood brand), the second most frequent form appears only 4.4% of the time, so we can change all letter sequences "mcdonald" . Similarly, for "Nissan", we can say that the first letter written with an uppercase letter and the next ones on lowercase is its standard form, and change (normalize) all other forms into "Nissan".

For "share", sometimes it could be the first word of a sentence or simply the text for a button in a website (Synthesio's crawler kept words like this). We should probably keep the two forms. The token "Yeah" often begins a sentence, or itself forms a sentence. It should begin with Y in uppercase and other letters in lowercase. But in user-generated content (UGC), people do not always respect rules that words should begin with uppercase : writing "yeah" is more convenient.

Token	Form	Percentage	Form	Percentage
crystal	**crystal**	58.8%	**Crystal**	39.5%
jack	**jack**	59%	**Jack**	39.8%
share	**Share**	64.1%	share	35.7%
however	**However**	53.7%	**however**	45.3%
nissan	**Nissan**	89%	nissan	7.8%
mcdonald	**McDonald**	90.3%	Mcdonald	4.4%
4runner	**4Runner**	57%	**4runner**	41.96%
yeah	**Yeah**	60.07%	yeah	39.1%

TABLE 3.9: Examples of the most frequent two forms

The goal of this uppercase/lowercase normalization is to limit the number of tokens. We first wanted to replace all case variations of a given word by their most frequent form present in more than 50% of the total number of occurrences. However, this rule would have caused some person names (words with the first letter in uppercase) to be replaced by all lowercase variants, rendering them undistinguishable from common nouns as in "Crystal" to "crystal". We decided to keep a variable number of the most used forms of individual words instead, representing 60% or more of all possible case variations.

We chose 60% as a test, to see if the generated word-embedding model corresponds to a more representative word meanings than the Google word2vec model.

That means, we kept only forms in bold in table 3.9 and other forms were changed into their most frequent form. As a consequence, in the data collection, for example, there would be only "Nissan", and the form nissan would be excluded.

After that normalization on uppercase/lowercase, we can still see OOV words in this data collection.

For frequent OOV words, we get some examples:

- Tokeniser errors (tokens begin with apostrophe):
  - 'ipt: 165948
  - 'tapatalk: 137830
  - 'The: 36234
- Noise/unanalysable:
  - T: 527860
  - b: 568487
- Non-standard words (could be corrected)
  - btw: 119526 (by the way)
  - il: 47936
  - tmrw: 4090 (tomorrow)

For less frequent OOV words, we get examples like:

- Tokenizer errors (words with punctuations):
  - " With": 5
  - Car'Opening: 1
  - a*USED: 1
- Non-standard words (could be corrected)
  - worryy: 5 (worry)
  - zithout: 5 (without)
  - 2murrow: 1 (tomorrow)
  - CROSSIANTS: 1 (croissants)
  - Catogary: 1 (Category)

  - Causually: (Casually)

- Non-standard words (named entities)
  - workshift: 5 (Japanese global crowdsourcing service)
  - wingsonic: 5 (a hardware brand name)

In table 3.10, we tried different filters and at each step, we showed the percentage of the remaining tokens, and the percentage of in-vocabulary (IV) words (using Aspell dictionary, standard American and British English).

Initially, the number of extracted (different) tokens was 1,1562,105 (as in the first row). The following rows in the table show applied filters and the number of remaining tokens. "Type" is an automatic label added by Synthesio's tokenizer for each token. "Type=None" means the token is a regular word only containing letters. Other possible type values are "number", "punctuation", "url", "email", "hashtag" (for tweets), etc. These type values are recognized by regular expressions defined in Synthesio's tokenizer. "hasAlpha" shows the number of remaining tokens if we keep only tokens with at least one alphabetical letter. "lowerform" changes all tokens into their lowercase form (all letters in lowercase). "removeApos" removes apostrophes as the first character of a token with at least two letters after the apostrophe (to keep tokens like "'s" and "'n"). "removeNonEnglish" uses a module "langID" which is a 5-grams model trained on English words. In this step, we only keep words which have more than 80% of probability of being in English. From this step on, we have reduced the number of words (from about 11 million to about 600,000) enough to become able to check them one by one if they are In-Vocabulary (IV) words in a reasonably limited time. The following four lines concern the number of occurrences. We can see that the step which filters the most tokens is "removeNonEnglish", processed by the language identifier of Synthesio.

We want to choose a threshold of the vocabulary size. Sometimes even a token is not a standard word, for example "tmrw" for "tomorrow", if there is a lot of term counts for "tmrw", we can consider that since people often write like that, "tmrw" is a comprehensible word, and this should be considered as a standard word (a new In-Vocabulary word).

Table 3.11 shows the final chosen filter process.

Since we chose 5 as window size for context (the two previous words and the two following words), we studied the most frequent contexts in the automotive industry domain in this data collection.

Step	# words	percentage	words in Aspell
Initial	11,562,105	100%	
Type=None	11,522,036	99.7%	
hasAlpha	9,801,682	86.8%	
lowerform	8,242,706	71.3%	
removeApos	8,169,132	70.7%	
removeNonEnglish	644,725	5.6%	0.7%
Termcount >= 5	100,428	0.87%	4%
Termcount >= 100	13,166	0.11%	23%
Termcount >= 200	8,064	0.07%	33%
Termcount >= 300	6,186	0.05%	39%

TABLE 3.10: Vocabulary filters

Step	# words	percentage
Initial	11,562,105	100%
Type=None	11,522,036	99.7%
hasAlpha	9,801,682	86.8%
beginWithAlnum	9,704,710	83.9%
length<20	9,416,062	81.4%
length<15	8,810,122	76.2%
langID, prob>0.8	1,213,098	10.5%

TABLE 3.11: Final vocabulary filters

Table 3.12 shows the most frequent 5-grams in this data collection. When the column "previous words" is empty, that means the "current word" is the first word in a sentence.

We can see that the top-3 most frequent 5-gram is like "quote originally posted ...", "quote originally posted by ..." and "originally posted by ...".

Then we can find sequences like "I don't", "i have a" which is frequent in our corpus.

With 232,590 term count, the sequence "'m 'm 'arket details id com quoord" is clearly due to raw data and errors of Synthesio tokenizer.

After these, we can see more expressions from speaking English like "I'm not ...", "it's a ...", "I can't ...", "I didn't ...", "I don't know ...", "I have the ...", "thanks for the ..." and "I don't think ...".

The last two lines are also present in the raw data.

If we ignore the current word and only consider the most frequent context, we obtain table 3.13 which shows the most frequent context in the automotive industry domain (same data of previous table). Single words in this table mean that the word was the only word in the sentence because Synthesio's tokenizer first separates text into sentences and then tokenizes them

Term count	previous words	current word	Next words
5,641,073		quote	originally posted
5,641,058	quote	originally	posted by
1,828,859		originally	posted by
650,758		i	do n't
428,743		i	have a
232,590	'm arket	details	id com
232,590	arket details	id	com quoord
232,590	'm 'm	arket	details id
216,939		i	'm not
158,093		it	's a
150,643		i	ca n't
146,403		i	did n't
142,136	i	do	n't know
129,117		i	have the
123,238		if	you have
122,322		thanks	for the
119,494	i	do	n't think
116,295	app_android_url	'm 'm	arket details
116,295	'm obiquo	smartbanner	tapatalk2 png
116,295	tapatalk2	var	app_iphone_id 3078807

TABLE 3.12: Most frequent 5-grams in automotive industry domain

separately.

If we represent the current word by "W", we can express the most frequent context. We can see that the top-3 context are coming from a website pattern: "W originally posted ...", "quote W posted by ...", "W posted by ...".

The next four lines showed some valid context like "... in the W", "... on the W", "... of the W" and "... W don't".

There are then still three lines which shows the raw data: "... posted by W view post ...", "... kib viewed W times attachment", "... jpg kb W views" and later "... kib viewed W times". They are clearly associated with websites and attached files.

After these, we can also see some text sequences like "... to the W", "... for the W", and "W have a".

Then another sequence associated with websites "attached images W jpg kb".

The last 5 lines showed other common context: "W is a ...", "W it is ...", "W's a ...", "W" (the current word only forms a sentence like "Yeah") and "... with the W".

# count	previous words	next words
5,650,220		originally posted
5,641,066	quote	posted by
1,829,812		posted by
1,044,657	in the	
971,744	on the	
940,075	of the	
875,005		do n't
822,903	posted by	view post
662,645	kib viewed	times attachment
647,622	jpg kb	views
571,335	to the	
556,493	kib viewed	times
551,768	for the	
530,588		have a
480,067	attached images	jpg kb
471,680		is a
444,985		it is
351,782		's a
346,314		
345,179	with the	

TABLE 3.13: The most frequent context in automotive industry domain

As shown with these two tables, the top-20 most frequent 5-grams or only context contains lots of noise, these non-canonical texts are easily extended everywhere (by retweet or cited by responses on forums) and it is difficult to define a threshold (of term count) to exclude raw data.

3.2.3 Construction of Synthesio word2vec model

With the dataset extracted from Synthesio, elementary uppercase and lowercase normalization and after the application of rewrite rules as detailed in previous sections, Synthesio created a word2vec model which contains a vocabulary composed of about 2 million tokens (2,086,985 exactly).

This word2vec model is created with exactly the same configuration as the Google word2vec model from (Mikolov et al., 2013).

As mentioned in chapter 2, the pretrained Google word2vec model is trained on about 3 billion words from Google news. The model contains 300-dimension vectors for 3 million words and phrases.

Each word is represented as a vector in the space of the set of the entire text of 3 billion tokens. Here we chose a model with 300 dimensions, that

means, each vector of 300 dimensions represents one token in this space, and the distance between two tokens is calculated by the cosinus distance of the two vectors corresponding to the two tokens.

This word2vec model can also find the nearest words according to their cosinus distance. For example, table 3.14 shows nearest words for the phrase "Chinese river" from (Mikolov et al., 2013) [1].

Phrase	Cosinus distance
Yangtze_River	0.667376
Yangtze	0.644091
Qiantang_River	0.632979
Yangtze_tributary	0.623527
Xiangjiang_River	0.615482
Huangpu_River	0.604726
Hanjiang_River	0.598110
Yangtze_river	0.597621
Hongze_Lake	0.594108
Yangtse	0.593442

TABLE 3.14: Nearest words of "Chinese river"

From table 3.14, "Yangtze_River", "Yangtze", "Yangtze_river" and "Yangtse" are different ways for the same river, and "Yangtze_tributary" is for a tributary of the main river "Yangtze".

3.3 Use Google Request to find correct form for non-standard words

Google provides 100 free requests per days, and then charges $5 for 1000 requests. We just did some tests with their free services.

This method came from (Liu et al., 2011a). They found 3802 non-standard words and projected them into 2000 standard words.

If we compare this Non-Standard Words (NSW) set with the vocabulary of Synthesio, there are 2954 non-standard words in Synthesio's dataset. These NSW are generated from unannotated tweets with context and it is possible that one NSW can correspond to different standard forms. the same form of a NSW is rewritten with more than one correct form. For example, according to context, the token "2" can be rewritten as "to", "too" or "two".

[1] https://code.google.com/archive/p/word2vec/

The domains that we studied are still luxury, music and automotive industry. These are distinct domains and covered most important Synthesio clients.

First we got out-of-vocabulary words and changed them into lowercase. Secondly, we tried to get contexts for these OOV from these three domains. The context is considered as two words before and two words after (punctuations included). We then sorted the top-10 contexts (the 10 most frequent) for one OOV as 10 requests. And then we sent these 10 requests to Google, and we obtained the 10 first page descriptions that Google returned. Then we kept text between and in the HTML source code of the page (that are often corrections proposed by Google) without tokenization nor change of uppercase/lowercase. Lastly, we used edit distance to get the nearest forms of the OOV.

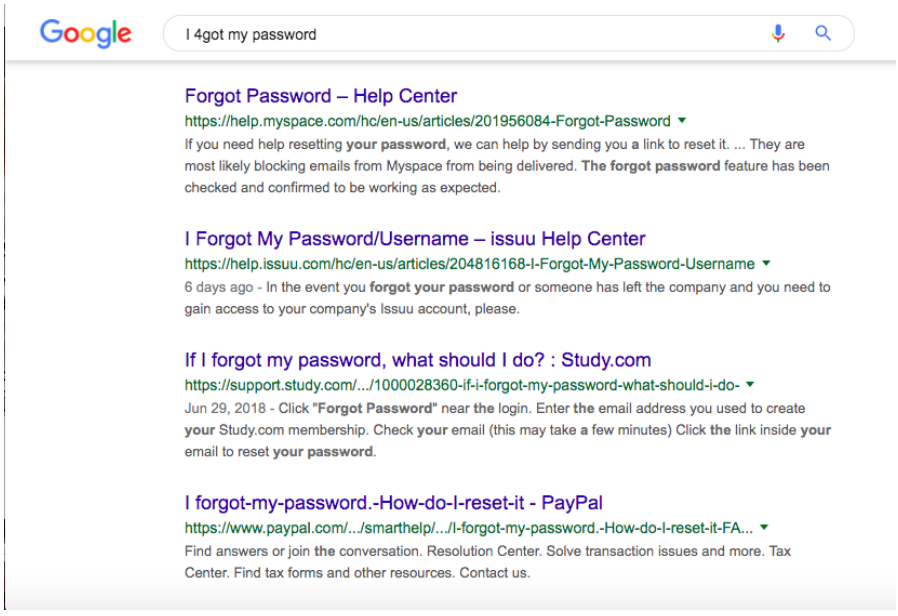

FIGURE 3.1: Google page with the request "I 4got my password"

Figure 3.1 shows an example of the Google page results with the request "I 4got my password". The Non-Standard Word (NSW) is "4got" (for "forgot"). From this Google page, there are proposed websites with urls and page descriptions under the urls. We can see that in page descriptions, words that matched with the request are in bold (as "The forgot password" in the first proposed website), and the correct form of the NSW "4got" is in bold in the page description.

We chose a OOV word "wfolarry" as an example to analyze. "wfolarry" has 244 occurrences in our data collection.

Table 3.15 shows its 10 most frequent contexts and the term count.

wfolarry and context	term count
posted by wfolarry view post	123
me wfolarry has done	6
quote wfolarry said leave	5
wfolarry	5
thanks wfolarry for finishing	4
don wfolarry bruce	4
posted by wfolarry you do	3
posted by wfolarry are you	3
scott or wfolarry or bean	3
's 'cuz wfolarry has worked	3

TABLE 3.15: Contexts and term count for "wfolarry"

We can see that "wfolarry" is probably an username since the most frequent context is "posted by wfolarry ...". However, we still want to see correction by Google. When we launch requests on `google.com`, we get HTML source code with pages explications like:

- ... View on Time on Tumblr: Hover over the upper-right corner of a posttoseethe
time-and the date if ... These blogs post the time and date every 15 minutes to
help put your Dashboard posts in the ... unwrapping postedthis.
- Below, you'II find out how to rearrange queued posts. viewarepostedon a specific <br> date.
- S$amp;S flywheels, wfolarry110 heads, Tman 662-2 cams, Dan Thayer 3 stage oil <br> pump, Gaterman 2023 lifters, Smith Brothers ... 1,089 views. 3 ...
- ... Blogging on tumblr is like a walk in a park. Everaything is made easy for its users, <br> from viewing postsof the blogs you follow to posting new stuff ;...
- ...

FIGURE 3.2: Page descriptions for "posted by wfolarry view post"

When we calculate the edit distance between each proposition and the original token "wfolarry", we can sort tokens obtained in the previous step. From this table, we can also see that there is a lot of noise in the Synthesio data. "WFO larry" could just be a user name but the frequency of the phrase "posted by WFOlarry" in the Synthesio data raised the number of occurrence but it should not be a word that appears regularly.

We calculated the edit distance between each proposition and the original token "wfolarry" and we kept tokens until the length of the original token -1

Context	Words between and
posted by wfolarry view post	post, posted, posts, posting, Posted
	View, view, views
	see
	viewing posts, View Post
	wfolarry, WFOLarry
me wfolarry has done	me, Me
	has, have
	Done
	has taken me, has taken, Has Taken Me
scott or wfolarry or bean	bean, Bean
	Scott, scott
	Scott Bean, Scott Bean&39;s
	trombean
wfolarry	WFO larry, wfo Larry
	wfolarry, WFOLarry

TABLE 3.16: Context of "wfolarry" and matched words

Edit distance	candidates
2	wfo Larry
4	WFOLarry
5	WFO Larry
6	will, Donald, working
7	Thank, trombean, scott, have, You ...

TABLE 3.17: Candidates and edit distance with original token

(that is 7 in the table). But we can see that possible candidates can have until 5 as edit distance.

Table 3.18 shows other results with different edit distance values.

We can see that until edit distance 5, the propositions of candidates are useful. More than 6, the propositions are becoming unrelated words.

Token	Term count	Edit distance	Candidates
thankyouverymuch	275	3	thank you very much
thses	289	2	these, thesis
toned-down	297	1	toned down
whulandary	35	1	Whulandary
thumpin	215	1	Thumpin, thumping
whippersnappers	280	1	Whippersnappers, whipper snappers
thebe	9	1	Thebe
therad	58	2	there, the Rad, The rad, thread
speights	32	1	Speights

TABLE 3.18: Other edit distance with original token

Using the Google search engine to find the correct form of a NSW word is

possible. But since late 2017, Google changed politics of their request/response API. Google provides 100 free requests per day and charges $5 for 1000 requests. So we did not proceed along this path.

3.4 Creation of Synthesio Reference corpus

In this section, we will explain how we created an annotated dataset from Synthesio data.

We will first describe how texts have been extracted from different domains and different ressources in available texts in Synthesio.

We will then describe how named entity types have been defined in section 3.4.2 . Finally, section 3.4.3 will present the text annotation procedure in Synthesio.

3.4.1 Data extraction

In Synthesio, a "mention" is a complete text which could be a tweet, an article in a journal or a post from a discussion forum etc as illustrated in figure 1.4. It is different from the definition of "mention" for named entity like mentioned earlier in this thesis.

A mention in Synthesio data contains fields like an unique identification, an url, a title, a content, a timestamp and eventually an author (a twitter account for example) or url for images (for instagram), representing metadata. In Synthesio, the field "title" is for journal articles or forum posts and is empty for "mentions" from Twitter. Since we only analyse data in form of text, only the fields "title" and "content" are textual data. However, we cannot connect the title text and the content text as one paragraph because sometimes the title is not a complete sentence and is not syntactically correct when directly pasted before the content. That is the reason why we will only extract the field "content" as text in the dataset. We kept the mention's unique identifier to avoid duplicated data.

Synthesio provides services to its clients by creating a "dashboard" to display diagrams of data analysis.

Figure 3.3 is a demonstration of a dashboard created by a client in the video game industry. This dashboard contains three widgets for gender analysis, age distribution and a word cloud.

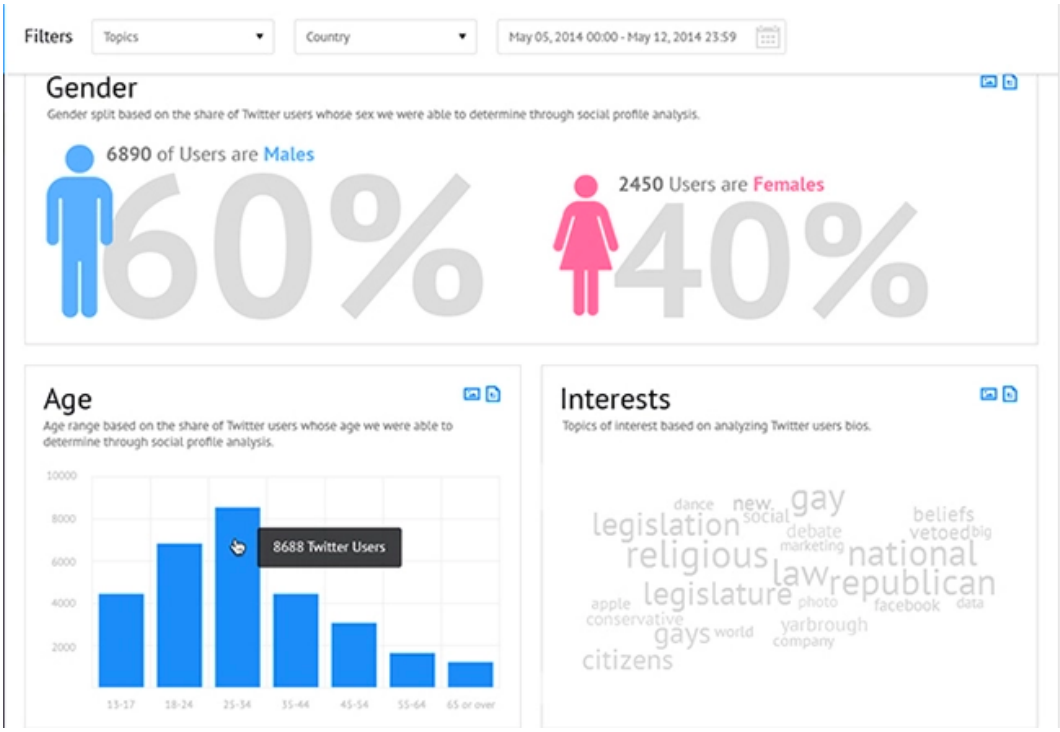

FIGURE 3.3: Examples of a dashboard with three widgets

When creating the dashboard, Synthesio's clients use one or several keywords (along with ressources and time periods etc.) and Synthesio's search engine returns related mentions.

Among these mentions, there are also mentions for other brands (for example their competitors) classified manually for the client. For example, when a dashboard with key word "Dunkin' Donuts" is created, mentions returned also include potential competitors of "Dunkin' Donuts". Mentions which contain other brands of fastfood (the same domain) like "McDonald's" and "Starbucks" can also be present in the dashboard.

From now on, we will only talk about texts, which are extracted from the field "content" of the "mentions" in Synthesio. We will not continue to use this definition of "mentions" for the rest of this thesis.

Synthesio processes texts of over 25 different languages and has a language identifier based on a five-grams language model. To identify the English language (both American English and British English) for a text, the probability of the language model is often fixed to more than 0.9. This score means that there is 90% of probability that the text (represented as a word sequence) is in English.

This threshold is supposed to filter texts from other languages. Since the language model is based on a context with five consecutive tokens, one text with two or three words in other languages may also be extracted as a text

written in English language.

We first fixed the time period to one month. That corresponds to the maximum data quantity (about 4GB) that we can handle with the language identifier using a laptop. We want to process only English text so we need precise data. That is why we tested the threshold with 0.9, 0.95 and 0.98. We found that even with a probability higher than 0.98, asian characters can always be present in the extracted texts. With the increasing threshold from 0.9 to 0.98, the quantity of extracted data reduced rapidly. Therefore, we kept the threshold at 0.9.

As mentioned in chapter 1, Synthesio's clients cover domains from automotive industry, cosmetic, electronic devices to music streaming services. When we create the reference annotated dataset, we want to get a corpus as close as possible to the data analysed by Synthesio in order to get as reliable results as possible.We chose four major domains: "automotive industry", "fast food", "online music streaming service" and "toy for children", which cover most clients at the moment and these four domains are independant from each other. We wanted to avoid related domains that may have similar vocabulary. For example, similar domains as "food" (such as "Danon") and "fast food" (such as Starbucks) may have similar vocabulary. In addition, we chose two clients, "Land Rover" and "Nissan" from the automotive industry to test if our model performs similarly in the same domain. Ideally, we should get similar performance for different texts in the same domain. We also chose one important client from each of the three other domains: Dunking Donuts from "fast food", Deezer from "online music streaming service" and Mattel from "toy for children".

In order to vary data ressources, for each client, we extracted 50 texts from long texts: forums and official web sites and 50 texts from short texts: Twitter, Facebook and Instagram. This data forms the Syntesio dataset with five clients from four domains and for each client, 50 long texts and 50 short texts.

Synthesio already has a tokenizer based on a CRFs model associated with an exception list. This tokenizer is configurated to separate tokens like "I'm" and "it's", so there are tokens beginning with apostrophe like "'m" and "'s". That is why "McDonald's" is tokenized as tokens "McDonald" and "'s". Further more, negative expressions like "don't" and "can't" are tokenized by separating the verb and the negative expression as "do" and "n't" then "ca" and "n't". The exception list contains words which were given a wrong tokenization by the tokenizer. In previous examples, we can see that the apostrophe

"'" is not considered as a word separator, it is attached with the letter that follows, as in "'s", "'m", etc. However, in a text like:

He said :" We are going to watch 'The Avengers'[a] tonight."

[a] An american film of 2012

FIGURE 3.4: An example of an attached apostrophe

the tokenizer returns a list of tokens with "'The" as one token. The apostrophe is thus attached to the word "The". Therefore, the exception list contain these cases for the tokenizer as a post-treatment to correct this sort of error.

Table 3.19 shows statistics of tokenized texts. Texts are first separated by sentences (as sequences in CRFs model), that is why the text of 50 tweets can have more than 50 sentences.

long text	Deezer	Dunkin	Land	Mattel	Nissan
sentences	146	208	183	174	123
tokens	2314	3116	3836	2958	2166
short text	**Deezer**	**Dunkin**	**Land**	**Mattel**	**Nissan**
sentences	52	50	59	57	74
tokens	854	827	1048	1123	1127

TABLE 3.19: Synthesio Reference Corpus.

3.4.2 Entity type definition

As mentioned in chapter 2, the definition of named entities first appeared in the Sixth Message Understanding Conference (MUC-6) on 1995 (Sundheim, 1995). The named entities to be extracted were: `person`, `organization`, `location`, `date`, `time stamp`, `currency` or `percentage figure`. The next MUC-7 task (Chinchor and Robinson, 1998) regrouped three types of information to be extracted: named entities, temporal expressions and number expressions and named entities included `organization`, `person` and `location`.

Later in 2003, the Conll conference defined four types of entities for its workshops on NER. These types were: `person`, `location`, `organisation` and `MISC` (others). Later, authors of (Ritter et al., 2011) defined ten types of entities: `Company`, `Person`, `Geo-location`, `Facility`, `Product`, `Tvshow`, `Sportsteam`, `Movie`, `Band` (`Music artist`), `Other`, because they were

the top ten types of entities with good cover in Freebase [2], in order to compare with the results by machine learning.

Synthesio's purpose is to extract texts of interest to their clients. These texts could be texts discussing the client's product and these clients could be famous persons, music bands, companies in some location or sports teams, facilities etc... Synthesio took advantage of the type definition in (Ritter et al., 2011) and regrouped `Tvshow`, `Movie`, `Band` (`Music artist`) into `Media`, and added another important type for Synthesio: `Job title`. The table 3.20 shows Synthesio's named entity definition.

Company	Company name
Person	Person name
Geo-loc	Location, country or city name
Facility	Organization name
Product	Product name
Media	Journal, music artist
Sportsteam	Sports team name
Job-title	Job names like director, CEO
Other	Holidays, events, etc

TABLE 3.20: Synthesio Named Entity Definition

3.4.3 Data annotation

As for annotation, several human annotators (from freelancers) worked with different text domains and then all annotations were checked (some named entities were missing) and corrected by a developer who knows well the NER problem and who has a tendancy of annotating most named entities possible with the goal to test strictly a machine learning model.

When we wanted to develop a method for named entity recognition (NER) for Synthesio, we first needed a corpus built with Synthesio data to represent the actual data and also annotate typed named entities in this corpus for evaluation. That is why we created a Synthesio reference named entity corpus. We first studied the types of named entities that Synthesio needed to extract, then we extracted data from multi-domain and multi-sources texts. However, we were limited in mentions in English. After that, we tried to annotate these corpus with types named entities to finally build this reference corpus for Synthesio.

Figure 3.5 and 3.4.3 show one example of a long text of Deezer and another example of short text of Mattel.

[2] https://en.wikipedia.org/wiki/Freebase

Spotify is expanding once again and is now available in Italy, Poland and Portugal. The music streaming service has been available in several European countries, but is still far behind Deezer in terms of reach. That's part of the strategy, Deezer is focusing on casting as wide a net as possible, Spotify .. (read more)

FIGURE 3.5: An example of long text

DISNEY ATLANTIS THE LOST EMPOIRE MILO THATCH MATTEL 00 MOC http://t.co/bkY0Vi7Fnv $4.99 @SILVERJACK810

FIGURE 3.6: An example of short text

In the Synthesio natural language process (NLP) procedure, according to language detected by the language identifier, the corresponding tokenizer is selected to tokenize the text first into sentences and then into a token list. And then for sentiment analysis, the stopwords are filtered from this token list and the sentiment of the text is calculated from a list of positive words and a list of negative words of the language detected.

Since the context is important to recognize a named entity, our NER module will work with the unfiltered token list (with all tokenized words in order in the text).

Reference corpus statistics

Table 3.21 shows named entities statistics by domain and by long/short text.

	Deezer		Dunkin Donuts		Landover		Mattel		Nissan		total
	L	S	L	S	L	S	L	S	L	S	
Company	50	32	62	48	49	27	99	106	66	57	596
Person	15	17	10	6	9	9	32	13	2	2	115
Geo-loc	11	1	13	6	10	5	10	5	4	11	76
Facility	4	0	2	0	3	1	3	1	0	0	14
Product	60	37	33	23	91	45	50	32	101	62	534
Media	3	16	0	1	2	1	14	16	1	1	55
Sportsteam	0	0	0	2	0	0	0	2	0	2	6
Job-title	2	1	4	2	4	0	11	1	0	4	29
Other	0	1	2	0	4	1	8	5	1	0	22
Total	145	105	126	88	172	89	227	181	175	139	1447

TABLE 3.21: Synthesio corpora named entities statistics

3.5 Conclusion on Synthesio's data

In this chapter, we first studied a subset data from Twitter. We analyzed most frequent tokens and categories. We then extracted data from three domains of Synthesio's large dataset and analyzed the most frequent contexts and most frequent tokens. Then, with the objective to limit the vocabulary size, some rewrite rules with regular expression were applied. After that, a simple normalization process replaced tokens with their most frequent forms in terms of uppercase and lowercase. With these procedures, we trained a word2vec model on this large dataset of Synthesio texts to be compared with Google's word2vec model (trained on Google news texts). We tried to use Google's services to normalize some out-of-vocabulary words and explained why we did not proceed further.

In the last section of this chapter, we discussed how we created an annotated dataset using Synthesio's data. We extracted texts from different domains and different ressources (long texts and short texts). Then definitions of Synthesio's named entity types were presented. After that, the annotation was explained and statistics of the dataset were given.

Chapter 4

Improve named entity recognition results by POS tagging

In this chapter, the first section presents existing Named Entity Recognizers, and aims at explaining why Synthesio needs to develop its own Named Entity Recognition model. The annotated dataset is then presented along with some results using a cross validation approach for evaluation. After that the prediction on Synthesio's dataset are presented, domain adaptation techniques and boostraping are used to improve the NER results. This content is also presented in (Tian et al., 2016).

In later sections, the aim and necessity to develop a POS tagger to improve NER results are presented, and the difficulties of POS tagging on user-generated content (UGC) are explained. Annotated part-of-speech datasets and Synthesio's tagset definition are presented. Then experiments about the training of a model on mixed data from different domains are examined. This part of the chapter is essentially published in (Tian et al., 2015).

Lastly, the preformance of the Named Entity Recognition (NER) model using the POS tagger developed in this chapter is presented.

Contents

3.1 Analysis of a Twitter subset data **59**

3.2 Basic normalization on Synthesio's data **61**

3.2.1 Use rewrite rules for tweet lexical normalization . . . 62

3.2.2 Uppercase/lowercase normalization 62

3.2.3 Construction of Synthesio word2vec model 70

3.3 Use Google Request to find correct form for non-standard words **71**

3.4 Creation of Synthesio Reference corpus **75**

3.4.1 Data extraction 75

3.4.2 Entity type definition 78
3.4.3 Data annotation 79
3.5 Conclusion on Synthesio's data 81

4.1 First NER experiments

In this section, we first have a look at previous work concerning named entity recognition and we discuss why we need to develop our own NER tagger. Then we use features inspired by different CRFs models to define our own CRFs patterns and we show the results of our first NER experiments.

4.1.1 Existing Named Entity Recognizers

There exists already a large number of named entity recognizers as shown in table 4.1.

Toolkit name	Program Language	License	Comment
ABNER	Java	CPL	for biomedical text
Apache OpenNLP	Java	Apache License 2.0	
GATE	Java	LGPL	
TwitIE	Java	GNU	a GATE pipeline
SpaCy	Python, Cython	MIT	multilingual
Stanford NER	Java	GNU v2	using CRFs

TABLE 4.1: NER Toolkits

Licenses of softwares are important because Synthesio wants to be able to modify and to integrate NER components into their own natural language processing (NLP) pipeline. Synthesio does not directly sell NLP software, but only the analysis automatically produced by NLP software. At the same time, Synthesio's development team do not want to publish its code so some licenses are fitting and others are not. Synthesio first tests softwares and study license items before deciding to use it in their final product.

A requirement of Synthesio is to control the NLP pipeline, that is, to have all source codes in a private library without calling external packages or programs. The Synthesio NLP pipeline is written in Python, so Synthesio is looking for an open-source NER Python library or if it does not exist, the company wants to create its own NER tagger in Python.

ABNER [1] (A Biomedical Named Entity Recognizer) is an open source text mining program that uses linear-chain conditional random field sequence models. It automatically tags genes, proteins and other entity names in text. As it processes only biomedical text, all CRF features are related to bio-medical domain affixes and lexicons. For example, if training sequences contain "PML/RAR alpha," "beta 2-M," and "kappa B-specific DNA binding protein", they will all be labeled with PROTEIN (Settles, 2004).

The Apache OpenNLP [2] library is a machine learning based toolkit for the processing of natural language text. It supports the most common NLP tasks, such as language detection, tokenization, sentence segmentation, part-of-speech tagging, named entity extraction, chunking, parsing and coreference resolution.

OpenNLP also has a command line tool which is used to train the models available from the model download page on various corpora. This function is potentially interesting for Synthesio but OpenNLP is written in Java and in Synthesio, the whole Natural Language Processing (NLP) chain is written in Python and Synthesio only wants to promote codes in Python. OpenNLP's Named Entity Recognition (NER) is based on previous steps in the pipeline (as language identifier and tokenizer) and even all translate in Python is no appropriate for Synthesio.

GATE [3] is an open source software toolkit and its has an information extraction module ANNIE (a nearly-new information extraction system) (Cunningham et al., 2002). This module proposes the following full solutions:

- Document Reset
- Tokenizer
- Gazetteer
- Sentence Splitter
- RegEx Sentence Splitter
- Part of Speech Tagger
- Semantic Tagger
- Orthographic Coreference (OrthoMatcher)

[1] http://pages.cs.wisc.edu/~bsettles/abner/
[2] https://opennlp.apache.org/
[3] https://gate.ac.uk/

- Pronominal Coreference

TwitIE is a full GATE pipeline and is available as part of the GATE Twitter plugin. This toolkit contains the following component:

- Social media data Language identification
- Twitter tokenizer, for handling smilies, user names, and URLS
- Twitter part-of-speech tagger
- Text normalization
- Information extraction

However Synthesio has already its own models of language identification and tokenizers. We are only looking for a NER model for Social media data.

SpaCy [4] is is an open-source software library for advanced Natural Language Processing, written in the programming languages Python and Cython. The library is published under the MIT license and currently offers statistical neural network models for English, German, Spanish, Portuguese, French, Italian, Dutch and multi-language NER, as well as tokenization for various other languages.

Since Spacy appeared in February 2015, Synthesio had already began the NER project with CRFsuite since December 2014, so we kept this software aside, but it is a tool which should have be evaluated.

The Stanford Named Entity Recognizer [5] is a CRFs java implementation with various options for defining feature extractors. It also contains models trained on different languages and with texts from different domains. This Named Entity Recognizer defined named entities such as CoNLL2003, that are:

- 3 classes: `Location`, `Person`, `Organization`
- 4 classes: `Location`, `Person`, `Organization`, `Misc`
- 7 classes: `Location`, `Person`, `Organization`, `Money`, `Percent`, `Date`, `Time`

In addition, this Named Entity Recognizer is trained on well-formed texts and (Ritter et al., 2011) tested the Standford Named Entity Recognizer on their tweet dataset. Figure 4.1 shows an example of a tweet prediction.

[4] https://spacy.io/
[5] https://nlp.stanford.edu/software/CRF-NER.shtml

[Yess]$_{ORG}$! [Yess]$_{ORG}$! Its official [Nintendo]$_{LOC}$ announced today that they Will release the [Nintendo]$_{ORG}$ 3DS in north [America]$_{LOC}$ march 27 for $250

FIGURE 4.1: An example of Stanford NER prediction

"Yess" and "Nintendo" are out-of-vocabulary words and they are recognized as an organization name and a location name respectively.

However, their CRFs features and template may be useful for us and we will discuss it in the next section.

4.1.2 Existing CRFs Named Entity Recognizer features and patterns

As discussed in the previous section, Synthesio needs to create its own Named Entity Recognizer with a Python implementation. This NER tagger is supposed to be right after the Synthesio process language identification and tokenizer, so this model will process tokenized text and will try to label tokens in the text with named entity labels (the BIO annotation scheme can be used in this context). This task corresponds to a sequence labelling problem and it is known that the CRFs are efficient for this kind of task. When this study was performed, Synthesio was also beginning to include CRFsuite [6] into its own NLP process library because it is fast, complete and simple to integrate, so it seems to be logical to continue in this direction. An alternative implementation of CRFs is wapiti. But without any precisions, all experiments in this thesis are done with CRFsuite.

We can train our CRFs model with existing annotated corpora in Named entity. All we need at the moment is CRFs patterns.

The Stanford Named Entity Recognizer (NER) uses a CRF pattern similar to the baseline local+Viterbi model in (Finkel, Grenager, and Manning, 2005), as shown below.

- Current word
- Previous word
- Next word
- Current word character n-gram (all)

[6] http://www.chokkan.org/software/crfsuite/

- Current POS tag
- Surrounding POS tag sequence
- Current word shape
- Surrounding word shape sequence
- Presence of word in left window (size 4)
- Presence of word in right window (size 4)

Brown cluster information is added as an option in this Stanford Named Entity Recognizer. Distributional similarity features improve performance but these models require somewhat more memory.

As we can see, the Stanford team used word value as it is in the text, plus the current word shape. They did not precise the meaning of word shape but we think features like "if the word begins with a uppercase letter" is included.

The principal contribution is the Java code implementation of the CRFs algorithm. The English NER models were trained on a mixture of CoNLL, MUC-6, MUC-7 and ACE named entity corpora, and as a result the models are fairly robust across domains.

Later on, more studies about CRFs patterns and CRFs implementations have been made accessible.

N.V, Mitra, and Ghosh, 2010 tried to extract named entities in geological texts. A precise definition of named entities was given: country name, state name, mineral name (like Zinc), waterbodies (like Indian Ocean), person and organization. They used multiple features for their CRFs pattern: F= W_{i-2}, W_{i-1}, W_i, W_{i+1}, W_{i+2}, |prefix| <= 3, |suffix| <= 3, POS tag, Digit information.

Context word feature: Previous and next words of a particular word.

Word prefix: A fixed length prefix of the current and/or the surrounding word(s).

Word suffix: Word suffix information (assists in identifying NEs). For example, suffixes like -pur, -bad, etc are indicators of a name of a location.

Part of Speech (POS) Information: The POS of the current and/or the surrounding word(s).

Digit features: Several binary digit features have been considered depending upon the presence and/or the number of digits in a token (e.g., ContainsDigit [token contains digits], FourDigit [token consists of four digits],

TwoDigit [token consists of two digits]), combination of digits and punctuation symbols (e.g. ContainsDigitAndComma [token consists of digits and comma].

From these patterns, we can first see that the context of the word is important but they did not use all tokens of the sequence, but only a window of size 5. That is, two words before and two words after the current word. Secondly, we can use until 3 letters of the word's suffix and prefix. Then, the POS tag of the current word can be useful to determine if the word is part of a named entity.

Nooralahzadeh Pattern

Nooralahzadeh, Brun, and Roux, 2014 proposed a CRFs pattern with a lot of features. They first applied this pattern to train a POS tagger for French User Generated Content from Seddah et al., 2012. They then showed that this pattern works better than the pattern in Ritter et al., 2011 on T-POS corpus (90.1% vs 88.3%).

Their pattern contains a lot of features covering word context, prefix/suffix, digital characters and etc. Since the CRFs model will compute a weight for each feature function, important features will be combined with an important weight. We try to add the most features possible to let CRFs choose.

Inspired by the pattern in Nooralahzadeh, Brun, and Roux, 2014, we created a similar CRFs pattern as shown in table 4.2. In this table, C means class in brown clustering.

This CRFs pattern employs features about token value, characters' types, context information (token value and characters' types) and word cluster informations. Nooralahzadeh, Brun, and Roux, 2014 trained their own Brown cluster with the data from French User Generated Content corpus (Seddah et al., 2012) on 500 classes and they also employed MKCLS cluster (Kneser and Ney, 1993). We just used the the results of the Brown cluster on English tweets.

Constant1 Pattern

(Constant et al., 2011) proposed two CRFs patterns for POS tagging in French. Since the model of (Nooralahzadeh, Brun, and Roux, 2014) works well in French and English, we try to implement the patterns in (Constant et al., 2011) to training with our data.

Table 4.20 shows features and feature locations for the model Constant1.

Feature name	Locations
w_i	i=[-2, -1, 0, 1, 2]
$w_i w_j$	i,j={(-1, 0),(0, 1),(-2, 0),(0, 2)}
$w_i w_j w_k$	i,j,k={(-2, -1, 0),(0, 1, 2),(-1, 0, 1)}
$w_i w_j w_k w_l w_m$	i,j,k,l,m=(-2,-1,0,1,2)
hasPunct,longPunct,hasNumber,allNumber	[-1, 0, 1]
allUpper, fstUpper	[-1, 0, 1]
isAbbrev, isTime, isDecimal,	[-1, 0, 1]
isURL,isEmail,isRT,isUSR,isHashTag	[-1, 0, 1]
tokenType,asciiVector	[-1, 0, 1]
Combination of capital type : allCap, shortCap,longCap,noCap,initCap,mixCap	[-1, 0, 1]
prefix/suffix : first/last n letters (0<n<10)	[-1, 0, 1]
$C(w_i)$	i=[-2, -1, 0, 1, 2]
$C(w_i)C(w_j)$	i,j={(-1, 0),(0, 1),(-2, 0),(0, 2)}
$C(w_i)C(w_j)C(w_k)$	i,j,k={(-2, -1, 0),(0, 1, 2),(-1, 0, 1)}
$C(w_i)C(w_j)C(w_k)C(w_l)C(w_m)$	i,j,k,l,m=(-2,-1,0,1,2)

TABLE 4.2: CRFs pattern inspired by Nooralahzadeh, Brun, and Roux, 2014

Feature name	Locations
w_i	i=[-2, -1, 0, 1, 2]
$w_i w_j$	i,j={(-1, 0),(0, 1),(-1,1)}
w_i, lowercaseform	0
hasDash, hasNumber, fstUpper, allUpper	0
prefix/suffix: (0<n<5)	0
AC_i: list of categories in Penn Treebank	i= [-2, -1, 0, 1, 2]

TABLE 4.3: Model Constant1

This model uses a bigram feature, current token value and its previous token value and the bigram feature for current label. The rest of the features are all unigram features. They also used a list of grammar categories of the token (and its context) in the French Treebank (Abeillé, Clément, and Toussenel, 2003). So we use here categories in the Penn Treebank.

The model Constant2 uses the same window size for all features and does not require the POS tag.

- Locations for all features : [-2, -1, 0, 1, 2]
- Features :
  w_i, fstUpper, allUpper, allNumber, allPunct, prefix/suffix (n<=3)

We can see from table 4.4 that the Constant2 model has the simplest pattern of the three.

Feature name	Locations
w_i	i=[-2, -1, 0, 1, 2]
$w_i w_j$	i,j={(-1, 0),(0, 1),(-1,1)}
fstUpper, allUpper, allNumber, allPunct	i= [-2, -1, 0, 1, 2]
prefix/suffix: (0<n<3)	i= [-2, -1, 0, 1, 2]

TABLE 4.4: Model Constant2

4.1.3 Annotated dataset for UGC in English

(Ritter et al., 2011) proposed a dataset with 2400 tweets. This dataset is tokenised and then annotated with ten types of named entities. Part of this dataset is also annotated with Part-of-Speech (POS) tags, called the "T-POS" corpus and is employed later in this chapter on Part-of-Speech (POS) tagging section.

Table 4.5 shows the number of each type. Annotations on tokens follows the convention of "BIO" labels. As for multi-word expressions: "B-entity" is for the token that begins the named entity, "I-entity" is for tokens inside the named entity and "O" for tokens that are not part of any named entities.

Entity	number
Person	449
Geo-loc	276
Other	225
Company	171
Sportsteam	51
Facility	104
Product	97
Music artist	55
Movie	34
Tvshow	34

TABLE 4.5: Annotated named entities in (Ritter et al., 2011)

We can see from the statistics of this dataset that the most frequent types of entities are "Person" and then "Geo-location". "Music artist", "Movie" and "Tvshow" are less frequent. We will first try to develop a named entity recognition model using CRFs on cross validation on this dataset before training a CRFs model and then testing with the Synthesio dataset.

4.1.4 First NER results with CRFs on corpus from Ritter et al., 2011

As mentioned in chapter 2, Synthesio wants to control its text processing pipeline: instead of using an existing NER system, the company prefer to train its own NER model with labelled data. All the natural language processing pipeline is implemented using Python based on CRFsuite.

First we based our approach on two previous works of CRFs patterns, from Constant et al., 2011 and Nooralahzadeh, Brun, and Roux, 2014. We want to test these CRFs patterns using cross validation en T-POS corpus. We divided randomly the T-POS corpus into 10 parts: 9 parts of the corpus are used for training and the 10th part is used for testing. We tested two CRFs patterns from Constant et al., 2011 in table 4.20 and Nooralahzadeh, Brun, and Roux, 2014 in table 4.2.

As for entity types, according to Synthesio's interest, we tried to merge 10 labelled entities (original entity type in T-POS) into 7.

Synthesio has few clients in the music artist, movie, tv show or sports team domains. Therefore, we classified `Musicartist`,`Movie` and `Tv show` into only one class `Media` and we placed `Sports team` into `Company` class.

Evaluation of models is done with micro-average, that is, weighted by entities numbers in each type.

Again we include orthographic, contextual and dictionary features; our dictionaries included a set of type lists gathered from Freebase. (Ritter et al., 2011) also used similar features.

	10 entity types			7 entity types		
Entity	Precision	Recall	F1	Precision	Recall	F1
Person (449)	0.67	0.42	0.51	0.67	0.42	0.51
Geo-loc (276)	0.63	0.28	0.37	0.63	0.28	0.37
Other (225)	0.33	0.12	0.17	0.36	0.13	0.18
Company (171)	0.92	0.31	0.44	0.9	0.26	0.38
Sportsteam (51)	0.1	0.01	0.03			
Facility (104)	0.51	0.22	0.3	0.47	0.19	0.26
Product (97)	0.3	0.07	0.1	0.4	0.05	0.09
Music artist (55)	0	0	0	0.03	0.01	0.02
Movie (34)	0	0	0			
Tvshow (34)	0	0	0			
Micro-Average	0.55	0.25	0.33	0.58	0.25	0.35

TABLE 4.6: NER results with Constant1 pattern

Table 4.6 shows the NER results with the Constant1 pattern, described in table 4.20. The numbers after each type of entities are numbers of named entities in the annotated dataset for evaluation.

Table 4.7 shows under the same condition with the pattern of Nooralahzadeh, Brun, and Roux, 2014, described in table 4.2.

	10 entity types			7 entity types		
Entity	Precision	Recall	F1	Precision	Recall	F1
Person (449)	0.65	0.47	0.54	0.63	0.43	0.5
Geo-loc (276)	0.56	0.32	0.39	0.56	0.31	0.4
Other (225)	0.4	0.19	0.24	0.37	0.18	0.24
Company (171)	0.89	0.34	0.46	0.8	0.28	0.39
Sportsteam (51)	0.2	0.04	0.07			
Facility (104)	0.57	0.36	0.41	0.62	0.35	0.42
Product (97)	0.42	0.07	0.12	0.45	0.09	0.14
Music artist (55)	0.2	0.03	0.06	0.27	0.05	0.08
Movie (34)	0	0	0			
Tvshow (34)	0.02	0.02	0.02			
Micro-Average	0.56	0.25	0.33	0.57	0.29	0.36

TABLE 4.7: NER results with Noor pattern

We can see from the two results from table 4.6 and table 4.7 that the complex pattern from Nooralahzadeh, Brun, and Roux, 2014 works only a little better than the pattern Constant1 from Constant et al., 2011. Since the complex pattern from from Nooralahzadeh, Brun, and Roux, 2014 took a largely longer time to train a model and obtained only 0.01 more on F1-measure comparing to the pattern from Constant et al., 2011, we continue our experiments with Constant1 pattern.

Table 4.8 shows the results with Constant1 model with ten types of named entities and different configurations of parameters C1, C2 $\in$ {0, 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1} and we obtained the best F1-measure on the cross validation on dataset from Ritter et al., 2011.

C1	C2	Precision	Recall	F1-Measure
1	1	0.71	0.42	0.53
0	1	0.74	0.49	0.59
0.03	0.0625	0.74	0.48	0.58
0.001	0	0.79	0.37	0.51
0.01	0.03	0.71	0.53	0.60

TABLE 4.8: NER results with constant1 pattern

To be compared with the CRFs model in the literature, Ritter et al., 2011 has obtained 0.73 in precision, 0.61 in recall and 0.67 in F1-measure.

Type error :

- Can't believe the Super City is nearly open !
- Company (gold) vs Geo-Loc (predicted)

Missing :

- Today wasz Fun cusz anna Came juss for me <3(: hahaha
- Person

Noise :

- Check us out - we're FEATURED on iTunes this week !
- iTunes : Product

Alignment error :

- Checking out this weeks tell'em Steve Dave Podcast
- predicted : Steve Dave Podcast : Person
- gold: Steve Dave : Person

4.1.5 CRFs NER models on Synthesio annotated data

Macro	Deezer	Dunkin	Land	Mattel	Nissan
Precision	0.56	0.09	0.31	0.18	0.02
Recall	0.14	0.06	0.05	0.07	0.01
F1-measure	0.22	0.07	0.08	0.08	0.01
Micro	Deezer	Dunkin	Land	Mattel	Nissan
Precision	0.74	0.39	0.56	0.19	0.06
Recall	0.09	0.08	0.05	0.02	0.02
F1-measure	0.16	0.13	0.09	0.03	0.03

TABLE 4.9: Synthesio corpus evaluation results

We can see that the model trained on the Ritter corpus does not perform well on Synthesio data, compared to cross validation results on the Ritter corpus in Ritter et al., 2011. This clearly shows these data come from a different domain. The training data from Ritter et al., 2011 comes only from Twitter, the annotated Synthesio dataset comes from different ressources (main stream and social media) and is more product-oriented

Meanwhile, all Synthesio data are also different because they are from different domains (food and coffee, automobile, etc). Moreover, none of the annotated part of these corpora is big enough to build an effective model.

4.1.6 Iterative training

Synthesio has large amount of unlabelled texts in its database. We thus tried to employ these unlabelled data to improve the NER task results, using the model trained on the Ritter corpus as in Putthividhya and Hu, 2011.

We first extracted all texts in the `Deezer` domain published during one day (2015 October 5th) in forums, blogs etc., what we call *long texts* (in contrast to texts coming from tweets, which are shorter). We then filtered repeated text sequences and similar text sequences. We obtained an unlabelled long text `Deezer` data with $1M$ sequences, that is more than $41M$ tokens. These data are quite noisy, with some sequences really different from well-formed English texts.

Our iterative training procedure follows (Garcia-Fernandez, Ferret, and Dinarelli, 2014). The idea is to annotate unlabelled data with an initial model (here the CRF model trained on the Ritter corpus). We then pick up all annotated sequences for which the model has a confidence score higher than a given threshold. We add these sequences to the initial data to train a new model. At this stage, we keep the same features for the Ritter corpus and the predicted Synthesio data. This process repeats until no more sequence passes the threshold. This procedure was originally applied to sentiment classification. In the NER task, where most of the labels are “O" (“Outside" an entity in a BIO annotation), most sequences which pass a high threshold (for example 0.9) were predicted with only “O" labels. In order to add only meaningful sequences to the training data, we add only sequences which pass the threshold and contain at least one named entity.

We chose first a high threshold of 0.8 and there were 1608 sequences with more than one entity. We also tested 0.9 but there were only 189 sequences with at least one named entity. Then for each Synthesio domain, kept as test domain out of five domains, we trained a mixed model with the Ritter corpus, plus these 1608 predicted sequences and the other eight corpora in the other domains (4 of *short texts* and 4 of *long texts*). We can evaluate thus on all Synthesio reference corpora.

The table 4.10 shows the results of this procedure with only a first iteration. As we can see the model performs far better than the baseline 4.9 for Dunkin Donuts, Mattel and Nissan, but not for Deezer. Among the 1608 sequences annotated, even the word “Deezer" was not labelled as an entity.

We also found by comparing the table 4.9 and the table 4.10 that the f1-measure of all four domains are improved except the domain of "Deezer". However, the predicted sequences that we added into the training data were

long text	Deezer	Dunkin	Land	Mattel	Nissan
Precision	0.68	0.5	0.23	0.55	0.58
Recall	0.13	0.24	0.08	0.28	0.09
F1-measure	0.22	0.32	0.12	0.37	0.16
short text	Deezer	Dunkin	Land	Mattel	Nissan
Precision	0.5	0.6	0.05	0.49	0.3
Recall	0.08	0.23	0.06	0.19	0.09
F1-measure	0.14	0.33	0.05	0.27	0.14

TABLE 4.10: Evaluation of a mixed model trained with the Ritter corpus and predicted sequences with a confidence score more higher than 0.8.

all in the domain of "Deezer". We can see that our model can still be improved by adding more annotated data, even annotated data from other domains. Since we have much more selected data than the original corpus Ritter, we also tried to filter annotated text sequences with a threshold of 0.9. The table 4.11 shows the number of selected sequences for each domain after first prediction.

Domain name	Accepted	Entities
long Nissan	9	9
short Nissan	6	6
long Mattel	1010	1056
short Mattel	19	24
long LandRover	2846	3057
short LandRover	47	48
long DunkinDonuts	22550	24156
short DunkinDonuts	102008	103176
long Deezer	1054	1105
short Deezer	32469	33166

TABLE 4.11: Number of selected sequences in each domain

From these selected sequences, we can note some examples:

Argyle fan in North Yorkshire

FIGURE 4.2: Example of "North Yorkshire"

Here our model annotated the "Geo-loc" entity "North Yorkshire" even when this phrase is absent in our training data, which means the model is somehow able to infer generalisations.

Figure 4.3 is an example of boundary error. This "job-title" entity should be the whole phrase "witch doctor" but our model extracted only "doctor".

but people are saying that she's a witch doctor

FIGURE 4.3: Example of "witch doctor"

In this example, neither company "Nissan" nor product "Note Nismo" is extracted as entity, but "Japan" is annotated correctly as "Geo-loc" entity.

As shown in table 4.11, the unsupervised Synthesio data is not homogene. Some domains have more selected data than others. We leave as future work how this data could be used to trained (perhaps with a higher confidence probability) to create a more suitable training dataset, and maybe to use an entity list as CRF feature to improve the recall.

4.1.7 Domain adaptation with reduced features

Since Synthesio data are different from those of the Ritter corpus, we consider them like two different domains. Following the annotation adaptation approach in Raymond and Fayolle, 2010, we consider the Synthesio data as the target domain, the Ritter corpus as the source domain. We thus use the complete set of features for Synthesio data and only token values and pos tags (from Synthesio pos-tagger which tags 17 grammatical categories) for the Ritter Corpus. This affects the importance given by the CRF model to features extracted from the two corpora, giving more weight to those extracted from the target domain. Here, the best model is the one which takes into account only a window of size 3 (that is tokens in positions -1, 0 and 1).

The table 4.12 shows our results for the long-text and short-text Deezer corpora with a model trained with Ritter plus the Synthesio corpus with the full set of features (baseline templates). Here the Synthesio corpus contains the 8 reference corpora provided by Synthesio and the 1608 sequences obtained in the previous experiment.

Model	Corpus	Precision	Recall	F1
full features	long text	0.83	0.08	0.15
	short text	0.42	0.03	0.06
token1+pos5	long text	0.63	0.06	0.11
	short text	0.52	0.03	0.06
token3+pos3	long text	0.67	0.06	0.11
	short text	0.41	0.04	0.07

TABLE 4.12: Evaluation of model with Ritter on reduced features with Deezer.

We can see that compared to the “full features" model, the model with reduced features gives a slightly better result on Twitter data (short-text corpus).

4.2 Aim and necessity to develop a POS tagger

The main goal of our proper POS tagger is to help our NER task.

We tested our first NER model with CRFs using as feature the POS tag of the token in the Penn Treebank corpus as in previous section. Now we want to develop a customized POS tagger trained on User Generated Content (UGC), and using the predicted POS tag as feature of our NER model. We should thus be able to see whether we get a better result in NER task.

At the same time, as described in chapter 1, the goal of this thesis is to help Synthesio to process their data better and analyse the sentiment of the text.

The POS tagger could also help to find words that play the role opf a topic in a discussion (proper noun, common nouns, etc), or words which may contain sentiment information (verbes, adjectives, etc). Synthesio has its own tokenizer and the POS tagger would be the next process in the chain.

There exists already some POS taggers, such as CRF Tagger [7], Stanford tagger [8], even commercial software LingPipe [9] for canonical text in English and SEM [10] (Constant et al., 2011) for French canonical text, but we need a customized POS tagger for our NER task.

First, we work with User Generated Content (UGC). Sometimes we can not parse the entire phrase as journal-like texts, so we do not need a delicate grammatical categorization.

Secondly, compared to existing POS tagger trained by tweets in English like the CMU Tweet POS tagger [11] and the GATE Twitter part-of-speech tagger [12], Synthesio process 80 natural languages which means we need a universal definition of POS tags.

At last, the POS tagger in (Gimpel et al., 2011) is trained on the tweet corpus, with their proper definition of POS tag set. For tokens like "we'll", they

[7] http://crftagger.sourceforge.net/
[8] https://nlp.stanford.edu/software/tagger.shtml
[9] http://alias-i.com/lingpipe/
[10] http://www.lattice.cnrs.fr/sites/itellier/SEM.html
[11] http://www.cs.cmu.edu/~ark/TweetNLP/
[12] https://gate.ac.uk/wiki/twitter-postagger.html

have a tag "pronounverb", but in Synthesio, its tokenizer already separated "we" and "'ll" so this exist postagger can not be fit for our task.

4.3 Difficulties of developing a POS tagger for UGC

In the token level, User Generated Content (UGC) contains lots of abbreviations, spelling errors, emoticons and other specialities which do not exist in canonical text. Tweets contain also sometimes tokens which do not exist in canonical text such as "RT" for retweet, at mention and hashtags.

In user generated content (UGC), like SMS (Short Message Services), many abbreviation are created and expanded over time like "ASAP" for "as soon as possible", "OMG" for "Oh my god", etc. As for tweets, they are limited to 140 characters (this limit has been doubled up to 280 characters since September 26th 2017), so they may contain more abbreviations than SMS.

The usage of certain numbers to replace a word to shorten the text is also common like using "2" for "to" or "too", or using "4" instead of "for", etc. Certain symbols became significant like "+" for "plus", "&" for "and", etc. These numbers or symbols should be tagged as their correct forms as preposition or verb.

Figure 4.4 is an example of a tweet from the corpus (Ritter et al., 2011).

Today wasz Fun cusz anna Came juss for me <3 : hahaha

FIGURE 4.4: An example of tweet

In this example of 4.4, there are multiple difficulties for our POS tagger :

- spelling errors: wasz (was), cusz (because), juss (just)
- uppercase/lowercase inversions: Fun (fun), anna (Anna), Came (came)
- emoticon: <3
- interjection: hahaha

The correct form of this tweet 4.4 should be:

Today was fun because Anna came just for me <3 : hahaha

FIGURE 4.5: Correct form of tweet 4.4

In the view of canonical text, in this tweet, only four tokens (among 12) are correct: "Today", "for", "me" and the punctuation ":". This tweet remains comprehensible for a human, but causes multiple problems for a POS tagger.

Words with spelling errors are often out-of-vocabulary (OOV) words. They would be tagged as `unknown` words with a POS tagger trained by canonical text. Words beginning with an uppercase character are often proper nouns if they are not in the first of a sentence. Emoticons are often combinations of punctuations or numbers strings, they express together a face or a sentiment. In the example 4.4, the token "<3" represents a heart if the reader leans his head on the right. But a POS tagger trained by canonical text would probably separate the punctuation "<" and the number "3", or tag them together as an unknown word. The interjection "hahaha" repeats the syllable "ha" three times. So "hahaha" is also considered as an OOV word.

omg simone is coming over then 2morrow we foin 2 da fall festival cant wait 4 GAC 2night!

FIGURE 4.6: Number usage in tweets from (Ritter et al., 2011)

In this tweet 4.6, we can find:

- abbreviation: omg (oh my god)
- uppercase/lowercase: simone (Simone)
- spelling errors: foin (going?), da (the), cant (can't)
- number replacing words or syllables: 2morrow (tomorrow), 2 (to), 4 (for), 2night (tonight)

We can see that the number "2" can not only replace the word "to", but also the syllable "to" in other words like "tomorrow" and "tonight". Sometimes a spelling error can also be caused by a missing apostrophe ' like "cant" for "can't". At last, we believe that the word "foin" may mean "going" as the letters "f" and "g" are neighbors in the keyboard. These OOV words could be tagged as `unknown` by a POS tagger trained by canonical text.

Figure 4.7 shows another problem for POS tagger.

Eduardo Surita: your a freaking ... `http://tumblr.com/xmciuda0t`

FIGURE 4.7: Another exemple of tweet

In this example 4.7, the token "your" should be two tokens "you" and "'re" to be correct syntactically. And the tokenizer would probably separate the token "you're" by two tokens "you" and "'re" (that is what the Synthesio tokenizer does). Since we process only tokenized text in this thesis, we will not correct this sort of errors. So "your" would be tagged as `pronoun` but not `pronoun` and `verb`.

Another characteristic of tweets is its system of RT (retweet), urls, hashtags (symbol # followed by a character string) and at-mentions (symbol @ followed by a username).

The pattern of a retweet for another tweet begins with "RT : " followed by the content of original tweet.

Url are often at the end of a tweet like shown in the example of figure 4.7.

Hashtags (token begins with "#") help Twitter users to label their tweets and one can tap on a hashtag to see all the tweets related to that topic. An at-mention, the symbol @ followed by a username, insures that the Twitter username is mentioned in the tweet and to add this user on a conversation that's currently happening.

The problem with POS tagger is that, the hashtag token or at-mentions token could be part of the sentence or not according to author's wish.

The hashtag in the example Figure 4.8 is a proper noun which modifies "age".

My #twitter age is 458 days 0 hours 3 minutes 49 seconds

FIGURE 4.8: Hashtag as part of sentence

Figure 4.9 shows a tweet with a hashtag which itself is a noun phrase as the attribute of the sentence "it's time to get out". There is not only "its" written without apostrophe, but also "TimeToGetOut" written written as a single token (without blank characters between words) and begin with an uppercase character "#TimeToGetOut".

On Thanksgiving after you done eating its #TimeToGetOut unless you wanna help with the dishes

FIGURE 4.9: Hashtag with multi-words

This kind of hashtags need a specific tokenizer before analysis. But we will not process these hashtags in this thesis.

Similarly, at-mentions could be a syntactic part of the sentence as attribute in the example 4.10 or not, as in the example 4.11. But hashtags at the end

New book blogger @GennaSarnak launches weekly feature , Poetry Sunday: http://tinyurl.com/47vbdy5 #Books #Poetry

FIGURE 4.10: At mention as part of sentence

of the tweet like 4.10 "#Books" and "#Poetry", often do not have any syntactic function in the sentence. Their interest is just for the tweet to be easier to be found.

Ryerson Quidditch team this Sunday at 4 p.m. Anyone know where to get cheap brooms ? #Ryerson @RUQuidditch #Rams

FIGURE 4.11: At mention at the end of sentence

We can also see that in the tweet shown in figure 4.11, the verb "know" should be its form in the third person singular "knows". That is also a spelling error. It is comprehensible for a human but causes problems for an automatic language parser.

The hashtags and at-mentions make tweets difficult to analyze with POS tags. In this thesis, we will simplify that by processing at-mentions and hashtags with label `Hashtag` and `Username` whatever their syntactic function is in the sentence like in (Ritter et al., 2011). This means that, these labels can appear wherever they want in the text.

4.4 Annotated part-of-speech datasets

As mentioned in the first section, we need to build a cutomized POS tagger trained on user generated content (UGC).

At this moment, there are three labelled POS tag microblog datasets, an IRC text corpus and a SMS dataset available.

The T-POS corpus (Ritter et al., 2011) contains 787 tweets composed of 15185 tokens and annotated with 45 tags from the Penn Treebank annotation scheme (Marcus, Marcinkiewicz, and Santorini, 1993) four special POS tags specific to annotate data from twitter: RT for retweet, HT for hashtag, USR for at-mentions and URL. That is 49 tags in total.

The DCU dataset (Foster et al., 2011) is related to sentence parsing. This corpus of 14k tokens contains 519 sentences from Twitter. Links, usernames and hashtags are all annotated as proper nouns and are assimilated to a single word noun phrase. "RT" is annotated as a noun within a single word noun phrase. This strategy is useful for parsing as canonical text, but in our

case, it is important to keep track of at-mentions and hashtags since these expressions are relevant for the task.

The ARK corpus made of of 39K tokens (Gimpel et al., 2011) has its own tokenizer and its own tagset, which is simpler than the tagset of the Penn TreeBank. For example, in the corpus, the `VB`, `VBD`, `VBG`, `VBN`, `VBP`, `VBZ`, and `MD` (for infinitif verb) tags from the Penn TreeBank are grouped into one category of `VERB`. The developer of this corpus also defined combined tags like `pronoun+verb` for tokens like "you're" (only one token) and "I'm". This corpus has less part-of-speech tags and the performance of POS tagging was 92.8% accuracy (Owoputi et al., 2012).

For our own problem, we had a similar idea: to define a simpler tagset as only one tag `VERB` could be used for all verbs. But the Synthesio's tokenizer is more similar to T-POS (Brendan O'Connor's Twitter tokenizer (O'Connor, Krieger, and Ahn, 2010)) and we have already used T-POS for the NER task (although there is only a small intersection of the two corpora for the two tasks). That is the main reason why we want to train our POS tagger with the T-POS dataset.

At last, the IRC text and SMS dataset represents messages that one sends to another (point to point). In Synthesio, texts to process are mainly one to others (one sends for public). So these two datasets are not important to be studied.

4.5 Synthesio tagset definition

As mentioned in the first section, the objective of this POS tagger is to help the named entity recognition (NER) task, and possibly also sentiment analysis. The user generated content (UGC) that we try to process at Synthesio, is often non-canonical text that may not contained well formed sentences as can be found in journalistic texts. We may not need a part-of-speech tagset precise as Penn Treebank and T-POS. Instead, we can use a tagset with less tags like the ARK corpus (Gimpel et al., 2011). In addition, Synthesio processes 28 different languages so we need a tagset for all languages.

(Petrov, Das, and McDonald, 2012) proposed a universal POS tagset of 12 tags for 22 different languages as well as mapping tables for 25 different Treebanks, including the Penn Treebank. The table 4.13 shows Penn Treebank's tagset definition and the mapping with this universal tagset.

As mentioned before, we want to extract named entities in the text. Whether a noun is a proper noun or a common noun is important. So we separated

Universal Tag	PTB Tag	Definition	Comment
NOUN	NN	common noun, singular or mass	
	NNS	common noun, plural	
	NNP	proper noun, singular	
	NNPS	proper noun, plural	
VERB	MD	modal verb	all verbs that do not take an -s ending in 3rd person singular present like can, dare, etc
	VB	verb, base form	this tag subsumes imperatives, infinitives and subjunctives
	VBD	verb, past tense	includes the conditional form of the verb "to be"
	VBN	verb, past participle	
	VBG	verb, gerund or present participle	
	VBP	verb, present tense, other than 3rd person singular	
	VBZ	verb, present tense, 3rd person singular	
ADJ	JJ	adjective and ordinal number	
	JJR	Adjective, comparative	adjectives with the comparative ending -er and a comparative meaning
	JJS	Adjective, superlative	adjectives with the superlative ending -est
Adverb	RB	Adverb, negation	words end with \-ly, degree words like quite, too and negative markers like not and never
	RBR	Adverb, comparative	adverbs with the comparative ending -er and a comparative meaning
	RBS	Adverb, superlative	adverbs with the superlative ending -est
	WRB	Wh-adverb	includes how, where, why etc
DET	DT	Article	Determiner
	PDT	predeterminer	determiner-like elements when precede an article or possessive pronoun (quite/PDT a mess)
	WDT	Wh-determiner	includes which and "that" when used as a relative pronoun
	EX	Existential there	there
PRON	PRP	personal pronoun	
	PRP$	prossessive pronoun	
	WP	Wh-pronoun	
	WP$	possessive wh-pronoun	
CC	CC	Coordinating conjuction	includes and, but, nor, or, yet and mathematical operators plus, minus, etc
ADP	IN	preposition or subordinating conjuction	preposition precedes a noun phrase or a prepositional phrase, subordinate conjunction precedes a clause
PRT	RP	particle	
	POS	possessive ending	's, or just ' as in parents'
	TO	to	for the preposition "to"
X	UH	Interjection	includes "yes", oh, please, see, uh, well, etc
	SYM	mathematical, scientific and technical symbols or expressions	names of chemicals, units of measurements should be tagged as nouns
	FW	foreign word	
	LS	list item marker	letters and numerals when they are used to identify items in a list
NUM	CD	cardinal number	
PUNCT	**-LRB-** # **-RRB-** $ " , . : ″ LS	LS, **-LRB-** and **-RRB-** : itemize list markers	each punctuation has itself as tag

TABLE 4.13: Universal tagset and Penn Treebank tagset mapping

the category NOUN into two different categories: NN for common noun (singular and plural) and NNP for proper noun (singular and plural). Practically this means that for the Penn Treebank categories, we regrouped NN (singular common noun) and NNS (plural common noun) and for proper noun, we regrouped NNP (singular proper noun) and NNPS (plural common noun).

Inspired by this universal POS tagset mapping for the Penn Treebank, we only made minor modifications for the mapping of POS tagset of T-POS.

For verbs, as done by ARK dataset, we regrouped `VB`, `VBD`, `VBG`, `VBN`, `VBP`, `VBZ`, and `MD` (infinitif) as one category of `VERB`.

At last, we did as T-POS, to process separately the four specific categories of Twitter: RT (retweet), HT (hashtag), URL and USR (at-mention).

Table 4.14 shows the mapping of the Synthesio tagset, the Penn Treebank tagset and the T-POS tagset.

Synthesio tags	Tag Penn Treebank	Tag T-POS	Comment
NN	NN NNS	NN NNS	common nouns
NP	NNP NNPS	NNP NNPS	proper nouns
VB	MD VB VBD VBG VBN VBP VBZ	MD VB VBD VBG VBN VBP VBZ	verbs
ADJ	JJ JJR JJS	JJ JJR JJS	
ADV	RB RBR RBS WRB	RB RBR RBS WRB	WRB : *where, when*
DET	DET PDT WDT EX	PDT DT WDT EX TD	PDT : *half*
PRON	PRP WP PRP$ WP$	PRP PRP$ WP WP$	pronouns
CC	CC	CC	
PREPCS	IN	IN	prepositions and subordinate conjuction
PRT	POS TO RP	POS TO RP	*particle*
X	# $ FW SYM INTJ	INTJ SYM FW	
NUM	CD	CD	number
PUNCT	" , -LRB- -RRB- . : ” LS	" ( ) . , : NONE O LS	
RT		RT	*Retweet*
HT		HT	*Hashtag*
URL		URL	
USR		USR	at mention

TABLE 4.14: PTB, Ritter and universal tagset correspondance

We can see from the table that the tagset of T-POS has minor differences with the Penn Treebank tagset.

Label "TD" was also only once for the token "a" in the context "doing a sit up" so we consider that to be an error for "DT" (determiner). Label "O" was only one case for a token of punctuation ("..). That maybe an annotation error so we took it into category punctuation. Label "None" was used for hooks [and] so we consider them as punctuation.

4.6 Feature space for a POS tagger

Considering the definition of a part-of-speech tag, there are a lot of features that may be useful to categorize tokens. These features form a feature space for the POS tagging task. They are also useful to create a pattern for models

like Conditional Random Fields (CRFs). CRFs pattern is defined by a set of features and their window sizes. Since CRFs compute automatically weights of each feature function, we want to define as many features as possible and let CRFs choose functions more or less important.

4.6.1 Features related to character type

This kind of features contains two types of features according to the value that they return.

Binary features

These features return a True or False value.

allLetter: all characters in the token are alphabet letters. allNumber: all characters in the token are digital numbers (without symbols nor punctuations) hasNumber: the token contains one or more digital numbers onePunct: the token has only one character and it is a punctuation allPunct: the token contains only punctuation character(s) hasPunct: the token contains one or more punctuation character(s) longPunct: the token contains only punctuation characters (more than one) hasDash: the token contains the dash "-"

Some features can be completely included in another feature, for example, onePunct and allPunct are included in hasPunct (if one token contains one punctuation or contains only punctuation, it returns True for the feature "hasPunct"). In the training phase, CRFs will determine which feature functions are more important than others by defining different weights.

Token type feature

This feature "tokenType" aimes to imitate characters which form the token. We use "X" to replace all uppercase letters, "x" to replace all lowercase letters and "9" to replace all numbers in the token. We then obtain a value which has only "X", "x" and "9" as characters. For example, this feature returns "9xxxxx" for "2night", "XxXxxxxx" for "McDonald" and "xXxxxx9" for "iPhone7".

4.6.2 Features about uppercase/lowercase letters

For canonical text, only proper nouns and the first word of a sentence an uppercase letter at the initial of the word and other letters in lowercase and some proper nouns and abbreviations can have all letters in uppercase (like WTO for World Trade Organization). In the titles of articles, books' sections,

full words begin with uppercase letters. This kind of features is important to detect proper nouns. As showed before, some User Generated Content does not always follow this rule. So we define here all possibilities of uppercase/lowercase alternance.

- fstUpper: if the token begins with a letter in uppercase and other letters in lowercase.
- shortCap: if the token has only one uppercase character.
- longCap: if the token has only uppercase characters (more than one).
- mixCap: if the token has on the same time uppercase character(s) and lowercase character(s).
- hasUpper: if the token has at least one uppercase letter.
- allUpper: if the token contains only uppercase letter(s).
- allLower: if the token contains only lowercase letter(s).

4.6.3 Features about token value

Token values

These features concern different forms to represent the token.

- tokenValue: the token form in the text
- lower: the token value all in lowercase
- ASCIIVector: the token represented by a vector ASCII code (American Standard Code for Information Interchange) corresponding tot the letter of the word under consideration. For example, this feature returns "99-97-116" for "cat" (on ASCII code).

Prefix/suffix features

These features concern token prefix and suffix.

We use from 1 to n first characters as prefix and from 1 to n on the right for suffix. Here we count until n=10.

4.6.4 Features using regular expression

These features are used to find special tokens like URL, hash tags, etc.

isDecimal: the token is a decimal number, including numbers with comma "," or even e for Euler's number.

isEmail: the token contains symbol @ but not the first character, and a period before the end of the token.

isAbbrev: the token contains only uppercase letters, maybe separated by period like "U.N." for United Nations.

isDate: if the token represents a date.

isTime: if the token represents a time.

isRT: if the token is exactly "RT".

isUSR: if the token begins with @ (is an at-mention).

isHashTag: if the token begins with # (is an hash tag).

isURL: if the token is an url link.

4.6.5 Features using extern ressources

Similarly to NER task, here we use POS tag in Penn Treebank and the Brown clusters results as extern ressources.

Penn Treebank POS tag

We first change all the tokens in the Penn Treebank corpus into its form with all letters in lowercase, and then we count the number of possible POS tag that this token have in the corpus Penn Treebank. Then we find that one token in lowercase can have up to 8 possible POS tags in the Penn Treebank. That is "a" and "down".

a							
NN	FW	SYM	LS	JJ	IN	DT	NNP
10	8	11	2	2	1	24691	56

TABLE 4.15: POS tags of token "a" in Penn Treebank

down							
RP	RB	NN	RBR	VBP	JJ	IN	NNP
257	468	2	1	1	14	194	2

TABLE 4.16: POS tags of token "down" in Penn Treebank

So we created eight features related to 8 possible POS tags in the Penn Treebank, from the most frequent to the less frequent. For tokens having less

possible POS tags, the other possible POS tags are defined as "not concerned" (NCND), this is the value that the feature returns. When two tags have the same number of frequency, they will ordered by tags' names in alphabet.

token	the most frequent POS in PTB	2nd most frequent	3rd	4th	5th	6th	7th	the less frequent
a	DT (24691)	NNP (56)	SYM (11)	NN (10)	FW (8)	JJ (2)	LS (2)	IN (1)
down	RB (468)	RP (257)	IN (194)	JJ (14)	NN (2)	NNP (2)	RBR (1)	VBP (1)
papers	NNS (36)	NNP (2)	NCND	NCND	NCND	NCND	NCND	NCND

TABLE 4.17: Penn Treebank POS tags feature values examples

Table 4.17 shows the 8 features about POS tag in Penn Treebank corpus for three tokens: "a", "down" and "papers".

Brown cluster on tweets

Brown clusters is a hard hierarchical agglomerative clustering technique based on distributional information proposed by (Brown et al., 1992) and have been used successfully in NER applications (Miller, Guinness, and Zamanian, 2004), (Liang, 2005), (Ratinov and Roth, 2009), etc.

(Owoputi et al., 2012) proposed to use Brown clusters on tweets and they obtained 1000 clusters for 56,345,753 tweets (that is 847,372,038 tokens). These tweets are from a sample of 100k tweets per day from September 10th 2008 to August 14th 2012.

Here we just use their results as word class.

class	token	frequency
0010100	juuuuust	84
0010100	kust	92
0010100	jhus	102
0010100	jussss	103
0010100	jhuss	105
0010100	justs	106
0010100	jussst	116
0010100	jusx	117

TABLE 4.18: Sample of Brown Clusters for Tweets

Table 4.18 is a sample of the clusters results[13].

The first column shows the class that the token belongs to, among the defined 1000 classes. The hierarchical is included in the class code. From the left to the right, the positions of the class code means the hierarchical level. Then the value of 0 or 1 shows that it is a binary classification, so there

[13] http://www.cs.cmu.edu/ ark/TweetNLP/

is only two classes in each level. For example, class "001010" has two son classes, "0010100" and "0010101". As we can see, this clustering method can recognize that all tokens in the table 4.18 are in the same class. In fact, they are all variances of the word "just" and they are all out-of-vocabulary (OOV) words. The Brown cluster class information can be useful for OOV words presented in tweets. And with this clusters results, we can define numbers of classes we want, among 2, 4, 8 until 2^{14}. For example, if we want only $2^6 = 64$ classes, we can keep only the first 6 numbers of these class code and ignore all descendants of one class are considered as belonging to this class.

4.7 Experiments of CRFs POS tagger and results

Since we are working with tokenized text, that means that training a POS tagger corresponds to sequence labelling. CRFs has already been applied successfully to train a POS tagger as in (Pandian and Geetha, 2009), (Silfverberg et al., 2014) and (Constant et al., 2011) for French. We try to train our own POS tagger with T-POS corpus and with our defined tagset.

4.7.1 Experiments uni-corpus

In this part of experiments, we try to train our CRFs models with patterns defined in the previous section. First, we divide randomly the corpus T-POS in three part: 80% for cross validation, 10% for development and 10% for test. In the training of CRFs models, regularization is usually required to prevent the model from over fitting the training data. The two most common regularization methods are L1 and L2 regularization. L1 regularization penalizes the weight vector for its L1-norm (the sum of the absolute values of the weights), and L2 regularization uses its L2-norm (Tsuruoka, Tsujii, and Ananiadou, 2009). As in (Nooralahzadeh, Brun, and Roux, 2014), we uses the elastic-net penalty of the form:

$$\rho_1 * |\theta_1| + \frac{\rho_2}{2} * |\theta_2|^2$$

To get the best L1 and L2 parameters, we use 5-block cross validation and choose the best combination of L1 and L2 parameters from L1 $\in$ {0, 0.0625, 0.125, 0.25, 0.5, 1, 2, 4, 8, 16} and L2 $\in$ {0, 0.0325, 0.0625, 0.125, 0.25, 0.5, 1, 2, 4, 8, 16} as in (Owoputi et al., 2013). We added {0.01, 0.03, 0.1, 0.3} for L1 and L2.

The table 4.19 shows for three variations of this CRFs pattern. The first variation used all the 14 values from the class code in Brown Cluster trained by tweets. That means, we have 2^{14} different classes for all tokens. We also added the bigram feature Y_{i-1} for current label Y_i (chapter 2).

As for the seconde variation, we used only unigram feature Y_i and we did not consider the bigram, as in (Nooralahzadeh, Brun, and Roux, 2014). But we kept 2^{14} classes corresponding to Brown Cluster feature.

The third variation used only first the 9 positions of Brown Cluster class code. That means, we have $2^9 = 512$ classes for all tokens, which is closer to 500 classes in (Nooralahzadeh, Brun, and Roux, 2014). We also use only the unigram feature. This is the most similar model to (Nooralahzadeh, Brun, and Roux, 2014).

For each variation of this model, we show in the table 4.19 the two best results (with best regularizations L1 and L2), and the last column shows the result on development data with these two best regularizations.

Model name	L1	L2	cv result	dev result
14 values in Brown Cluster	0.03	0.0625	0.8628	0.8947
	0.0625	0.0625	0.8616	0.8912
unigram features	0.0625	0.0625	0.8660	0.8863
	0.03	0.625	0.8640	0.8724
first 9 values for brown (Nearest to the reference)	0.25	0.5	0.8638	0.8745
	0.03	0.0625	0.8672	0.8891

TABLE 4.19: Results with model Nooralahzadeh2014

As we can see in the table 4.19, we got the cross validation best results with the third variation of the model, that is the most similar to (Nooralahzadeh, Brun, and Roux, 2014). But the best result with development data was given by the first variation of the model, that is, with model including more classes in Brown Clusters.

We also discovered that the model (Nooralahzadeh, Brun, and Roux, 2014) uses a lot of CPU ressources, to generate all these features. Maybe that is why the authors of the corpus used only unigram features. Our implementation with CRFsuite also requires a dynamic generation of feature values for each experiment (keep all values in the memory instead of keep on storage on the disk). So this model requires lots of ressources and time. As Synthesio processes data in real time, we want to test simpler models.

(Constant et al., 2011) proposed two CRFs patterns for POS tagging in French. Since the model of (Nooralahzadeh, Brun, and Roux, 2014) works

well in French and English, we try to implement the patterns in (Constant et al., 2011).

Table 4.20 shows features and feature locations for the model Constant1.

Feature name	Locations
w_i	i=[-2, -1, 0, 1, 2]
$w_i w_j$	i,j={(-1, 0),(0, 1),(-1,1)}
w_i, lowercaseform	0
hasDash, hasNumber, fstUpper, allUpper	0
prefix/suffix: (0<n<5)	0
AC_i: list of categories in Penn Treebank	i= [-2, -1, 0, 1, 2]

TABLE 4.20: Model Constant1

This model uses a bigram feature, that is the current token value and its previous token value and the bigram feature for current label. The rest of the features are all unigram features. There is also feature of a list of grammar categories of the token (and its context) from the French Treebank (Abeillé, Clément, and Toussenel, 2003). So we use here categories in Penn Treebank as mentioned in the previous section.

We tested regularization of L1, L2 $\in$ {0 ,0.001,0.003,0.01,0.03,0.1,0.3}. Similar to previous experiment, we first tested all combinations of L1 and L2 with cross validation (with the same part of corpus) and we kept only two best combinations of L1 and L2. Then we tested these two best combinations to development data.

Table 4.21 shows that the best combination of L1 and L2 regularization is l1=0.01 and l2=0.001, and is close, although not better than the model in (Nooralahzadeh, Brun, and Roux, 2014).

c1	c2	CV results	dev result
0.03	0.01	0.8701	0.8856
0.01	0.001	0.8699	0.8926

TABLE 4.21: Model Constant1 results

This result encouraged us to test to also test a simpler model pattern, the model Constant2, to see if it works as well as the model Constant1.

The model Constant2 uses the same window size for all features and does not require the categories in a Treebank.

- Locations for all features : [-2, -1, 0, 1, 2]
- Features :
  w_i, fstUpper, allUpper, allNumber, allPunct, prefix/suffix n<=3

We tested regularizations L1, L2 $\in$ {0 ,0.001,0.003,0.01,0.03,0.1,0.3} and we got the results in table 4.22.

c1	c2	CV results	dev result
0.03	0	0.8399	0.8068
0.01	0.001	0.8374	0.8445

TABLE 4.22: Model Constant2 and results

This result is worse than the model Constant1.

If we want to get a better report of performance and time, it is better to use the model constant1.

4.7.2 Experiments with multi-corpus

T-POS remains a small annotated corpus (with only 787 tweets) and the Penn Treebank is much larger. But the Penn Treebank was texts reporting news from Wall Street Journal and the POS tagger trained on canonical text performs poorly with User Generated content (Ritter et al., 2011), (Gimpel et al., 2011).

In this section, we want to test our CRFs models with a mixed training corpus. That is, to use mixed of Penn Treebank and T-POS with different proportions of Penn Treebank, to see if we can get a better POS tagger.

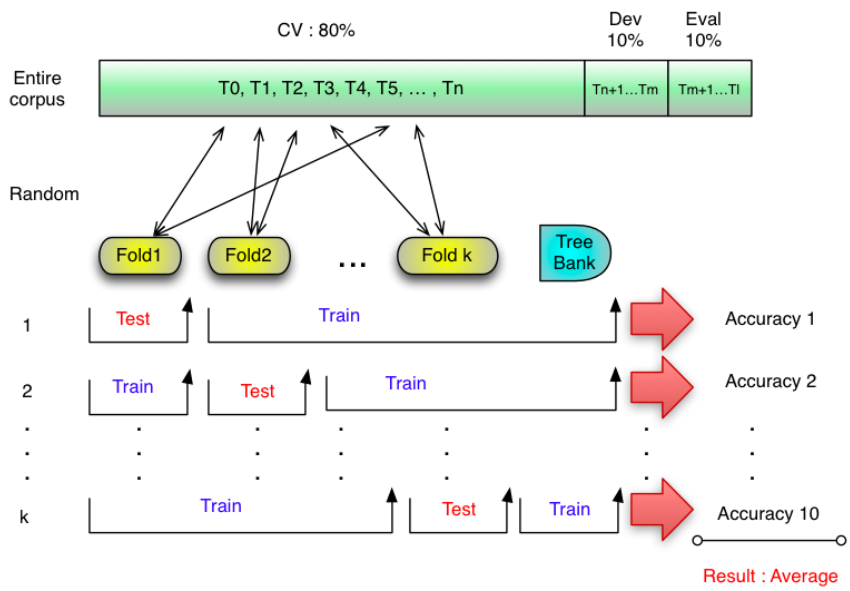

FIGURE 4.12: Cross validation adding Penn Treebank data

As showed in the figure 4.12, for cross validation evaluation, we add some proportions of Penn Treebank to the training data and then test the model

with the last fold of data as previous experiment. In the end, we always calculate the average result.

Model Name	mixed corpus	proportion with Ritter	c1	c2	result on cv
Constant2011	None	None	0.03	0.01	0.8701
Nooralahzadeh	PTB	1:1	0.25	0.125	0.8689
		4:1	0.25	0.125	0.8718
		9:1	0.25	0.125	0.8712
Constant1	PTB	9:1	0.001	0.125	0.8725
			0.125	0.0625	0.8726
			0	0.25	0.8671
			0.001	0.01	0.8728

TABLE 4.23: Results of mixed-corpora experiments

Table 4.23 shows the results of two CRFs patterns: (Nooralahzadeh, Brun, and Roux, 2014) and Constant1. The seconde line of the table is what we obtained in the section of uni-corpus training.

4.7.3 Results with artificial examples

Before creating Synthesio's annotated POS tag corpus, we want to see the developed POS tagger's performance on some artificial sentences. Here we want to apply our POS tagger to some concrete sentences as showed in 4.24. These sentences are created following context of a tweet. As discussted in the previous section, the model in Nooralahzadeh, Brun, and Roux, 2014 uses too many features and it took a long time to train a model for our applicative context. When we try to use the model to process any sentence, we should first generate all features for every token in the sentence. The model Constant1 does not work as well as the model in Nooralahzadeh, Brun, and Roux, 2014 but took much less time for training. In the long term, Synthesio may think about training regularly (for example every week) a new model on its own data, so training time is important. That is why we chose the model Constant1 as our final POS tagger, and we want to test it with some artificial sentences and some tweets.

The first two phrases were made artificially to test the behavior of our POS tagger in regard to some proper nouns. We can find this style to talk about a product in tweets. They contain products names and are for objective to give some commentary on these products. That could be a text talking about a product in Synthesio to process. The third sentence is a real tweet which contains some person names that have never been seen from any training

data. That is to test the POS tagger to recognize proper nouns that are absent from training data. Because in reality, new words and new names appear everyday. The POS tagger should be able to give them the correct POS tag instead of a UNKNOWN tag. The fourth sentence is a real text in Synthesio, chosen with a product name and a company name. Product names are not evident to detect and sometimes they maybe a multi-word expression.

1	Yesterday I bought a new Renault. It was cool, better than my old Ford focus.
2	got an iPhone6 for my birthday, better than my old BlackBerry, wonderful!
3	A paper by Maggie Simpson and Edna Krabappel from New York was accepted by two scientific journals `http://vox.com/e/7103628?utm_campaign...`
4	If I had one wish it would be to drive a bugatti V for a day!

TABLE 4.24: Examples to test POS tagger

Table 4.25 shows the results of our POS tagger (from model Constant1) with the first sentence in 4.24.

token	Yesterday	I	bought	a	new	Renault	.	It	was	cool	,
True	NN	PRON	VB	DET	ADJ	NP	P	PRON	VB	ADJ	P
Pred	NP	PRON	VB	DET	ADJ	NP	P	X	X	X	P
token	better	than	my	old	Ford	focus	.				
True	ADJ	PREPCS	PRON	ADJ	NNP	NNP	P				
Pred	ADJ	PREPCS	ADV	ADJ	NNP	NNP	P				

TABLE 4.25: Prediction results with first sentence
"Yesterday I bought a new Renault. It was cool, better than my old Ford focus."

As we can see, there are errors in the first sentence. "Yesterday" is tagged as proper noun but it is an adverb. That maybe due to its position in the sentence and it begins with a uppercase letter. "It was cool" are all tagged as "X", we believe that means unknown words. Maybe the phrase is too short (only three words) and in the training corpus, maybe there does not exist this sort of construction of phrases. "my" is tagged as adverb instead of pronoun. That must due to the tag sequence "PREPCS+ADV+ADJ+NNP". That is not impossible to have this sequence in the training corpus. This error is due to ill-formed tweets in the training data.

Table 4.26 shows the results of the POS tagger prediction with a sentence similar to the first sentence. We can see that the first word of this sentence begins with a lowercase letter and the POS tagger did not found the correct tag ("VB") by tagging with "X" (UNKNOWN). "iPhone6" is tagged as number because there is a number "6" in the token. And then "my" is also tagged as

token	got	an	iPhone6	for	my	birthday	,	
True	VB	DT	NN?	PREPCS	PRON	NN	P	
Pred	X	DT	CD	PREPCS	ADJ	NN	P	
token	better	than	my	old	BlackBerry	,	wonderful	!
True	ADJ	PREPCS	PRON	ADJ	NNP	P	ADJ	P
Pred	ADJ	PREPCS	ADV	ADJ	NN	P	ADJ	P

TABLE 4.26: Prediction results with second sentence
"got an iPhone6 for my birthday, better than my old BlackBerry, wonderful!"

adverb instead of pronoun as in the first sentence. At last, "BlackBerry" is tagged as common noun but not proper noun.

Table 4.27 shows the prediction result with the third sentence. We can see that our POS tagger has problem with the boudaries of the proper noun "New York" that it thought the three following tokens "was accepted by" were all proper noun. At last, it tagged "..." as X (UNKNOWN) but not punctuation because in the model Constant1, there is only a feature about if token contains a dash ("-") but not if the token has punctuation characters.

token	A	paper	by	Maggie	Simpson	and	Edna	Krabappel	
true	DT	NN	PRCS	NP	NP	CC	NP	NP	
pred	DT	NN	PRCS	NP	NP	CC	NP	NP	
token	from	New	York	was	accepted	by	two	scientific	journals
true	PRCS	NP	NP	VB	VB	PRCS	NUM	ADJ	NN
pred	PRCS	NP	NP	NP	NP	NP	NUM	ADJ	NN
token	urlhttp://vox.com/e/7103628?utm_campaign...					...			
true	URL					P			
pred	URL					X			

TABLE 4.27: Prediction results with third sentence
"A paper by Maggie Simpson and Edna Krabappel from New York was accepted by two scientific journals urlhttp://vox.com/e/7103628?utm_campaign... ..."

Table 4.28 shows our POS tagger's prediction with the fourth sentence. All tokens were correctly tagged except "bugatti", which begins by a lower-case letter. Instead of proper noun, it is tagged as common noun.

token	If	I	had	one	wish	it	would	be	
True	PRCS	PRON	VB	NUM	NN	PRON	VB	VB	
Pred	PRCS	PRON	VB	NUM	NN	PRON	VB	VB	
token	to	drive	a	bugatti	V	for	a	day	!
True	PRT	VB	DT	NP	NP	PRCS	DT	NN	P
Pred	PRT	VB	DT	NN	NP	PRCS	DT	NN	P

TABLE 4.28: Prediction results with fourth sentence
"If I had one wish it would be to drive a bugatti for a day!"

4.7.4 Compare T-POS tagset and Synthesio tagset

In this section, we try to use our pattern in k-fold cross validation (with k=10) on T-POS corpus to compare the original T-POS tagset (49 tags) and Synthesio tagset (18 tags).

We also tested our idea of adding annotated Penn Treebank texts into training data to improve the cross validation result. We did first a cross validation with only T-POS corpus (with two tagsets). Then we added gradually the same quantity (1:1), four times plus (1:4) and nine times plus (1:9) of Penn Treebank sequences, to see if the POS tagging results are better (always with two tagsets).

As for optimization, we first let $L2 = 0.00001$ (default value in wapiti [14]), and we tested possible value of L1 $\in$ {0.01, 0.03, 0.1, 0.3, 1, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6,1.7, 1.8, 1.9, 2.0} and obtained the L1 value which gives the best result (accuracy in average for k-fold). Then we use this L1 value and tested L2 $\in$ {0.00001, 0.0001, 0.001, 0.01, 0.1, 0.5, 1} and obtained the best result.

Cross validation with only T-POS

tagset	L1	L2	accuracy average
Ritter	0.1	1	87.21%
Universal	0.01	1	89.25%

TABLE 4.29: cross validation with corpus T-POS

From the results in table 4.29, we can see that Synthesio tagset has obtained 3% more in accuracy average, that means, just like in (Gimpel et al., 2011), tagset with less tags has better performance than tagset with more tags.

Although Synthesio tagset imports more possible sequences than before, this results gave us chance to a better syntactic analyse for Twitter data.

Mixed model

In this section, we are trying to create three mixed models with different proportions of data from Penn Treebank are added into the training data of the T-POS cross validation part.

We propose here first to add the same numbers of sequences of Penn Treebank into training data, then four times plus and nine times plus. These three parts of Penn Treebank are disjoint. We stopped at nine because we think more than nine times plus could not consider the part of T-POS corpus.

[14] https://wapiti.limsi.fr/

The mixed training data is created like followed: for each iteration of cross validation, the part of Penn Treebank added into training data remains the same, only the fold of T-POS corpus change, to get one results of the cross validation.

The evaluation of models are calculated by average of 10 folds. The results are showed in the table 4.30.

We optimized L1 and L2 with the same procedure of the previous section.

Tagset	Proportion	L1	L2	Average of accuracy
Ritter	1:1	1.5	0.0001	85.40%
Ritter	4:1	1.8	0.001	86.72%
Ritter	9:1	1.9	0.0001	87.18%
Synthesio	1:1	1.0	0.01	89.11%
Synthesio	4:1	1.0	0.5	89.27%
Synthesio	9:1	1.6	0.01	88.95%

TABLE 4.30: Results of mixed models with different tagsets

This result shows that with the T-POS tagset, adding sequences from the Peen Treebank is not really helping for a better average of accuracy than without in cross validation (we had obtained 87.21% as accuracy).

That is because these two corpora are not so similar: the sequences in Penn Treebank added unknown tokens and new tags sequences. These data can not help to better predict Twitter sequences in T-POS corpus.

It seems that the results are lightly better with Synthesio tagset, which the average of accuracy shows a light improvement.

We also noticed that with Synthesio tagset, more sequences added from Penn Treebank (mixed with T-POS) can improve the POS tagger's performance.

4.8 Experiments using POS tagger results for NER

In this section, we use the prediction of our POS tagger as features for our CRFs model and this feature takes two words before and two words after plus current word as context window. This POS tagger is trained by both PTB dataset and T-POS dataset with the proportion of 4:1.

Table 4.31 shows the results of each named entity type. Old scores without POS tagger are presented before flash to be compare with. Score in green are better.

Even though precision of company reaches 1.0 and the recall of person type increased from 0.29 to 0.41, we can see that almost all company names in

	Precision	Recall	F-Measure
Company	0.125/1.0	0.03/0.03	0.05/0.05
Person	0.36/0.35	0.29/0.41	0.32/0.38
Other	0.5/0.5	1.0/1.0	0.67/0.67
Product	0.5/0.5	0.05/0.03	0.10/0.05
Media	0/0.5	0/0.07	0/0.12
Geo-Location	0/0	0/0	0/0
Job title	0/0	0/0	0/0
Micro-Average	0.28/0.34	0.09/0.1	0.11/0.13

TABLE 4.31: NER results using POS tagger as feature

different domains are missing, both geo-location and job title named entities are also missing.

We believe that the reason of this is that the user-generated content contains a lot of out-of-vocabulary words, non-standard words and syntactic structures that are different from well-formed text as showed in section 4.3. (Ritter et al., 2011) also showed that traditional natural language processing (NER) tools like POS tagger and Named Entity Recognizer performed must worse than on well-formed texts.

In future chapters, we want to transform user-generated content into almost well-formed text, that means normalize some lexical words by replacing them with correct forms.

Chapter 5

Lexical normalization of tweets with neural networks

In our days, we find more and more user generated content (UGC) through the Internet, including forums, tweets, products reviews, etc. Due to user's habit (intentionally) or because of the limited number of characters allowed by the medium of communication, especially Twitter autorised only 144 characters in one tweet, these texts contain specialities compared to journalistic standard text (canonical text): they contain new words (like "Brexit"), spelling errors (like "im" for "I'm"), abbreviations (like "u" for "you", "b/c" for "because") and out-of-vocabulary words (like "mooooo" for "more"). This writing behaviour poses problems for existing natural language processing (NLP) such as tokenisation, part-of-speech (POS) tagging, chunking, parsing and in our case, Named Entity Recognition (NER). The tools performing these tasks face a degradation of their performance.

In order to improve the results, we want to add a pre-treatment that will replace the parts of non-canonical text. This step is called lexical normalization, and the words to be corrected are called Non-Standard Words (NSW).

Contents

4.1 First NER experiments **83**
4.1.1 Existing Named Entity Recognizers 83
4.1.2 Existing CRFs Named Entity Recognizer features and patterns 86
4.1.3 Annotated dataset for UGC in English 90
4.1.4 First NER results with CRFs on corpus from Ritter et al., 2011 91
4.1.5 CRFs NER models on Synthesio annotated data . . . 93
4.1.6 Iterative training 94
4.1.7 Domain adaptation with reduced features 96

4.2 Aim and necessity to develop a POS tagger 97
4.3 Difficulties of developing a POS tagger for UGC 98
4.4 Annotated part-of-speech datasets 101
4.5 Synthesio tagset definition 102
4.6 Feature space for a POS tagger 104
4.6.1 Features related to character type 105
4.6.2 Features about uppercase/lowercase letters 105
4.6.3 Features about token value 106
4.6.4 Features using regular expression 107
4.6.5 Features using extern ressources 107
4.7 Experiments of CRFs POS tagger and results 109
4.7.1 Experiments uni-corpus 109
4.7.2 Experiments with multi-corpus 112
4.7.3 Results with artificial examples 113
4.7.4 Compare T-POS tagset and Synthesio tagset 116
4.8 Experiments using POS tagger results for NER 117

5.1 Annotated datasets for the lexical normalization of tweets

5.1.1 Dataset from (Li and Liu, 2014)

(Li and Liu, 2014) collected a dataset of 2,577 Twitter messages (from the Edinburgh Twitter corpus in (Petrović, Osborne, and Lavrenko, 2010)) and this dataset is used in (Pennell and Liu, 2011).

5.1.2 Dataset from workshop ACL2015

The workshop of ACL2015 proposed a task based on noisy user-generated text (WNUT) to normalize noisy text (tweets) (Baldwin et al., 2015).

The organisers provided a dataset of 3k tweets with about 48k tokens.

For annotators' convenience, all corrections for Non-Standard Words are in lowercase, so even all standard words' corrections are all in lowercase.

From the table 5.1 we can see that even for a dataset built especially for the detection of Non-Standard Words (NSW), the number of NSW is only

class	number of tokens	percentage	comment
OOV-SW	22247	46%	named entities, punctuation, numbers
IV-SW	22138	45.8%	
OOV-NSW	3557	8%	
IV-NSW	386	0.8%	

TABLE 5.1: Numbers of tokens in each class

about 8.8% of all tokens. We can also see that half of standard words are out-of-vocabulary tokens. That means it would be difficult to classify tokens into the four classes.

(Li and Liu, 2015) proposed some features to this task of classification.

- number of vowel/consonant characters
- maximum consecutive vowel/consonant characters
- if the token contains consecutive three times the same character

We used these features to classify all tokens in this dataset with a 10-folds cross validation and we obtained an accuracy about 60%.

5.1.3 Typology Analysis of labeled corpora

(Tarrade et al., 2017) proposed a typology of NSW in French including a morpholexical analysis and a morphosyntactic analysis. Their work was based on tweets and SMS in French but it is also useful for English. Here we used their classes that concerned English language phenomenon. According to their definition of NSW substitution table in figure 5.1, we can get for English classes in table 5.2.

5.2 Use SVM for NSW and SW classification

Based on table 2.11 concerning methods of text normalization, we want to first try to classify NSW and SW words with SVM. As defined before, NSW words are Non-Standard Words that we can correct with another form. But SW words case is more complicated. First, there are protected categories like url, at mention and hashtags. These words do not exist in any vocabulary (OOV) but we cannot correct them. So they are considered as standard words. Secondly, punctuations and numbers are not always standard words. Sometimes the symbol "@" replaces the word "at", "+" replaces "plus" and "&" replaces "and". As for numbers, "4" could be another way to write "for" in

Substitution	graphie complète d'un mot ou groupe de mot	correspondance phonologique	lettre		c->s'est/ses/sais/sait, g->j'ai, i->y, k->qu'à/cas, l->elle, m->aime, n->haine, o->eau/au, q->cul, r->aire/air, s->est-ce, u->eu
				initiale	b->bé (aussi interjection), c->c'est/ces, d->des/dé/dès, e->eu, h->hache, j->j'y, o->oh, p->paix/pet, t->t'es/tes, v->vais
			chiffre		7->cet
			symbole		+ ->plus
		approximation phonologique			k->que, c->se/ça, z->je
	graphie partielle d'un mot	correspondance phonologique	lettre		bo->beau, Dcédé->décédé, ossi->aussi
			chiffre		2main->demain
			symbole		+sieurs->plusieurs
			graphie		bizes->bises, bisoux->bisous, mwa-moi
				consonne double	come->comme
				chute du "e" instable	douch->douche
				lettre muette enlevée	fou->fous, pa->pas, peu->peut, vou->vous
				lettre muette ajoutée	peux->peu, as->a
				initiale muette	ôtel->hôtel
			signe diacritique		voila->voilà, tantôt->tantot
		approximation phonologique			nan->non, kikou->coucou(avec graphie du 'c' modifiée), pô->pas, ui->oui
			signe diacritique		ca->ça, appelé->appele
	typographie				est ce que->est-ce que
		casse d'un caractère			suzanne -> Suzanne
		élision			m en->m'en
	rébus	caractère spécial			de grandes @ (oreilles)
		emoji			Merci pour vos RTs &
	écrasement				jsuis->chuis (précédemment : agglutination+compactage)
	code-switching				yes -> oui, screen->écran
	néologisme/jargon				catche (compris), walou (rien)
	verlan				wam->moi
	contraction				p'tit->petit
	autre				chzton->chaton, les->la

FIGURE 5.1: NSW substitution

complet graphe for one word	phonological correspondance	letter		y->why
			initial	b->be, w->with
		number		4->for, 2->to
		symbol		+->plus
	phonological approximation			da->the
partial graphe for one word	phonological correspondance	letter		ur->your
		number		2morrow->tomorrow
		written form		cuz->because
			double consonnant	
			mute letter removed	wat->what
			mute letter added	
	phonological approximation			ya->you
typography	letter case			anne->Anne
	elision			dont->don't
rebus	special character			@ (for ears)
	emoji			
crushing				
contraction				gonna->going to
other				

TABLE 5.2: NSW substitution in English

some tweets. Third, that is also which interested us, named entities. They may not exist in the vocabulary (OOV) but they are already in their standard form so we cannot correct them. At last, there are noises like "jkkk". In the tweet "One of my favourite songs, Asu wa kuru kara", the second part of the tweet is the rewrite part of another language (Coreen). It is a named entity (an album name) but considered as noise for the English corpus. But we will classify these tokens as standard words, because we cannot correct them.

For these two corpus: Li and Liu, 2015 and Baldwin et al., 2015 , we removed duplicated tweets. Sometimes, only the same tweet repeats with the same annotation and sometimes, they have different annotation. In this case, we choose a correct annotation according to their context and kept only one. Then we also removed tweets who have only correct tokens since we want to maximize the NSW percentage in the text. Finally we obtained the two datasets as showed in table 5.3.

reference	number of tweets	number of tokens	number of NSW
Li and Liu, 2015	2303	36414	4051 (about 11%)
Baldwin et al., 2015	1955	7832	1157 (about 15%)

TABLE 5.3: Two datasets for text normalization and statistics

For SVM classifier, we chose features independent of word's context and we considered here only features of one token by character.

- nbVowel, nbConsonant, maxiNbVowel, maxiNbConsonant, has3SameLetter, len, inAspell, inSynthesioVocab,
- nbLetter, nbNumb, nbPunct ,
- allLetter, singleLetter, hasNumber, allNumber, isDecimal, onePunct, allPunct, hasPunct, longPunct,
- isAt, endWithIn, endWithNt, endWithVe,
- isUSR, isRT, isURL,isHashTag

We want to eliminate influence of doubled tweets for tokens, so we removed ambiguous tokens in doubled tweets that we kept only the correct form. But with different contexts, there always exists tokens that in some cases are NSW and SW in other cases. We have three sort of strategy for this problem. First, we count the most frequent case. For one token, if the occurrence number of NSW is more than SW word, we tag it as NSW, or SW otherwise. In case where the numbers are equal, we count the token as NSW. That is our choice because NSW is often rare in corpus and we want to keep this NSW case. The second one is also in the idea of keeping rare NSW cases, that is to keep a token as a NSW as soon as it is once NSW in the corpus. That means, once the token tagged as NSW, it is considered as NSW in all context. The third possibility is to consider the token as SW as soon as it is once tagged as SW. This strategy privileged the SW and we do not think it help to detect NSW, so we did not test this case.

The training process is defined like this: we did first cross validation with 5 folds using the two datasets. For each experiment, we tested all occurrence and the most frequent case. To generalize our model, we tried to train the model with one dataset and test with the other dataset.

Table 5.4 showed the results of 5-folds cross validation with the two datasets. We tested the 5 classes of features (F1 to 5) and we showed in this table the best result with each model and each dataset. VS is the feature of vocabulary of Synthesio, that is the vocabulary generated in the previous section. This feature depends on if the token is present in the Synthesio vocabulary or not. The column "Model" showed the strategy of considering all token occurrence "all Occ" and that of most frequent class "Freq". The last three columns showed recall, precision and F1-measure of the NSW class.

Model	Corpus	Features	Accuracy	Recall	Precision	F1
all Occ	2577 tweets	F12-VS	0.9206	0.6652	0.5791	0.6188
MFreq	2577 tweets	F12345	0.7743	0.6174	0.8078	0.6994
all Occ	WNUT	F1	0.913	0.7153	0.6459	0.6787
MFreq	WNUT	F125-VS	0.8583	0.6189	0.1117	0.1881

TABLE 5.4: Results of cross validation in 5 folds

Table 5.5 showed the results of training a model with one dataset and test with the other.

Train	Test	Features	Accuracy	Recall	Precision	F1
2577	WNUT	F1245-VS	0.8841	0.5927	0.5949	0.5938
WNUT	2577	F1245-VS	0.9079	0.6472	0.3782	0.4774

TABLE 5.5: Train a model with one dataset and test with the other

5.3 Experiments of SW and NSW classification with neural networks

Based on the results of SW and NSW classification with SVM, we try to use neural network models to classify tokens in datasets. We first proposed a context-free model considering only the current token, then we proposed several convolutional models using the 5-grams (including the current word), with or without character embedding of the current word and a simple LSTM model with different argument configurations.

5.3.1 Context-free experiment

In this section we did some experiments with only the token without considering their context. So there is the same problem than the previous SVM model.

We used the dataset from (Li and Liu, 2015) with 2303 valid tweets. This dataset contains 36414 tokens including punctuations where 4051 are NSW (about 11%). The critic of a NSW is only that its correction is the same of its original form. We divided the dataset into two equal parts and used one part to train and the other part to test.

The evaluation measures contain accuracy (global evaluation) and precision, recall and f1-measure for NSW class.

The input of our neural network is one token completed by the maximum length with blanks. Each character is first changed into lowercase and then represented by its ascii code (input of neural networks should be a numeral vector).

There are two output possibilities for our neural network. One of them is two probabilities of the two classes (NSW or SW) and then we took the class with higher probability. The other one is only one dimension (between 1 and 0) with sigmoid activation function.

Our neural network contains:

- Embedding layer: characters representations vector
- Convolution layer: with different filter length
- Merge several convolution layers with different filter length
- GlobalMaxPooling layer: reduced output dimension of convolution layer
- Full-connected layer with "reLU" activation function
- Dropout layer: removed randomly some data to prevent overlapping
- Output layer: 2 dimensions with softmax function

Table 5.6 showed the results of uni-layer convolution experiments.

Table 5.7 showed results of experiments with merged multi convolution layers.

Model	Accuracy	Recall	Precision	F1
Convolution(5,relu)+GlobalMaxPooling1D	0.9243	0.7077	0.5719	0.6281
Convolution(5)+lambda				0.6063
Convolution(5)+maxpooling+flatten	0.9248	0.7396	0.515	0.6044
Convolution(5)+GlobalMaxPooling	0.9227	0.7378	0.5068	0.5916
Convolution(3,relu)+GlobalMaxPooling	0.9236	0.758	0.4727	0.5806

TABLE 5.6: Results of experiments with uni convolution layer

Model	Accuracy	Recall	Precision	F1
10 convolution couches merged (1-10)	0.9298	0.7387	0.5824	0.6503
5 convolution couches merged	0.9248	0.7235	0.5477	0.619
10 convolution 'relu' couches merged (1-10)	0.9259	0.7581	0.5101	0.602
2 convolution 'relu' couches merged (2,3)	0.923	0.7315	0.5032	0.5928
2 convolution couches merged (2,3)	0.9215	0.7149	0.5108	0.5927
3 convolution 'relu' couches merged (1,2,3)	0.9241	0.7633	0.4752	0.5828

TABLE 5.7: Results of experiments with multi convolution layers

5.3.2 5-grams experiments

As talked before, the context-free method cannot resolve the ambiguous problem and here we try to consider each word with its context, that is two words before and two words right after the current word.

As a detail, we used numbers which represent no character to mark the beginning of a sentence, position -2 and -1 if we begin a sentence with position 0. Similarly, we have two special positions in the end of a sentence, for position of the sentence length and sentence length plus 1.

From the dataset (Li and Liu, 2015), we extracted 36004 different 5-grams in total. This dataset has 36414 tokens (all occurrence included). The rest of the occurrence of 5-grams have the same label for the current word in the same 5-gram. That means, there is no ambiguous NSW/SW word since we consider the context.

In these 5-grams, there are 4044 NSW 5-grams (the current word of 5-gram is a NSW), where 2782 NSW 5-grams have only SW in context (two words before and two words after), 1262 NSW 5-grams have one or more NSW in their context. There are 31960 SW 5-grams (the current word of 5-gram is a SW) where 20542 alphabetic SW 5-grams (since punctuations are considered as SW) have only SW in their context.

First experiment

We selected randomly 20% of this dataset to be test data, to follow the same distribution as 10% NSW and 90% SW. Then the training data contains the reste 3235 NSW 5-grams and we picked up randomly the same quantity of SW 5-grams.

The network structure we chose is one of the model from the previous section. The embedding layer contains the concatenation of 5 words with each word a maximum number of characters to keep the same dimension of the input layer. Then the convolution layer merged filters of 1, 2 and 3. After that, the global max pooling layer reduced the dimension before a full-connect layer. Just before the output, a dropout removed some samples to prevent over-training and at last we used categorical cross entropy as loss function and softmax as activate function.

As the result, we got about a weak precision of 15% and a recall as 90%.

Experiment by class

Here we want to see which class is better classified, and if it is the NSW context that leads errors in NSW's label.

We divided all 5-grams into three classes:

- class1: 2782 NSW 5-grams have only SW in context
- class2: 1262 NSW 5-grams have one or more NSW in context
- class3: 20542 alphabet SW 5-grams have only SW in context

For class 1, we are sure that the model is trained with correct context. The context of class 2 contains at least one NSW so training with this context could be some wrong context. The class 3 has only SW 5-grams with only SW context.

We try to train a model with only correct context, that is context with only SW words. That is class1 for NSW and half of class 3 (the other half for test). We want to test this model on class2, NSW with NSW context and the other half of class3 for SW.

As for neural network structure, we used the character embedding for the current word then word embedding (word2vec representations from (Mikolov et al., 2013)) for context. Table 5.8 shows details of this neural network structure.

$EmbedW_{-2}$	$EmbedW_{-1}$	$EmbedW_0$	$EmbedW_1$	$EmbedW_2$	$CharEmbedW_0$
Convolution	Convolution	Convolution	Convolution	Convolution	Convolution
MaxPool	MaxPool	MaxPool	MaxPool	MaxPool	MaxPool
Hidden layer Dense(128, relu)					
Dropout(0.5)					
Output layer Dense(2, softmax), loss= categorical crossentropy					

TABLE 5.8: Neural Network structure of character and word embedding model

Table 5.9 shows the results with different classes. We can see that the global results is better than previous section and the best result is with the all SW class.

test set	accuracy	precision	recall	f1-measure
total		77.444%	77.812%	77.628%
moreThan1NSW	77.81%	1.0	77.81%	87.52%
allSW	97%	1.0	97%	98.58%

TABLE 5.9: NSW and SW classification results by class

This neural network performs better than the one of previous section.

5 words embedding and characters model with sigmoid as output

In this section, we try to use the same convolutional model structure and change only output layer. We tested 2 output for two classes with softmax function and only one output with sigmoid function.

Emb(W_{-2})	Emb(W_{-1})	Emb(W_0)	Emb(W_1)	Emb(W_2)	CharEmbW_0
Flatten ()					Convolution1D
Dropout(0.2)					MaxPooling
Hidden Layer Dense (dimension = 128, activation = relu)					
Dropout(0.5)					
Output Layer Dense (2, softmax), loss = categorical_crossentropy			Output Layer Dense (1, sigmoid), loss = binary_crossentropy		

TABLE 5.10: 5 words embedding and characters Convolutional model structure

We did three times the same experiment and we calculated the average of the three times.

times	precision	recall	f1
1	0.7023	0.5745	0.6097
2	0.6455	0.6537	0.6284
3	0.5838	0.7658	0.6561
average	0.6439	0.6647	0.6314

TABLE 5.11: 5 words embedding and characters convolutional model with softmax functon

times	precision	recall	f1
1	0.6982	0.593	0.6377
2	0.6666	0.6429	0.6376
3	0.6568	0.6463	0.6416
average	0.6739	0.6274	0.6390

TABLE 5.12: 5 words embedding and characters convolutional model with sigmoid functon

Optimisation of configurations

Now we try to optimize configurations of this model with 5-folds cross validation and then compare it with the model of previous section (only character embeddings for 5-grams).

Since Initialization of parameters are random, we did three times the same experiment to get a reliable result.

In the table 5.13, the first three lines showed the result of basic model. Then instead of completing to maximum of characters by adding blanks in the right of the word, we add blanks on the left of the word (the character order remains the same). There are also three times of experiments. Then we tried Xavier Initialization from (Glorot and Bengio, 2010) for the full-connect layer just before the output. After that, we tried dropout = 0.2 (in the basic model, this parameter is 0.5). All experiments are done with output of 2 dimensions and activation function softmax. For each experiment, the number of iteration is fixed to 100 times and we chose at last the one with minimum of loss function, that is categorical cross entropy.

	precision	recall	f1
1	0.77	0.85	0.81
2	0.70	0.88	0.78
3	0.67	0.91	0.77
left1	0.68	0.86	0.76
left2	0.74	0.80	0.77
left3	0.68	0.88	0.77
Xavier	0.66	0.90	0.76
drop0.2	0.67	0.88	0.76
drop0.2	0.67	0.87	0.77
drop0.2	0.69	0.86	0.77

TABLE 5.13: Optimisation of character embedding and word embedding model

$wordEmb(W_{-2})$	$wordEmbW_(-1)$	$wordEmb(W_0)$	$wordEmb(W_1)$	$wordEmb(W_2)$
Convolution	Convolution	Convolution	Convolution	Convolution
Max Pooling				
Full-connected layer (128, relu)				
Dropout(0.5)				
Output layer Dense(2, softmax), loss= categorical crossentropy				

TABLE 5.14: 5 words embeddings convolutional model structure

Other Neural network structures

In this section, we want to test a model without character representation of current word. That means, we will use only word embedding of all 5 words in 5-gram.

- charEmb init: Xavier,
- zeros on the right/ left
- dense relu init:
  default/ Xavier
- dropout0.5 /0.2/
  withoutdrop before output
- output dense 2 softmax
- nb epoch:100 (about 15s/epoch)

	precision	recall	f1
1	0.65	0.49	0.56
2	0.53	0.66	0.59
3	0.43	0.76	0.55
left1	0.51	0.65	0.57
left2	0.56	0.57	0.56
left3	0.58	0.56	0.57
Xavier	0.0	0.0	0.0
drop0.2	0.58	0.59	0.58
drop0.2	0.55	0.62	0.58
drop0.2	0.57	0.60	0.59

TABLE 5.15: optimisation of 5 word embedding model

Another model that we will test is based on the previous model and to change only the convolution layer into a LSTM layer.

Then we tried to test the LSTM model using 5-folds cross validation with 2577 tweets dataset. Since the LSTM model used much ressources (processor

$wordEmb(W_{-2})$	$wordEmb(W_{-1})$	$wordEmb(W_0)$	$wordEmb(W_1)$	$wordEmb(W_2)$
LSTM	LSTM	LSTM	LSTM	LSTM
Merged layer				
Full-connected layer (128, relu)				
Dropout(0.5)				
Output layer Dense(2, softmax), loss= categorical crossentropy				

TABLE 5.16: 5 words embeddings LSTM model structure

	5wordsCharAndEmbConv	5wordEmbLSTM
charEmb=XavierInit repr chars: zeros on right relu init : default dropOut(0.5) before output output = 2, softmax	Precision: 0.54 Recall:0.59 F1:0.56	Precision: 0.61 Recall:0.73 F1:0.66
charEmb=initWithSeed chars: zeros on right relu init : default dropOut(0.5) before output output = 2, softmax	Precision: 0.44 Recall: 0.58 F1: 0.50	Precision: 0.56 Recall:0.70 F1:0.62
charEmb=default chars: zeros on right relu init : default dropOut(0.5) before output output = 2, softmax	Precision:0.54 Recall:0.64 F1:0.58	Precision: 0.54 Recall:0.64 F1:0.58

TABLE 5.17: Compare convolution and LSTM model

time and memory), we first tried with 10 iteration, even such 10 iteration (for one fold) took 8580 seconds (2.4 hours).

	precision	recall	f1
cv1	0.8162	0.5030	0.6225
cv2	0.7669	0.5162	0.6171
cv3	0.7903	0.4873	0.6029
cv4	0.7317	0.5264	0.6123
cv5	0.7524	0.4767	0.5836
average	0.7715	0.5019	0.6077

TABLE 5.18: LSTM model on 2577 tweets

Then we still did with 100 epochs one time and we noticed the results in the table 5.19. This experiment took 87376 seconds (that is 24 hours).

Then we tried an option Early Stopping in Keras with 100 epochs, that took 10892 seconds (3.03 hours). Table 5.20 showed the results of each cross validation and the average scores. The result is not much better than 10 iteration, so we did not continue with this model.

	precision	recall	f1
cv1	0.7005	0.6894	0.6949
cv2	0.6903	0.6304	0.6590
cv3	0.7370	0.6212	0.6742
cv4	0.7019	0.6332	0.6658
cv5	0.7125	0.6409	0.6748
average	0.7084	0.643	0.6737

TABLE 5.19: LSTM experiments with 5-fold cross validation

	precision	recall	f1
cv1	0.8030	0.5262	0.6358
cv2	0.7846	0.4630	0.5824
cv3	0.7385	0.5381	0.6226
cv4	0.7199	0.5161	0.6012
cv5	0.7480	0.4510	0.5627
average	0.7588	0.4989	0.6009

TABLE 5.20: Early Stopping with 100 epochs

5.3.3 Experiments with pre-trained word2vec models

From the previous section, we tested three models with variances and we can see that the 5 words embedding and current word in characters with convolutional model with two dimensions output and softmax function is the best. In this section, we use only this model but try to improve the results of NSW/SW classification.

First, in stead of random initialization for word embedding, we try to use pre-trained models: one with Google's word2vec model and another with a word2vec model trained by Synthesio data. This time, our experiments are using two corpus.

Then based on the obtained results, we tested different optimizers to improve the results.

Word2vec models

Here we want to compare two word2vec modelsthe Google word2vec model (Mikolov et al., 2013) and the Synthesio word2vec model. (Mikolov et al., 2013) proposed a word2vec model trained with Google news. We call this word2vec model as Google word2vec. This model is trained with over 3 million words included numbers but not punctuation. One word could be all surface forms: all in lowercase, begin with uppercase, all in uppercase

or a mix of uppercase/lowercase characters. For example, in this model, "Nissan", "nissan" and "NISSAN" all exist with different vector values.

In Synthesio, as talked about in the previous chapter, we already normalised some types of entities as date, time, currency, etc. We also normalised tokens with uppercase/lowercase, we replaced the token with its the most frequent form and we kept forms until 60% of the total number of this token in all forms. For example, in the Synthesio word2vec model, there exists only the token "Nissan", other forms as "nissan" or "NISSAN" are all changed into this standard form. This Synthesio word2vec model is trained by three different Synthesio domains: luxury (L), music (M) and automobile (A). The choice of the three domain was considered as different one from another and they are economically interesting domains for Synthesio. Automobile (A) is the domain where there is the most data. We trained in total five word2vec model using Synthesio data including the same timeline 12h00-14h13 for all the three domains (T for all domains, L for luxury, M for music and A for automobile) and 12h00-1259 for a smaller automobile data (called a).

Table 5.21 showed the NSW/SW classification results on dataset (Li and Liu, 2014) using this CNN model with different word2vec models. In the dataset, the number of vocabulary is 8260. We showed the number of vocabulary of each word2vec model. For each word2vec model, we also calculated the number of intersections of words in the dataset and the word2vec model.

word2vec name	# of v	v in dataset	prec NSW	rec NSW	F1 NSW
keras embedding	–	–	0.6881	0.6011	0.6301 (10)
google	3 000 000	5706	0.7922	0.7789	0.784 (20)
Synthesio M	515 886	6189	0.7444	0.8081	0.7735 (50)
Synthesio L	633 706	6226	0.7658	0.8167	0.7896 (100)
Synthesio a	788 260	6360	0.8324	0.6944	0.7548 (100)
Synthesio A	1 806 789	6743	0.8173	0.7525	0.7828 (20)
Synthesio T	2 086 985	6945	0.7741	0.8237	0.7962 (50)

TABLE 5.21: CNN model with dataset in (Li and Liu, 2014)

The number in the parenthesis is epoch number in training process (number of iteration). Since Keras kept random initialization for weights and some configuration, even if we kept only the model with the minimum of loss function, we cannot get each time the same result. We executed numbers of epoch of 10, 20, 50 and 100, and then we chose the best result with the number of epoch in the parenthesis. We can also see that it is not always the most of iteration works the best.

We can see that even if the Google word2vec model has a larger vocabulary size (3 billion) than the Synthesio T word2vec model (about 2 billion), the Synthesio T word2vec model gave better classification results (0.7962 of NSW's F1-measure). When we examine the number of words in the vocabulary intersection, we can see that the Synthesio T word2vec model has the largest intersection with the dataset. However, even though the Google word2vec model has less words in the dataset, it has a better F1-measure for NSW class (0.784 vs 0.7735).

Table 5.22 showed the NSW/SW classification results on dataset (Baldwin et al., 2015) using this CNN model with different word2vec models. In this dataset, the number of vocabulary is 11710. We want to see if we get similar results than with dataset (Li and Liu, 2014).

word2vec name	# of v	v in WNUT	prec NSW	rec NSW	F1 NSW
Keras embedding	–	–	0.63	0.7253	0.6618 (10)
Google	3 000 000	8963	0.8768	0.7349	0.7989 (50)
Synthesio M	515 886	9484	0.8335	0.7272	0.7765 (20)
Synthesio L	633 706	9777	0.876	0.723	0.79 (50)
Synthesio a	788 260	9497	0.9514	0.6478	0.7704 (20)
Synthesio A	1 806 789	10000	0.8565	0.7395	0.7924 (100)
Synthesio T	2 086 985	10470	0.9167	0.7413	0.8178 (100)

TABLE 5.22: CNN model with dataset (Baldwin et al., 2015)

We can see that it is still the model Synthesio T who has the most intersection vocabulary with the dataset (Baldwin et al., 2015) and this word2vec model works the best.

Then we merged the two datasets and we did the same 5-fold cross validations (randomly). The fusion of these two corpora has 17306 different tokens. This mixed dataset has more variances in data and we want to see if our model gives similar results.

word2vec name	# of v	v in corpora	prec NSW	rec NSW	F1 NSW
Keras embedding	–	–	0.6655	0.6767	0.6654 (20)
Google	3 000 000	13324	0.7458	0.7445	0.7451 (50)
Synthesio M	515 886	13132	0.7108	0.6978	0.7014 (10)
Synthesio L	633 706	13467	0.7522	0.7677	0.7554 (20)
Synthesio a	788 260	13316	0.8128	0.6679	0.7313 (10)
Synthesio A	1 806 789	14194	0.7602	0.7388	0.7408 (10)
Synthesio T	2 086 985	14857	0.7265	0.8208	0.7696 (50)

TABLE 5.23: CNN model with 2 datasets

Conclusion to word2vec model

Among these six word2vec models, Synthesio T got the best results in all these three datasets: (Li and Liu, 2014) dataset, (Baldwin et al., 2015) dataset and the mixed of these two datasets. This word2vec also has two third of Google word2vec model's vocabulary, smaller and faster to use. In addition, with uppercase/lowercase normalization and quantity named entity normalization, the Synthesio T word2vec model's vectors are more representative and we used this word2vec model for next experiments.

5.3.4 Experiment with optimizers

In this section, we tried to test different optimizers with the same neural network model and the pre-trained Synthesio T word2vec model. Similarly, we did experiments with datasets (Li and Liu, 2014), (Baldwin et al., 2015) and the mixed of the two datasets.

Similarly, we tested with different epoch number, 10, 20, 50 and 100, with each experiment, after all iterations we kept the model with the minimum of loss function. Table 5.24 compared the results of two optimizers: rmsprop and adagrad, using the dataset in (Li and Liu, 2014) with 5-fold cross validation.

optimizer name	#epochs	prec NSW	rec NSW	F1 NSW
rmsprop	10	0.7935	0.7429	0.7662
	20	0.8203	0.7093	0.7587
	50	0.7927	0.7544	0.7729
	100	0.8075	0.7099	0.7543
adagrad	10	0.7938	0.7494	0.7707
	20	0.8088	0.7348	0.7699
	50	0.8185	0.7339	**0.7737**
	100	0.8223	0.7301	0.7733

TABLE 5.24: Optimizers experiment using (Li and Liu, 2014)

We can see from the results that the model which gives the best f1 measure of the NSW class is not the one with the most epoch number (100 epochs). For both of the optimizers, experiments with 50 iterations was the best classifier who got the best F1-measure for the NSW class.

Table 5.25 shows the same experiments with the dataset in (Baldwin et al., 2015). We added another optimizer: adadelta.

optimizer name	#epochs	prec NSW	rec NSW	F1 NSW
rmsprop	10	0.8998	0.7303	0.8049
	20	0.9014	0.7258	0.8031
	50	0.8663	0.7497	0.8031
	100	0.8557	0.7592	0.803
adadelta	10	0.888	0.7421	0.8071
	20	0.8947	0.7553	0.8191
	50	0.876	0.7684	0.8186
	100	0.8781	0.7687	0.8196
adagrad	10	0.8976	0.7361	0.8087
	20	0.9057	0.7488	0.8196
	50	0.9032	0.7564	0.8232
	100	0.9018	0.7584	**0.8239**

TABLE 5.25: Optimizers experiment using (Baldwin et al., 2015)

This time even 10 iterations experiment fund the best f1-measure score for NSW class. But the other two optimizers are helping to find better model with 100 times iterations.

We can also see from the comparison of the two datasets that with both datasets, optimizer adagrad helped to find the best model.

Then we tried the same experiment with 5-fold cross validation using the mixed of the two datasets. Table 5.26 showed the results with two optimizers: rmsprop and adagrad.

optimizer name	#epochs	prec NSW	rec NSW	F1 NSW
rmsprop	10	0.9734	0.7676	0.858
	20	0.9375	0.8531	0.8928
	50	0.8954	0.8888	0.8909
	100	0.9268	0.9037	0.9149
adagrad	10	0.9627	0.8458	0.9004
	20	0.9771	0.8899	0.9315
	50	0.9831	0.9294	0.9555
	100	0.9824	0.9413	**0.9614**

TABLE 5.26: Optimizers experiment on mixed dataset

With the mixed dataset, we have got the best score ever in 5-fold cross validation. We obtained 0.9614 of f1-measure for NSW class, it is comparable of 98% in the workshop although we mixed two datasets. We kept the same conclusion that the best optimizer is adagrad and most of the best number of iterations is 100.

Experiment with WNUT test

WNUT test is some data with similar ressources, it is just a part disjoint of the corpus WNUT2015 to be used in the test. This data contains 1967 tweets with 2776 Non-Standard Words and 26645 standard words. It is about 2/3 size of WNUT training data.

We first tested with model trained with WNUT training data, which is the standard procedure of the workshop. We varied the word embedding model, Google word2vec and the word2vec model trained by 3 domains of Synthesio data. Optimizers that we tested are rmsprop, adadelta, adagrad and sgd. We then tested the same model trained with WNUT training data and the 2577 tweets data. The idea is to add more data in the training phase to see if the model works better. Table 5.27 showed the results of this experiment.

training	word2vec	optimizer name	prec NSW	rec NSW	F1 NSW
wnut	Google	rmsprop	0.8084	0.9079	0.8553
		adadelta	0.8480	0.9024	0.8744
		adagrad	0.8881	0.8930	0.8906
		sgd	0.8908	0.9024	0.8966
	synthesio3	rmsprop	0.8044	0.9024	0.8506
		adadelta	0.8776	0.8888	0.8832
		adagrad	0.9491	0.8728	0.9094
		sgd	0.9508	0.8783	**0.9131**
both	google	rmsprop	0.7748	0.9253	0.8434
		adadelta	0.8442	0.9026	0.8724
		adagrad	0.9004	0.8905	0.8954
		sgd	0.9001	0.9015	0.9008
	synthesio3	rmsprop	0.8251	0.8966	0.8594
		adadelta	0.9122	0.8883	0.9001
		adagrad	0.9595	0.8683	0.9116
		sgd	0.9509	0.8775	0.9127

TABLE 5.27: Experiment with WNUT test, model trained with WNUT training

We can see from the results that using the Synthesio word2vec model is almost always better than Google word2vec model (except training with wnut experiment using rmsprop). For each experiment serie, using only wnut or using the two corpora, using Google or Synthesio word2vec model, the optimizer "sgd" gave always the best F1-measure result of NSW class.

Similarly, we tested in each optimizer, learning rate (lr) and learning decay. Here we used the two datasets to train our model, to see the best classification results that we could have. For each optimizer, we tested the default value (in Keras) and then tested the value of decay = 0.01*lr or decay =

0.001*lr and kept the best results. The results are showed in the table 5.28.

word2vec	optimizer name	prec NSW	rec NSW	F1 NSW
google	rmsprop lr=0.001, decay=0.0	0.7748	0.9189	0.8407
	rmsprop lr=0.001, decay=0.00001	0.7791	0.7774	0.7782
	adadelta lr = 1.0, decay = 0.0	0.8388	0.9143	0.8749
	adadelta lr=1.0, decay=0.01	0.8821	0.5227	0.6564
	adagrad lr=0.01, decay=0.0	0.8871	0.893	0.8901
	adagrad lr=0.01, decay=0.0001	0.8774	0.7374	0.8013
	sgd lr=1.0, decay=0.0	0.8295	0.8357	0.8326
	sgd lr=0.1, decay=0.001	0.9465	0.7262	0.8219
synthesio3	rmsprop lr=0.001, decay=0.0	0.8028	0.9072	0.8518
	rmsprop lr=0.001, decay=0.00001	0.6513	0.79323	0.7153
	adadelta lr = 1.0, decay = 0.0	0.915	0.8779	0.8961
	adadelta lr=1.0, decay=0.01	0.9007	0.5915	0.7141
	adagrad lr=0.01, decay=0.0	0.9449	0.8675	0.9046
	adagrad lr=0.01, decay=0.0001	0.9348	0.7233	0.8156
	sgd lr=1.0, decay=0.0	0.9509	0.8775	0.9127
	sgd lr=0.1, decay=0.001	0.9413	0.7388	0.8279

TABLE 5.28: Experiment with WNUT test, model trained with WNUT and 2577tweets

We can see that among all models, the optimizer sgd got the best F1-measure score and the optimizer rmsprop got the best recall score. We fund similar conclusion that the Synthesio word2vec model works better than Google word2vec model.

5.4 Word corrector

There are two types of corrector: first, we can only consider NSW to correct. That means we did first the classification of NSW and SW, then the corrector will only propose correct forms for NSW. Another corrector will consider all tokens in dataset. As discussed before, punctuations and numbers are also considered as SW. So this corrector will propose candidates for any NSW and SW token, but the token form as in the original text is also one of the candidate that this corrector propose and then we will try to sort the list of candidate and obtain the most probable candidate of correction.

First we want to represent tokens in the dataset. For example, we have a sentence as showed in figure 5.2. In this sentence, the NSW is "txtd" and its correct form is "texted", the user ignored the two vowel characters 'e'.

If we already know that "txtd" is a NSW and we try to find its correct form "texted", we want first represent the token "txtd" by character and with some transforms, we want to get the representation of its correct form "texted". We

I txtd you last night.

FIGURE 5.2: An example of sentence with NSW

define that the possible longest word has seven characters, the input token is showed in table 5.29. The token is completed until seven characters by adding blanks on the right of the token.

t	x	t	d			

TABLE 5.29: Token "txtd" in characters

52	56	52	36	0	0	0

TABLE 5.30: Token "txtd" representation as vector

The correct form that we want to get is "texted" as showed in table 5.31. It can be represented by a vector as in table 5.32, that would be the output of our neural network model. To output these results, in fact, the neural network will compute for each number position, the most probable number in the possible character set (among 70 possibilities including punctuations and numbers).

t	e	x	t	e	d	

TABLE 5.31: Correction "texted" in character

52	37	56	52	37	36	0

TABLE 5.32: Correction "texted" vector

To better represent characters, we chose the convolution model and its structure is showed in the table 5.33.

character embedding
convolution filter_length=3
Dropout(0.5)
TimeDistributed(Dense(128)), activation='relu'
Dropout(0.5)
TimeDistributed(Dense(70)), activation='softmax'
loss = 'categorical_crossentropy', optimizer = 'adagrad'

TABLE 5.33: CNN structure for corrector

We did 10-folds cross validation with all tokens in 2577 tweets dataset and we got the results in the table 5.34.

# Epochs	accuracy
10	0.6377
20	0.6507
50	0.6553
100	0.6183

TABLE 5.34: First results of Corrector using CNN model

5.4.1 Context-free corrector

First we consider of a context-free corrector. This corrector will just study the form of the token and try to propose a correction without its context. The problem of this corrector is always ambiguous tokens. Sometimes the ambiguity comes from the token, sometimes because of the annotation error as showed in table 5.35.

form	correction	sentence
2	to	thanx 2 u I will b gettin better
2	too	u got 2 much going 4 u
qonna	gonna	2nites qonna be qreat
qonna	qonna	im qonna chanqe my identity

TABLE 5.35: Ambiguous token examples

In the last line, qonna is not corrected so it is considered as a SW.

The context-free method ignored the context of one token and if we train a model to correct NSW, we want our model to learn non-ambiguous words, so we kept only the correction the most frequent. For example, in the dataset, we have tokens, their corrections and the number of occurrence of the (token, correction) pair, as showed in the left of the table 5.36. That means, for the token "2", there are 92 times that "2" is corrected to "to" and 10 times "2" is corrected to "too". As for "qonna", it should be corrected as "gonna" (here we kept this oral expression instead of correcting to "going to"), but this happens only one time, the other five times, this token kept its form "qonna".

When we keep only the correction (or its original form) the most frequent, it remains as the middle of the table 5.36, the token and its correction (for the SW the correction is itself). The right column showed pairs of token and its correction, that is our training data.

Similarly, if we consider only NSW, and keep the correction the most frequent, in the final training data, we will have "u" rewriting as "you", "2" rewriting as "to" and "qonna" rewriting as "gonna".

The table 5.37 showed the neural network structure for training NSW.

u	you	203	u	you	203	u	you
2	to	92	2	to	92	2	to
2	too	10					
qonna	qonna	5	qonna	qonna	5	qonna	qonna
qonna	gonna	1					
ur	your	21	ur	your	21	ur	your
ur	your're	4	...	...			
...	...	...	...	...			

TABLE 5.36: Word list: corrector for all tokens

Character embedding layer (input: 45) output dim: 50, Xavier init
Convolution layer
Batch Normalization layer
Dropout(0.5)
Hidden Layer Dense (dim = 256, activation = relu, Xavier init)
Dropout(0.5)
Hidden Layer Dense (dim = 128, activation = relu, Xavier init)
Output layer (dim = 30, activation = softmax), categorical_crossentropy

TABLE 5.37: CNN structure for training NSW

The input has 45 possible characters (including punctuations and numbers) for each position, but for output we have only 30 possible characters (26 alphabetic characters plus ', -, _ and blank).

The evaluation is calculated by correct proposition divided by total number of tokens.

We executed a 5-folds cross validation with dataset in (Li and Liu, 2014). First we trained a model with all tokens (SW included) and we got an accuracy as 65% with all tokens and only 12% for NSW tokens.

Since the output is the most probable character for each position, sometimes we cannot get the correct form immediately. If we want to correct a NSW with a in-vocabulary (IV) word, we should find a way to match the proposition correction to a vocabulary set.

Then we trained another model with only NSW and tested on NSW and we got 25% as accuracy and this score reaches to 54% with a post-treatment even though sometimes the post-treatment can bring other errors.

Here we first used edit distance to find the most similar IV word in all the SW words in training data. Table 5.38 showed some succeeded examples.

We can see that the CNN model succeeded to find that there should be a "g" at the end of the token "shoppin" but the model was not sure about the position of the second "p", instead it predicted a blank. The second example showed that the CNN model has difficulties to add some character but it can

token	prediction	dict map	true
shoppin	shop ing	shopping	shopping
2gether	toether	together	together
l8ter	l ter	later	later

TABLE 5.38: Succeeded examples using edit distance

be corrected by matching to vocabulary as post-treatment. Similarly with the third example, the CNN model knows that the "8" should not be there (we gave only characters as possible character for output), but with edit distance matching with vocabulary, we can get its correction "later".

Table 5.39 showed failed examples with this post-treatment. The first line of the results is about predictions errors not correcting by post-treatment. We can see that even the token "holla" is in the vocabulary, it is not the correct form that we try to find. As for the prediction "moo", the word "too" in the vocabulary is more similar to the prediction that the correct form "more" because "moo" and "more" have distance as 2 and only 1 between "moo" and "too". The seconde line showed that the sometimes the prediction is correct but it is corrected by another word in the vocabulary. This is due to vocabulary, if we had a larger vocabulary than only SW in training data, this examples could be corrected. But the last token "hourse", that is difficult to predict the correction without the context because both "house" and "horse" are in vocabulary.

	token	prediction	dict map	true
wrong prediction	brekkers	brekkers	preggers	breakfast
	holla	holla	holla	holler
	moooooooo	moo	too	more
correct prediction	lamberts	lambert's	robert's	lambert's
	typin	typing	taking	typing
	hourse	hou se	house	horse

TABLE 5.39: Failed examples using edit distance

5.4.2 Corrector with context

In this section, we want to add context to the current token, that is two words before and two words after. The context is represented by word embedding as in previous section for NSW/SW classification. The table 5.40 showed the size of each vocabulary.

Figure 5.3 showed the convolutional neural network for this experiment.

data name	# vocab
training words	11,712
Google word2vec model	3,000,000
Synthesio word2vec model	2,086,985
Aspell	100,904
intersection(synthesio word2vec, Aspell)	76,318

TABLE 5.40: Vocabulary sizes

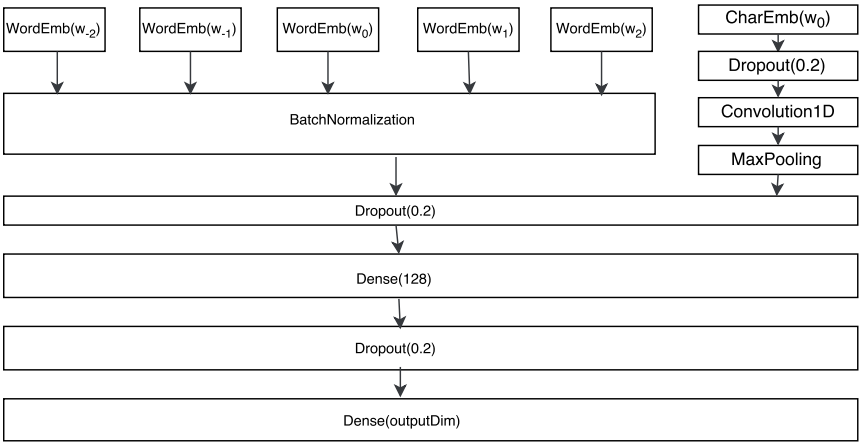

FIGURE 5.3: CNN structure for corrector with context

First we trained a model with all SW and NSW and then tested with all SW and NSW. The results are showed in the table 5.41.

data	word2vec	# vocab	1 epoch	accuracy
training words	Google	11712	180s	0.644
training words	Synthesio	11712	180s	0.7342
intersection of synthesio and Aspell	Synthesio	76318	890s	0.7235

TABLE 5.41: Corrector with context results on all words

Table 5.42 showed times needed for different vocabulary.

data	# vocab	1 epoch	100 epochs
training words	11712	180s	5h
intersection of synthesio and Aspell	76318	890s	26h
all alphabet words from synthesio	1057676	12000s	2w
all words from synthesio	2000000	24000 s	4w

TABLE 5.42: Time needed for each vocabulary

Then we tried to only train on NSW words and evaluated with only NSW words in test. That means, we supposed that all the NSW in test data are already correctly classified. The results are showed in table 5.43.

data	word2vec	# vocab	1 epoch	accuracy
training words	Google	11712	13s	0.6310
training words	Synthesio	11712	13s	0.7014
intersection of synthesio and Aspell	Synthesio	76318	65s	0.5264

TABLE 5.43: Training and testing with only NSW

Then we tried to train a neural network model with an index of a in vocabulary word. In the case of OOV as prediction, the output will be a special index for OOV.

This time we followed Leeman-Munk, Lester, and Cox, 2015. We first trained a neural network with character embedding as input and the index of the word in the vocabulary as output. This training concerns all in vocabulary words, so that depends on the vocabulary size. The objective is to create a connection between characters sequence (spelling) and the index of the word in the vocabulary. Then we saved the character embedding weights in the trained model.

Instead of using random character embeddings by Keras, we used this trained character embedding (we call it ch2vec) to represent each character. We also tested the model with and without context.

- Input:
  - character embeddings for current word
  - with or without context word embedding
- output: word index from a vocabulary or a unknown label
- vocabulary: training data (as it or all in lowercase)
- training data: only NSW or NSW+SW
- test data: only NSW (evaluation by accuracy)

Figure 5.4 showed the character embedding convolutional Neural Network structure, without context.

Figure 5.5 showed the character embedding convolutional neural network with 5 grams context structure.

We tested different configuration of CNN model, these models are all using character embedding (pretrained or not) as input and the word index in the vocabulary as output.

ch2ind: using context-free character embedding (uppercase and lowercase), trained only by NSW

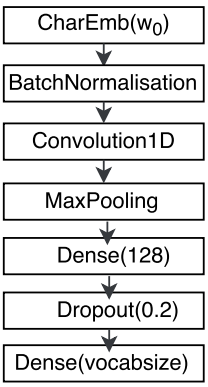

FIGURE 5.4: Context-free CNN model structure with ch2vec

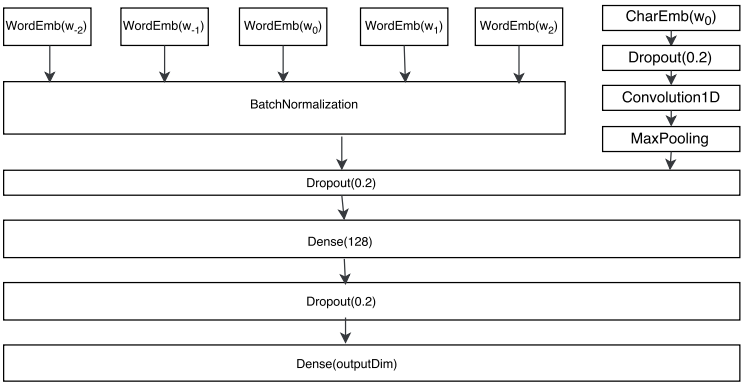

FIGURE 5.5: CNN 5-grams model structure with ch2vec

ch2indl: using context-free character embedding, all training data changed into lowercase, trained only by NSW
ch2indA: using context-free character (uppercase and lowercase) embedding, trained by NSW and SW
chE5gr: character embedding (uppercase and lowercase) with 5 grams context, trained only by NSW
chEw5gr: pretrained character embedding (uppercase and lowercase), with 5 grams context, trained only by NSW

Table 5.44 showed for each case (OOV or IV) by numbers of results (right or wrong) of each model by token category. The first line showed that an IV word should be corrected as another IV word (for model that trained by NSW and SW, this correction could be the token itself) and the model predicted an IV word as its correction (could be the token itself). The numbers of each model showed numbers of cases that the prediction is right (the first line) and wrong (the seconde line).

Since our model tries to predict a word in the vocabulary, it cannot predict how to correct a word into an OOV word. The model can only suggest a word

forme	corr	pred	r or w	ch2ind	ch2indl	ch2indA	chE5gr	chEw5gr
IV	IV	IV	right	969	860	876	956	952
IV	IV	IV	wrong	58	59	154	70	67
IV	IV	OOV	all wrong	7	8	4	8	15
IV	OOV	IV	all wrong	15	7	27	16	16
IV	OOV	OOV	as right	138	116	126	137	137
OOV	IV	IV	right	134	251	176	3	9
OOV	IV	IV	wrong	125	110	133	148	220
OOV	IV	OOV	all wrong	78	102	28	186	108
OOV	OOV	IV	all wrong	32	24	76	35	69
OOV	OOV	OOV	as right	89	108	45	86	52
				0.8091	0.8085	0.7398	0.7052	0.6729

TABLE 5.44: Results analysis by category by model

that already exists in our vocabulary. For OOV token as NSW, the model can predict either a word in the vocabulary, or just an unknown tag.

forme IV	correction IV	prediction IV
Yea	yeah	yeah
boi	boy	boy
BRUH	brother	brother
tho	though	though
wut	what	what
ja	just	in
thot	thought	hot
ko	to	know
nah	now	no

TABLE 5.45: Some result of formes IV in training data

Table 5.45 showed examples that the forme of NSW, its correction and the prediction of our model are all in-vocabulary words. These formes and corrections are present in the training data. The first five lines showed correct predictions and the last three lines are wrong predictions.

Table 5.46 showed examples in the same category: NSW forme, its correction and the prediction of our model are all in-vocabulary words, but the pair of NSW and it correction is not present in the training data. We can see that the CNN model can correctly find the forme in vocabulary for some NSW words (the first nine lines in the table) and there are also cases that the model fund another IV word (not the right word) to correct the NSW (the last seven lines in the table).

Table 5.47 showed the category that an IV forme as NSW, its correction is an OOV word but the model predicted an IV word. This category has only cases that the model failed to find the correction of the NSW. The first three

forme OOV	correction IV	prediction IV
hdve	have	have
tomorroww	tomorrow	tomorrow
2moro	tomorrow	tomorrow
aight	alright	alright
tlkin	talking	talking
uuu	you	you
gyu	you	you
demam	them	them
YUUH	you	you
Makinng	making	thing
goddd	god	good
ONOW	know	your
twitt	tweet	with
Moooooneeeeey	money	one
Cuase	cause	because
alon	alone	taylor

TABLE 5.46: Some result of formes IV not in training data

lines in the table showed the limit of our model. We can only treat the case one NSW to one correction. In the text, the annotation is to change "baby" into "babysit" and the next token "sit" into none. That means, to regroup two tokens "baby" and "sit" into only one token "babysit". Similarly with "I Phone" rewritten as "iphone" and "Fav it" rewritten as "favorite".

Then for token "m", it was a tokeniser error to separate the "m" from the number as its previous token "2m" to express "two millions". "diss" is a new word invented by Internet user to say "disrespect".

As for "realise", "sittin" and "freaking", there are only corrections all in uppercase present in the vocabulary so our model predicted with these tokens. A post-treatment to change all prediction into lowercase could correct these cases.

Table 5.48 showed examples that the correction of the token (IV or OOV) is OOV. That means, the model predicted the token with an OOV label. In total, 7 in-vocabulary NSW words which corrections are IV, are predicted with an OOV label; 78 out-of-vocabulary NSW words wich corrections are also IV, are predicted with an OOV label. For the first case, we can find examples from training data, as showed in the table 5.48.

The last category, that an OOV with also correction OOV are predicted with an IV word. That means, our model succeeded to find an IV word for the NSW token but its correction should be an OOV word. Some examples are showed in the table 5.49.

forme IV	corr OOV	pred IV	comment
baby	babysit	baby	baby sit -> babysit "
I	iphone	in	I Phone -> iphone "
Fav	favor	favorite	Fav it
m	millions	am	
diss	disrespect	this	I'm not gon diss you on the internet
realise	realize	REALIZE	
sittin	sitting	SITTING	
freakin	freaking	FREAKING	
cuss	curse	because	
now	nowadays	your	

TABLE 5.47: Predict an IV forme by an IV but correction is OOV

forme IV	correction IV	num=7
hw	how	
chill	child	
av	have	
OKC	okay	OKC -> okc in training data
dan	than	dan -> dan in training data
dun	done	dun -> don't in training data
dere	dare	dere -> dere in training data
forme OOV	correction IV	num=78
okee	okay	
cumin	coming	
girlfrnd	girlfriend	
rhe	her	
Absolut	absolute	

TABLE 5.48: Correction is IV but predicted with OOV

forme OOV	correction OOV	prediction IV (32)
secs	seconds	second
hwk	homework	week
busses	buses	because
pompo	pompous	people
dyed	dried	the
lyft	lift	left
transforma	transformation	brother
gayz	gays	Jesus
wrkd	worked	weekend
Fri	friday	from
FAVOUR	favor	favorite
ridin	riding	Riding
switchin	switching	watching

TABLE 5.49: OOV with OOV correction predicted as IV

5.5 Conclusion

In this section, we tried to use neural networks for lexical normalization.

First, we proposed CNN models and LSTM models for standard word (SW) and non-standard word (NSW) classification. We evaluated our model with precision, recall and F1-measure of NSW class. We have got a best F1-measure of 96% in 5-folds cross validation with the mixed dataset ((Baldwin et al., 2015) and (Li and Liu, 2014)).

Then, we tested the corrector module using Convolutional Neural Networks. The corrector is either trained by all SW and NSW tokens, or just by NSW tokens. In the first case, we consider that all tokens should have a correct form, even for SW, that is the token itself, and we evaluated our model by accuracy of all SW and NSW tokens. The best score we have got is about 60%. In the second case, we suppose that all NSW are correctly classified (by the previous classification). The corrector is only trained by NSW tokens and tested with only NSW tokens. Our neural network model will output an index of an in-vocabulary word or an label to mark that the correct form is an OOV word. The best accuracy of NSW prediction is 80.91%. We consider that the label OOV word is a correct prediction but in fact, our model did not propose candidate for the NSW, it cannot find an OOV word to replace the NSW, but it is considered as a correct case (explained in table 5.44).

Our objective was to use lexical normalization to improve our Named Entity Recognition (NER) results, but not to correct all NSW words. We want to add two features in our CRF model: NSW or SW class of the token and the most probable candidate for a NSW or the token itself for a SW. If we have an OOV word as correction of a NSW, that will add new values in the feature and it may not be so useful. We will just label the token that its correction is OOV word. That is why we did not continue to improve our corrector model.

Chapter 6

Using normalized text as CRFs features for NER

In chapter 3, we presented our first named entity recognition (NER) results using Conditional Random Fields (CRFs). Since the part-of-speech (POS) tag is important for NER tasks, even with other models like Maximum Entropy (MaxEnt) (Bender, Och, and Ney, 2003; Chieu and Ng, 2003; Curran and Clark, 2003) and Hidden Markov Model (HMM) (Florian et al., 2003; Klein et al., 2003; Mayfield, McNamee, and Piatko, 2003), we first used the POS tag of each token in the Penn Treebank (Marcus, Marcinkiewicz, and Santorini, 1993) as a feature in this CRFs model. However, a token can have more than one grammar category according to different contexts and we wanted to find the Part-Of-Speech (POS) tag of each token in its context as the feature POS tag for the token. This is the reason why we needed a Part-Of-Speech (POS) tagger to label our NER evaluation dataset of Synthesio.

Since the Synthesio dataset is created only with User-Generated Content (UGC) including social media data like forum posts and tweets, existing Part-Of-Speech (POS) taggers trained with well-formed texts perform poorly with it (Ritter et al., 2011; Owoputi et al., 2013). We thus decided to train a CRFs POS tagger on social media data (tweets from (Ritter et al., 2011)) and to use the predicted POS tag as a feature in CRFs model on named entity recognition (NER). The results are presented in chapter 4. There are improvements comparing to using all possible POS tags in Penn Treebank for each token.

We then discussed in chapter 5 that User-Generated Content (UGC) contains a lot of named entities, noisy tokens, abbreviations etc... which makes it difficult for all natural language processing (NLP) tasks including NER (Liu et al., 2013; Liu et al., 2011b). In the same chapter 5, we developed a normaliser to first classify all tokens in an UGC text and then try to correct non-standard words (NSW) with an in-vocabulary (IV) word. If the correction was an out-of-vocabulary (OOV) word, we only tagged the token with

an "unknown" label.

In this chapter, we first apply our classifier to the Synthesio dataset and our Named Entity Recognition (NER) training data from (Ritter et al., 2011) to see if our model can automatically detect non-standard words (NSW). After that, we test our CRFs NER model on the Synthesio dataset using this NSW/SW classification results as features. We then apply two combined context-free normalization models: normalization based on detected NSW by classification and normalization for all NSW and standard words (SW). This normalization will return the original token for standard words (SW). Finally, we try to use the normalized tokens as a feature of our CRF model in chapter 3 to improve the results of the named entity recognition (NER) experiment on the Synthesio dataset.

Contents

5.1 Annotated datasets for the lexical normalization of tweets 120
5.1.1 Dataset from (Li and Liu, 2014) 120
5.1.2 Dataset from workshop ACL2015 120
5.1.3 Typology Analysis of labeled corpora 121
5.2 Use SVM for NSW and SW classification 121
5.3 Experiments of SW and NSW classification with neural networks 124
5.3.1 Context-free experiment 125
5.3.2 5-grams experiments 126
5.3.3 Experiments with pre-trained word2vec models . . . 132
5.3.4 Experiment with optimizers 135
5.4 Word corrector 138
5.4.1 Context-free corrector 140
5.4.2 Corrector with context 142
5.5 Conclusion 149

6.1 Using NSW/SW classification prediction as CRFs features for NER

In chapter 5, we developed a NSW/SW classifier with a convolutional neural network model and we obtained 0.9614 as F1-measure of NSW class with the

mixed dataset from (Li and Liu, 2014) and (Baldwin et al., 2015). In this section, we first try to use this classifier to detect all non-standard words (NSW) on the Synthesio dataset and the dataset from (Ritter et al., 2011). Since these two datasets are not annotated with non-standard words and their corrections, we cannot evaluate our classifier but we still try to get a percentage of detected NSW on each dataset. We then try to employ this prediction of NSW/SW as a feature in our CRFs model for named entity recognition (NER). So both the training data (dataset from (Ritter et al., 2011)) and the Synthesio dataset are predicted with the same NSW/SW classifier and then we train our CRFs model with this binary feature value: 0 for NSW and 1 for SW and we present the obtained NER results.

6.1.1 NSW/SW classification experiments on the Synthesio dataset

Our contextual NSW/SW classifier from chapter 5 was developed with convolutional neural networks. The input of this CNN model was 5 grams word embedding as context and the word2vec model trained by the Synthesio data (all 3 domains), plus the character representation. The input layer then passes through the convolutional layer and another hidden layer, with the activation function "ReLU". The output is 2 dimensions, one for NSW class and the other for SW class. The activation function is softmax and the chosen optimizer was adagrad, which gave the best F1-measure of NSW class.

To get the maximum training data, we trained this CNN model with the mix of the datasets (Liu, Weng, and Jiang, 2012) and (Baldwin et al., 2015) with 100 epochs and we predicted all tokens in our evaluation dataset with the Synthesio dataset.

item	number of tokens	percentage
total tokens	19104	100%
predicted NSW	265	1.39%
predicted SW	19104	98.61%

TABLE 6.1: NSW/SW Classification on the Synthesio Dataset

We can see in table 6.1 that the number of predicted NSW is only about 1.3% of all tokens in the Synthesio dataset. Table 6.2 showed the numbers of predicted NSW in each domain and each type of texts from these corpora: long texts (like forum posts) and short texts (tweets). However, the percentage of NSW in training data ((Liu, Weng, and Jiang, 2012) and (Baldwin et al., 2015)) is between 9% and 10%.

text type	domain	NSW	total tokens	percentage
long text	Deezer	22	2314	0.95%
	DunkinDonuts	45	3116	1.44%
	LandRover	62	3836	1.62%
	Mattel	37	2958	1.25%
	Nissan	29	2166	1.34%
short text	Deezer	14	854	1.64%
	DunkinDonuts	14	827	1.69%
	LandRover	18	1123	1.60%
	Mattel	10	1048	0.95%
	Nissan	14	1127	1.24%

TABLE 6.2: NSW/SW Classification by domain and text type

First of all, since the training model considered the context of the current token, one token could be classified as NSW in some context and as SW in another context.

We then realised that the Synthesio tokenizer was different from this of the training data. In dataset (Liu, Weng, and Jiang, 2012) and (Baldwin et al., 2015), tokens with apostrophes like "I'm", "it's" and "can't" are considered as only one token. But the Synthesio tokenizer separated them into "I" and "'m", "it" and "'s" and "ca" and "n't". On one hand, it was demanded by the sentiment analyser in Synthesio since the negative expression is important to define the sentiment of a text. The negative marker like "n't" should be kept apart to calculate the sentiment of the text. On the other hand, the tokenizer of the NER dataset in (Ritter et al., 2011) also separated these tokens with apostrophe. It was also demanded for the POS tagger because on these tokens, there are subjects (pronoun) and verbs or verbs plus adverbs (not). It would be difficult for the POS tagger to predict only one grammar category for these tokens.

That is why there are tokens that begin with apostrophes like "'m" and "'s" which are recognised as non-standard words with the CNN model trained by datasets (Liu, Weng, and Jiang, 2012) and (Baldwin et al., 2015).

Table 6.3 shows some examples with the token "ca". We can see that when the token "ca" comes with the context "I can't...", it is labelled as a non-standard word. When the token "ca" begins a sentence (the case that we ignore the subject "we"), it is labelled as standard word.

When we analyse the training data, the token "ca" in dataset (Baldwin et al., 2015) does not exist and there is only one occurence of the "ca" token in dataset (Liu, Weng, and Jiang, 2012).

Figure 6.1 shows this sentence with the token "ca" in the training dataset

cases where "ca" labelled as NSW	Seems I **ca** n't sign into the website
	I **ca** n't think of an easier customizing task
	I **ca** n't stand to see food commercials on tv
	I **ca** n't reduce it,
cases where "ca" labelled as SW	Livermore, **CA** 94550, United States
	(participating US + **CA** stores)
	ca n't wait to see the Casesar Romero Joker!
	these Luxus **ca** n't all be wrong
	ca n't wait to get a car
	Ca n't explain how much I hate this shitty Nissan micra

TABLE 6.3: Classification cases with the token "ca"

Just hurt em at the icehouse comedy club in pasadena **ca**. Who's next?

FIGURE 6.1: Token "ca" in training data

(Liu, Weng, and Jiang, 2012). In this sentence, "ca" is replaced by "california" (a state in the United States of America) in the annotated dataset (the token "em" is also a NSW, it should be corrected as "them"). For the annotator's convenience, all corrections are in lowercase. With only one annotated example in the training data, the CNN model succeeded to classify cases that "ca" is standard words in these contexts (as in the last four lines in table 6.3).

We can choose here to correct the token "CA" or not depending on the degree of language that we want. Americans often use two letters as abbreviations of a state name or a city name like "NY" for "New York", "US" for "the United States (of America)".

Thirdly, according to the definition of non-standard word, the only tokens for which we can find a correct form is a non-standard words. Noisy tokens (like "PPOF"), tokens with numbers (like "$4bn") and the three protected categories: user name (begin with @), hash tag (begin with #) and urls, are all labelled as standard words because we cannot find a "correct" form for these tokens. Another category of standard word is the named entity with proper noun, as in 6.2.

I already tripped **Knox** I think, the device is n't under warranty anayway. Sent from my **C6903** using **Tapatalk**.

FIGURE 6.2: Example of named entities as standard words

"Knox" and "C6903" are product names of Samsung (mobile phones) and "Tapatalk" is a a community platform application for mobiles. Typically,

"Sent from my C6903 using Tapatalk" is a meta-message automatically generated by a mobile device. This sentence should not appear as part of the text to analyse.

text with non-standard words	correction
spotify premium is honestly a blessing in my life **ngl**	not gonna lie
All I can say is it is party time **lol**	laugh out and loud
please add it **asap** !!	as soon as possible
Plz choose me it would mean the world to me if I met them xx	please

TABLE 6.4: Example of a predicted NSW in Deezer short text

Table 6.4 shows some correctly predicted NSW. The token "ngl" (for "not gonna lie") is not present in training data, but with the context the character convolutional neural network model succeeded to predict it as a non-standard word.

The Synthesio dataset contains a lot of noisy tokens.They are not classified as non-standard words because we cannot find a correct form to replace them, or we do not know what the user wants to express by these words. Figure 6.3 shows an example of noisy tokens. In this text, "1DHQ" and "x" are noisy tokens since we cannot define what that means.

RT @onedirection: This weekend we'll be mostly listening to this @Spotify playlist.. 1DHQ x `http://t.co/G4hSeRiIcw`

FIGURE 6.3: Example of noise in Deezer short text

Named entities should not be labelled as non-standard words even though sometimes their forms could be strange. In figure 6.4, we believe that the token "5sos" is short for "5 Seconds of Summer", an Australian pop rock band, so it is a named entity and our model predicted it as a standard word. But on reading the details of this band [1], we found that this it is composed of four male musiciens. It is not so clear what "she" stands for in this context.

5sos she looks so perfect! **Plz** choose me it would mean the world to me if I met them xx

FIGURE 6.4: Example of named entity and a predicted NSW

The last problem is that in the Synthesio dataset, there are a lot of sentences which do not have meaning.

Figure 6.5 shows the complete content of a mention from the automobile domain. The second part of the text (from "issue Current") only listed serial

[1] `https://en.wikipedia.org/wiki/5_Seconds_of_Summer`

numbers of car models and motor models, the sentence "2010 BMW E90 LCI 320d Exclusive A/T-BMW Power pack 2006 BMW E90 320d Sport-pack A/T-de-dpf, 63mm down-pipe & ODR tuned 2002 BMW E39 525i A/T-sw and centre exhaust box 2011 LANDROVER DISCOVERY 4 A/T 2009 HONDA ACCORD 2.4 A/T Exclusive" is only a sequence of key-words for a car model and this sentence does not have any meaning and it can not be parsed automatically by a parser. However, in the lexical level, there is not any NSW token that we can correct.

Experienced the exact same issue, battery and alternator was changed, but the issue still persisted. problem was traced to be a software isse, the car was storing old error codes, a full software update from the agents fixed the issue Current: 2010 BMW E90 LCI 320d Exclusive A/T-BMW Power pack 2006 BMW E90 320d Sport-pack A/T-de-dpf, 63mm down-pipe & ODR tuned 2002 BMW E39 525i A/T-sw and centre exhaust box 2011 LANDROVER DISCOVERY 4 A/T 2009 HONDA ACCORD 2.4 A/T Exclusive [img][/img]

FIGURE 6.5: Example of a tweet in automobile demain

We can see from these examples that the NSW/SW classification model is not so efficient on the Synthesio dataset. Noises which can not be corrected, named entities, and texts with no syntactic meanings made it difficult to determine non-standard words to correct.

6.1.2 NSW/SW classification experiments on dataset from (Ritter et al., 2011)

We wanted to use the NSW/SW classification results as features in our CRFs model to try to improve the NER results on the Synthesio dataset. To train a CRFs model for NER, we need this information on our training data, that is the dataset from (Ritter et al., 2011).

We use the same NSW/SW classifier to predict dataset from (Ritter et al., 2011). Table 6.5 shows the result of this prediction. Our model found 2.94% NSW in this dataset. In datasets from (Baldwin et al., 2015), the percentage of NSW is 15% and in (Li and Liu, 2015), the percentage is 11%. This low number of predicted non-standard words (NSW) means that maybe our NSW/SW classifier is not so efficient for the (Ritter et al., 2011) dataset.

On the other hand, our classifier model has found non-standard words as shown on figure 6.6 in blue. Tokens in red are NSW that are missed by our classifier.

item	number of tokens	percentage
total tokens	46469	100%
predicted NSW	1365	2.94%
predicted SW	45104	97.06%

TABLE 6.5: NSW/SW Classification on (Ritter et al., 2011)

@Suzie55 whispering cause I may have had 1 too many vodka's last night and am a lil fragile

every year your "bestfriend" get yo ass in trouble smdh #doinme

@MrzEndy tru tru I'm leavin again on Tuesday yo

Bonfire tonite. All are welcome, joe included

Yeah I cant come to the meeting tomorrow.

This is the 2nd hospital ive been in today.

FIGURE 6.6: Example of succeeded NSW predictions

The case that "bestfriend" should be written as two words "best friend". In this case, we should either add a empty token in the tokenized text right after the token "bestfriend" then the correction of the token "bestfriend" would be "best" and the empty token that follows "bestfriend" should be corrected as "friend", or we should propose a blank between "best" and "friend".

We also found a tweet in Dutch in the dataset of (Ritter et al., 2011) as shown in figure 6.7. Only the token "nu" is detected as a non-standard word but the entire tweet should be considered as noise because the goal was to correct ill-formed English words and not to translate another language to English.

Kreeg net een bruikbare tip van iemand die vorige week was begonnen met een whiskydieet, hij was nu al 3 dagen kwijt

FIGURE 6.7: Example of Dutch tweet in dataset

Figure 6.8 shows other examples of missed non-standard words prediction.

Cant wait for the ravens game tomorrow....

@stjosephs Empls of the Month

FIGURE 6.8: Example of missed NSW predictions

We can see from figure 6.6 that the token "cant" is correctly detected as NSW in context "Yeah I cant come to"... but in figure 6.8 "Cant wait for"... the token "cant" was not detected as NSW.

The figure 6.9 shows a tweet which contains a lot of non-standard words. Some of them are correctly detected (in blue) and some of them are missed (in red).

Its stupid I hate getn a attitude from ppl when I'm jus tryna be nice n shit. WTF now I'm mad so dnt talk to me right now

FIGURE 6.9: Example of missed NSW predictions

Similarly to "Cant" in the beginning of the sentence, "Its" was not detected as NSW. This is also an ambiguous token, because "its" is a correct English word and it is just not correct in this context. It should be "It's" as a construction of subject plus verb but not as "its", the processive pronoun.

Our model has also detected named entity as NSW. In figure 6.10, "Scooter Braun" is a named entity of type person name in dataset (Ritter et al., 2011).

When your mom makes you go live with your dad Scooter Braun

FIGURE 6.10: Example of noises

The dataset also contains raw tokens that are not possibly to correct.

@Phoebe1_ and i also loved the last years eurovision entry!! hoppaa! they were cool too!!

FIGURE 6.11: Example of noises

In figure 6.11, "hoppaa" is detected as a non-standard words, but even for a human being, it is difficult to find a correct form to replace it, as it is just an interjection to express some feelings.

We also try our model on a tweet full of spelling errors in the dataset from (Ritter et al., 2011). This original tweet is showed in the figure 6.12.

We can see that there are only seven correct tokens with in total 32 tokens in this tweet. Our model based on context has a lot of difficulties to find all non-standard words in it. Our model only detected "lil", "whn", "gurl", "neva", "nt", and "trin" as non-standard words, others are all labelled as standard words.

We have reached a similar conclusion on the NSW/SW classification model: that it is not so efficient on the dataset (Ritter et al., 2011). Noises, named entities made it difficult to determine non-standard words to correct. It would be interesting to test another context-free classifier model.

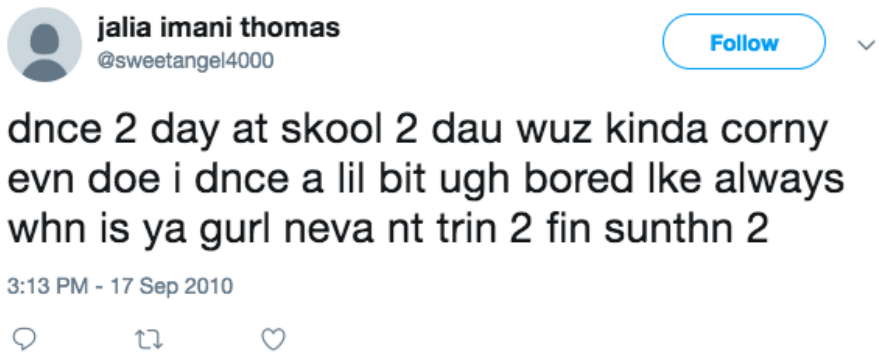

FIGURE 6.12: Tweet full of spelling errors

6.1.3 Using NSW/SW prediction as features to NER model

We tried to improve our Named Entity Recognition results using CRFs models by adding the prediction of NSW/SW information for each token as a feature.

Since the CRFs model will automatically choose features that count for the best annotation sequence, we only need to add correction proposals for each token as a feature and to let our CRFs model choose if this feature counts and how much this feature counts.

We chose the best CRFs patterns from chapter 3, that is the Constant1 model. And for the added feature, the corrections of all tokens, we used the token itself in lowercase for SW and corrections are also in lowercase as the annotation convention.

Each CRFs feature can be defined on all tokens sequences. We used the same window size as other features, that is two words before and two words after, plus the current token. Similarly, we let our CRFs model choose which is important as a feature's position.

The CRFs model is then evaluated by named entity type and with precision, recall and F1-measure as previously done in the chapter 3.

This feature of NSW/SW classification value is represented by either 0 (for NSW) or 1 (for SW). To use this feature as maximum, we also include this feature for the context of the current token. That is, two words before and two words after the current token.

Other features and window sizes of the pattern remain the same as 4.20 in chapter 3.

We first tried this model with a cross validation with 5 blocks using dataset form (Ritter et al., 2011) with their definition of named entity types.

We then trained our model on dataset from (Ritter et al., 2011) and tested with the Synthesio dataset with the Synthesio definition of named entity types.

	Precision	Recall	F-Measure
Company	1.0/1.0	0.0302/0.0369	0.0586/0.0712
Person	0.35/0.35	0.41/0.41	0.38/0.38
Other	0.5/0.5	1.0/1.0	0.67/0.67
Product	0.5/0.5	0.03/0.05	0.06/0.09
Media	0.5/0.5	0.0625/0.0625	0.11/0.11
Geo-Location	0/0	0/0	0/0
Job title	0/0	0/0	0/0
Micro-Average	0.34/0.34	0.1/0.1	0.12/0.1634

TABLE 6.6: NER results using NSW/SW classification feature

Table 6.6 shows the result on the Synthesio dataset. The only increased score is the recall of named entity `Company`, from 0.0302 to 0.0369 and the micro-average of f1-measure on this dataset has increased from 0.12 to 0.1634. That means, the CRFs model took account this NSW/SW classification results as feature, and this feature has increased the recall of one type of named entity.

The increase of the recall of the named entity `Company` is the result of four occurrences of three tokens, missing from earlier model without NSW/SW classification as features. The four tokens as named entity `Company` are: "BBC", two occurrences of "DD" and "DC".

when I tried windows phone Sky sports TV and sky go Bank apps BBC apps YouTube Spotify

FIGURE 6.13: Example of "BBC", predicted as NSW and detected as Company

In the text of the figure 6.13, the annotated named entities are:

windows	product
BBC	company
Youtube	company
Spotify	product

TABLE 6.7: Annotated named entities in the text of figure 6.13

These four named entities in the example of the figure 6.13 were all missed in earlier CRFs models without NSW/SW classification predictions as features. However, the NSW/SW classifier predicted "BBC" as the only non-standard word in this text and with this feature value, our CRFs model succeeded to extract "BBC" as a company name.

No DD in Oregon either.

FIGURE 6.14: Token "DD" detected as Company

In the text of the figure 6.14, "DD", which refers to "Dunkin' Donuts" (an American fast food company), was detected as a non-standard word. Our CRFs model then predicted it as a company name. Meanwhile, the token "Oregon", an American state, was missed as a named entity of `GEO-LOCATION`.

I ate a dozen blueberry munchkins from DD this morning and they were amazeballs!

FIGURE 6.15: Another occurrence of "DD"

The figure 6.15 showed another occurrence of the token "DD", also predicted as non-standard word and then detected as a named entity of the type `Company`. Meanwhile, our CRFs model did not succeed to extract the product "blueberry munchkins" from this text.

For that and a ton of DC properties, and MOTU, & Cancer, & Diabetes I too blame Mattel.

FIGURE 6.16: Example of "DC"

In the text of figure 6.16, the token "DC" was detected as a NSW because it is the abbreviation for "DC Comics". Then our CRFs model succeeded to label this token as a name of a `Company`. However, "Mattel", failed to be detected as `Company` again.

These four cases showed that the feature of non-standard word, along with the feature "contains only letters in uppercase" helped to find tokens with only letters in uppercase as named entities such as "DC", "BBC" and "DD".

In this section, we tried to use NSW/SW classification prediction as a feature of our CRFs model for named entity recognition (NER). As shown in table 6.6, only the recall of type `Company` increased with four succeeded tokens. Although our NSW/SW classifier has found less NSW in both datasets from (Ritter et al., 2011) and from the Synthesio dataset, this feature helped to improve our CRFs model for NER.

6.2 Using normalized text as feature for NER

In the previous section, we have discussed the results of non-standard words (NSW) and standard words (SW) classification on the Synthesio dataset and the dataset from (Ritter et al., 2011): the neural network classifier predicted 1.39% and 2.94% NSW of all tokens in the two datasets respectively.

In this section, we first use a normalization model (from chapter 5) to propose a correct form for detected NSWs by the classifier. This normalization candidate will be used as a feature for the CRFs model for named entity recognition (NER).

Secondly, we directly employ the second best normalization model, another context-free model, which was trained on both NSWs and SWs. The normalization model proposes for each token in the dataset, a form as correction. As for original SW tokens, the proposed corrections are supposed to be their original forms in the dataset. We then use this correction as a feature in CRFs model for NER.

6.2.1 Word normalization based on NSW/SW classification prediction

The normalization model based on NSW/SW classification prediction results proposes one candidate for each detected NSW by the classifier. In chapter 5, we have developed normalization models and we evaluated their performances by eight categories. The eight categories were about original form in the text, annotated canonical form and normalization candidate by normalization model, each of the three categories can be in-vocabulary (IV) word or out-of-vocabulary (OOV) word as presented in 5.44.

In this section we want to use the best normalization model obtained, which has the best accuracy of all NSWs by cross validation on 5 blocks on the mixed datasets of (Baldwin et al., 2015) and (Li and Liu, 2014).

All normalization models of chapter 5 proposes lowercase characters because of the annotator's convention from the two datasets. The best normalization model, with the highest accuracy score, was "ch2ind" from 5.44. As output, this model will propose an in-vocabulary (IV) word, the vocabulary being the interjection of the Aspell dictionary and the Synthesio vocabulary, or an "unknown" label for a candidate that is not in the vocabulary.

As explained in chapter 5, when this normalization model proposed an out-of-vocabulary word as correction for a NSW and the annotation of the

NSW was also an OOV word, we considered as if the normalization model proposed the correct normalized form for accuracy evaluation of all NSWs. Therefore, instead of choosing the model with the best accuracy, we chose the model with the most correct normalized forms and the least wrong normalized forms, the model "ch2indL" as shown in table 5.44 of chapter 5. That means, we chose the normalization model which has the best accuracy score without considering annotations when they are OOV words.

We first applied the same classifier model from the previous section on dataset from (Ritter et al., 2011) (for NER training) and then on the Synthesio dataset (for NER prediction) as we did in the previous section.

We then trained this normalization model, "ch2indL" from 5.44, on all annotated NSWs in datasets of (Baldwin et al., 2015) and (Li and Liu, 2014). "ch2indL" is a neural network which takes the current words and its context (two words before and two words after) as input, and its output is an id in the vocabulary (the vocabulary is the interjection of the Aspell dictionary and the Synthesio vocabulary) or a "unknown" label if the proposed candidate is not in the vocabulary.

We then applied the normalization model on predicted NSW for both training dataset from (Ritter et al., 2011) and the Synthesio dataset (the evaluation dataset). After that, we added normalized form for each token as CRFs feature value in the CRFs Named Entity Recognition (NER) model.

Normalization model on dataset from (Ritter et al., 2011)

The classifier found 2.94% NSW and we applied the normalization model on these NSWs.

Table 6.8 shows examples on dataset from (Ritter et al., 2011) where the normalization predictions were correct on detected NSW in this dataset.

Detected NSW	Proposed correction
and am a lil fragile	little
get yo ass in trouble	your
I'm leavin again on Tuesday	leaving
Whats goin on (in tally) tonight twitter!!	going
jus about to get dressed and leave again	just
enjoy it. like u just got married	you
I did tweet him b4 n justthen about kenny	before, and

TABLE 6.8: Succeeded normalization examples on (Ritter et al., 2011)

For cases where we should add letter(s) to the end of the token to correct a NSW, the classifier has succeeded to find "yo", "jus", "leavin" and "goin" as NSW, then the normalization model succeeded to find corrections "your" for "yo", "just" for "jus", "leaving" for "leavin" and "going" for "goin".

Tokens with only one letter like "u", "n" are also correctly detected as NSW and then the normalization model proposed "you" and "and" as correct forms. Cases with "lil" and "b4" also show that our normalization model can propose a candidate by removing a character in the end of the token (the last "l" from "lil" and "4" from "b4") then add other letters as candidates of these NSWs.

Detected NSW	Proposed correction	Correction
How was ur day 2day?	your, today	
it is dnt have a move tonite but imma make	not, tonight,my	I'm going to
bestfriend	"unknown"	best friend
i dont have any change for the customers..	"unknown"	don't
cant believe her brother is leaving	"unknown"	can't
Lmao	ass	laughing my ass off
Imma	my	I'm going to
smdh	the	shaking my damn head

TABLE 6.9: Wrong candidates for predicted NSW

Table 6.9 shows examples of wrong candidates and NSWs with "unknown" label. Wrong predictions and missing NSW from classifier are shown in red. Tokens in blue are predicted NSW by the classifier of the previous step. In the first line, if the classifier had correctly detected the token "2day" as NSW, the normalization model would have proposed its correct form "today". We can see that our normalization model could not add a proper character in the middle of the token, as in the example of adding a blank between "best" and "friend" for "best friend". It can turn into a tokenisation problem.

As for tokens like "dont" and "cant", according to the Synthesio tokeniser, tokens "don't" and "can't" are tokenised as "do", "n't" and "ca", "n't" respectly, so tokens "don't" and "can't" do not exit in the vocabulary, so the normalization model cannot propose these forms and it only returns a label "unknown" as the candidate is an out-of-vocabulary word.

We can also see from examples of "Lmao", "Imma" and "smdh" that the normalization model can't really find multi-tokens expressions as correction. If one character represents one word, there is not enough information.

Table 6.10 shows predictions on the last part of the tweet in Figure 6.12. Tokens in red are detected as NSW in the previous step, so they are over corrected words. We are not sure if the token "whn" means "when" and this

is the end of the tweet. There is only one correct token in this sequence of tokens and we find really difficult to understand a tweet like this one. The classifier could not detect all NSW and the normalization model could have corrected "ya" by "you", "2" by "to" and not so correctly "trin" by "nothing", and "fin" by "fuck".

Token	Proposed correction	Correction
whn	when	
is		
ya		you
gurl	girl	girl
neva	never	never
nt	not	not
trin		trying
2		"to" or "too"
fin		find

TABLE 6.10: Wrong candidate for predicted NSW

Similarly, this normalization model can neither just add a blank for the token "alot" nor add an apostrophe for the token "isnt". For both cases, a label for an "unknown" word shows that the candidate of correction is not in the vocabulary.

Combined normalization model on dataset from the Synthesio dataset

We can also see in table 6.11 that the normalization model proposed a word shorter than the original token as "shit" for "shitty". That means, the normalization model replaced the last two characters by blanks (since all tokens shorter than 29 characters are padded on the right). However, the word "shitty" is just an informal and oral form for "shit". We can choose to correct it or not according to different degree of our normaliser. It is similar to the case of "gonna" for "going to" in spoken English. On one hand, we want to change non-canonical text into standard, well-formed text and we are intended to correct cases like this, we consider the spoken English as non-standard language.

Further more, "gonna" is a typical case that we can generalised as "wanna" (for "want to"). On the other hand, we consider in this thesis only cases of one token to one correction (in the form of corrected token). These cases are like to change one token "gonna" into two tokens ("going to"). That is why tokens like "gonna" are not corrected in the training data and they are tagged as standard words, as we do not want to correct them.

Similarly for the word "BBQ". It is a common informal spelling of "barbecue", and it is an out-of-vocabulary word for the Aspell English dictionary. Our classifier has detected "BBQ" as a non-standard words, but the normalization model was not able to propose an in-vocabulary candidate to correct it. Only the label "unknown" is distributed for this token.

As for named entities, we can see that the normalization model over-corrected named entities "Knightley", of type person name, "Nissan", the car brand and another commun noun "BBQ". The token "Nissan" is the only occurrence in this dataset which is detected as NSW, all other occurrences are detected as SW.

The NSW/SW classifier is a model which took account of the word's context (two words before and two word after). The token "Nissan" following the determinant "a" in the context "of a Nissan note", the POS tagger has tagged both "Nissan" and "note" as common nouns. However, the form "Nissan" should be a proper noun as a company name (all other occurrences of "Nissan" are tagged as proper nouns), that is why the classifier labelled "Nissan" as a non-standard word, and then the normalization model proposed an in-vocabulary (IV) word, "saying". If "Nissan" was labelled with "proper noun" as POS tag, maybe it would not be detected as a NSW as other occurrences of "Nissan" in other contexts. Proper nouns are often SW tokens in training data, and we do not need to normalize them.

Token	Proposed correction	Correction
Apps like Spotify can sync and control even music	"unknown"	synchronize
It is gonna take alot more than deep pockets	always	a lot
but mainstream isnt a genre a music.	shit	isn't
After speaking to someone in your call centre	center	
Wtf.	with	what the fuck
even were given em away free,	them	
I hate this shitty Nissan micra	shit	
Keira Knightley Says No To Airbrushed Boobs	"unknown"	
we are having a backyard BBQ reception.	"unknown"	
I booked a test drive of a Nissan note	saying	

TABLE 6.11: candidate for predicted NSW on the Synthesio dataset

When we applied our classifier, we had predicted 2.94% of non-standard words with the dataset from (Ritter et al., 2011) and 1.39% for the Synthesio dataset. Meanwhile, the percentage of NSW in both training datasets ((Li and Liu, 2014) and (Baldwin et al., 2015)) was between 9% and 10%. If we assume that the NSW percentage should follow this distribution, there should be a lot of NSW detected as SW. That means, the normalization model would only

propose candidate of correction for these predicted NSW and it will never be applied to others from 97% to 98% tokens. In addition, the classifier has also predicted SW as NSW as in the last three examples in table 6.11, and the normalization model intended to propose candidates for these standard words, named entities "Knightley" and "Nissan".

Using predicted candidate for NSW and SW itself as feature value for NER

The normalization candidate is then added into the CRFs model's feature value for Named Entity Recognition (NER). This feature value will be an "unknown" label for an OOV correction for a NSW detected by the classifier.

Similarly to NSW/SW class feature, we trained the CRFs model with this feature value (the candidate for detected NSW) with the context window size of 5 (two words before and two words after, including the current word). For detected SW, the original form will be the value of this feature.

We applied this combination of NSW/SW classifier and the normalization model on predicted NSW to our training data from (Ritter et al., 2011) and on the Synthesio dataset. The candidate proposed by the normalization model on predicted NSW is employed as a feature value "correction" of NSW in CRFs model. As for predicted SW, only the original form in the dataset with all letters in lowercase (according to NSW annotation convention) is considered as the feature value.

We also want to keep the context window size as 5 (two words before and two words after) to maximize numbers of feature values as we did for the part-of-speech (POS) tag feature.

However, the NER results with our CRFs model did not change and from the generated CRFs model file, we did not see the feature "correction of NSW" appear in the optimized model. The CRFs model computed all feature values with many iterations (100 iterations as previous case) and finally reached an optimized set of feature and a weight for each feature function. That means, this "correction of NSW" feature value did not help for our NER task in the final generated CRFs model because it is not so useful comparing to other features. The optimized CRFs model remains the same and so are the results of NER prediction.

This result makes us consider the other possibility to train the normalization model, that is, training the normalization model with all NSW and SW tokens with their corrections. The correction of a SW word would be its original form with all letters in lowercase (according to NSW/SW annotation convention). This usage of the normalization model ignored the step of

NSW/SW classification and the normalization model proposes directly the correction to replace a token either a NSW or a SW.

6.2.2 Normalization model for all tokens prediction

We think that a normalization model which proposes a correction for all tokens is maybe more reasonable and it will propose the original form for standard words.

The normalization model that we chose, as the best NN model from chapter 5, takes only the current token as input. The token is a list of characters and each character (numbers and symbols included) is represented by an integer number as we did previously. The largest word in all training data has 29 characters. To cover all letters for all words, we fixed the length of words as 29 characters. Words containing less than 29 characters are padded to the right with the space character " ".

The convolutional neural network contains a convolution layer which follows the input layer. There is another hidden layer of 128 nodes before the output layer.

The output layer also has 29 positions to fill with, for each position, the most probable character. Blanks are also filled to complete until 29 positions. For each position, possible characters are: all 26 English alphabet characters, plus symbols "’", "-" and blank " " (for padding).

Sometimes the list of characters in the output do not form an in-vocabulary word. The vocabulary is only generated from all the corrections from training data. A post-processing step will match the list of characters to the nearest word according to the edit distance (Levenshtein, 1966) from the vocabulary.

To get the maximum data in training, we trained this neural network model on (Li and Liu, 2014) and (Baldwin et al., 2015).

Neither the dataset from (Ritter et al., 2011) nor the Synthesio dataset is annotated with normalized words. We cannot really evaluate the normalization model with this dataset. We only got the percentage of corrected words, they are supposed to be non-standard words and we tried to analyse some example from these words which had been corrected by the normalization model.

As described earlier, our correct model on neural network proposed a candidate for each token as a correction. This candidate, is a list of possible characters including the hyphen "-" to join two words, the apostrophe "’" for the omission of one or more letters (as in the contraction of do not to don’t);

the marking of possessive case of nouns (as in the eagle's feathers, or in one month's time); and the marking of plurals of individual characters (e.g. p's and q's). The third possible character except alphabet letters is the blank " ".

The candidate is a list of possible characters and the most probable character in each position of the candidate.

As post-treatment, we first used all SW tokens plus all corrections in the training data to test rapidly our model. The real size of the total vocabulary of these tokens counted until 10 thousand, and we will test our model with this total vocabulary later.

Tables 6.12 to 6.13 showed examples of the normalization model predictions with tweets in (Ritter et al., 2011), including the candidate that the normalization model proposed, and the in-vocabulary word found in all training data.

Tokens	Proposed candidate	IV word
pretty	pretty	pretty
bad	bad	bad
storm	storm	storm
here	heree	here
last	last	last
evening	evening	evening

TABLE 6.12: Correctly predicted tweet in (Ritter et al., 2011)

In the tweet from table 6.12, the normalization model proposed a wrong candidate "heree" for the token "here" but the post-treatment found the nearest IV word "here" from the vocabulary (from all SW and corrections in training data).

The token "'s" has no syntactic meaning in this tweet, it should even not be in this context.

In the tweet from table 6.13, the correct form of the token "lil", according to the context, should be "little", but here the normalization model predicted a candidate form like "lil l" and after the post-treatment to find the nearest IV word in training data, we found "llol" as its correct form. In fact, the token "llol" exists in the training data as a noisy token, as in figure 6.17, which can not be corrected because there is not a correct form for this token and in the training data, the token has been left as it is, as a standard word. That is why the form "llol" is distributed as the nearest form in the vocabulary.

In the training data, there is the token "fraggle" as a SW and that is why "fraggle" is also in the vocabulary. Fraggle Rock (also known as Jim Henson's Fraggle Rock or Fraggle Rock with Jim Henson's Muppets) is a children's

form in text	normalization prediction	matching in vocabulary
whispering	whispering	whispering
cause	cause	cause
i	i	i
may	may	may
have	have	have
had	had	had
1	1	1
too	too	too
many	many	many
vodka	vodka	vodka
's	's	's
last	last	last
night	night	night
and	and	and
am	am	am
a	a	a
lil	lil l	llol
fragile	fraggle	fraggle

TABLE 6.13: Miss-corrected tokens in (Ritter et al., 2011)

don't act lyke (like) you don't lyke (like) that llol
if you 4got (forgot) you still have me until winter

FIGURE 6.17: Token "llol" in training data

television series, created by Jim Henson [2]. So the token "fraggle" is a part of a named entity "Fraggle Rock" of type "tv show". That is why in the training data, there is not a correct form annotated for the token "fraggle" and we generated the vocabulary with all SW and corrections.

Watching some fraggle rock & making breakfast.

FIGURE 6.18: Token "fraggle" in training data

As mentioned in chapter 5, noise tokens and named entities are considered as standard words since there is no need and no possibility to find a "correct" form for these tokens. However, we can see the problem when we use standard words in the spelling vocabulary: the named entity "Fraggle" and the noise token "llol" are found as nearest spelling form for the word "fragile" and the token "lil", we over-corrected "fragile", which already was a well-formed word and we did not find the correct form of "lil", which should be "little".

[2] https://en.wikipedia.org/wiki/Fraggle_Rock

We then considered another standard English dictionary, the Aspell dictionary [3] as vocabulary. In the Aspell dictionary, there is only one occurence for one word, but this word could be all letters in lowercase, all letters in uppercase or only first letter in uppercase and others in lowercase. These words are also sorted by standard ASCII code (or "C" locale on Unix systems). That is, first in uppercase (from A to Z) and then in lowercase (from a to z) following alphabetical order. The only punctuation character in this dictionary is the apostrophe "'".

To cover all English words, we merged the American English and British English dictionaries to reach to an American-British English dictionary and we then sorted the words list in the same order.

When our post-treatment traverses all words in a dictionary, it stops at the first word which has the defined edit distance even though there are others words with the same edit distance.

In this example, we expect that the form proposed by our corrector "fraggle" be matched with "fragile". Table 6.14 showed the same sentence predicted by the same normalization model and then matching the nearest word from Aspell dictionary. We verified that tokens "fraggle", "Fraggle" and "FRAGGLE" are not in this dictionary.

We can see that the post-treatment has corrected the prediction "fraggle" into "fragile", which is the original form of this token. Meanwhile, we noticed that the post-treatment has found another word "Gill" for the prediction "lil l" from the Aspell dictionary.

"Gill" was the first word (in alphabetical order) in vocabulary which is the nearest from the prediction "lil l", with edit distance 2. The objective word that we are looking for, "little", has edit distance 3 from "lil l".

We also could define from all words which have the same edit distance as 3 with "lil l", only the words with the same first letter as correction. If there was always more than one nearest word, we chose the word with the same two first letters as correction, etc... until the number of letters minus the edit distance (2), the result is 3 (so for the first three letters).

In this case, our post-treatment found "lilt" and "lily" in the Aspell dictionary. As "lilt" appears before "lily" in alphabetical order, the returned correction for "lil l" will be "lilt".

Using Aspell dictionary can help in some cases to find the standard form for the prediction of the normalization model as post-treatment. However,

[3] http://aspell.net/

form in text	normalization prediction	matching in vocabulary
whispering	whispering	whispering
cause	causes	causes
i	i	i
may	may	may
have	have	have
had	had	had
1	1	1
too	too	too
many	many	many
vodka	vodka	vodka
's	's	's
last	last	last
night	night	night
and	and	and
am	am	am
a	a	a
lil	lil l	Gill, lilt
fragile	fraggle	fragile

TABLE 6.14: Miss-corrected tokens in (Ritter et al., 2011) using Aspell dictionary

sometimes the nearest form is not the correction of the non-standard word (as "lilt" for "lil").

On the other hand, Traversing a list of words to find the nearest word in edit distance could take very long time according to the size of the vocabulary. The size of the Aspell dictionary is 100905 words, and the vocabulary in the training data is only 13926. Using the Aspell dictionary takes almost 10 more time than only using standard words and corrections of non-standard words in training data in case of uni-thread processing. We can surely process on multi-thread and then find a better way to choose a candidate among nearest tokens by edit distance in the vocabulary.

6.2.3 Using normalized texts as features for NER

In this section, we add the normalized texts to CRFs features for named entity recognition (NER).

Other features of the CRFs model is the same as in chapter 4.

In the previous chapter 3, the pos tagger helped to improve our NER model, in this step, we will try to get two results, with the pos tagger and without the pos tagger.

The pos tagger model that we chose is the mixed model trained by dataset (Ritter et al., 2011) and the PTB dataset. The POS tagset is the Synthesio universal tagset with 17 possible tags.

Since the CRFs model will automatically choose features that count for the best annotation sequence, we only need to add correction proposals for each token as a feature and to let our CRFs model choose if this feature counts and how much this feature counts.

We chose the best CRFs patterns from chapter 3, that is the Constant1 model. And for the added feature, the corrections of all tokens, we used the token itself in lowercase for SW and corrections are also in lowercase as the annotation convention.

Each CRFs feature can be defined on all tokens sequences. We used the same window size as other features, that is two words before and two words after, plus the current token. Similarly, we let our CRFs model choose which is important as feature's position.

The CRFs model is evaluated by named entity type and with precision, recall and F1-measure as previously done in the chapter 3.

Table 6.15 shows the results of each named entity type.

	Precision	Recall	F-Measure
Company	1.0	0.03/0.05	0.06/ 0.10
Person	0.35/0.38	0.41/0.52	0.38/0.44
Other	0.5	1.0/1.0	0.67/0.67
Product	0.5	0.03/0.04	0.06/0.07
Media	0.5	0.07	0.12
Geo-Location	0/0	0/0	0/0
Job-title	0/0	0/0	0/0
Micro-Average	0.45/0.49	0.07/0.10	0.12/0.17

TABLE 6.15: NER results using POS tagger as feature

Named entities of type `Company`, `Product` and `Person` have better recall score and for `Person`, the precision is also better than without normalized text. For `Media`, `Geo-location` and `Job-title`, the result of NER have not been improved.

However, the true positive predictions increased from 18 to 30 for named entity `Company`, from 47 to 60 for `Person` and from 16 to 21 for `Product` and for other types, the scores remain the same.

In this Synthesio dataset, there is 596 labelled named entities of the type `Company` and 534 labelled named entities of the type `Product`, Therefore, this dataset is more business-oriented.

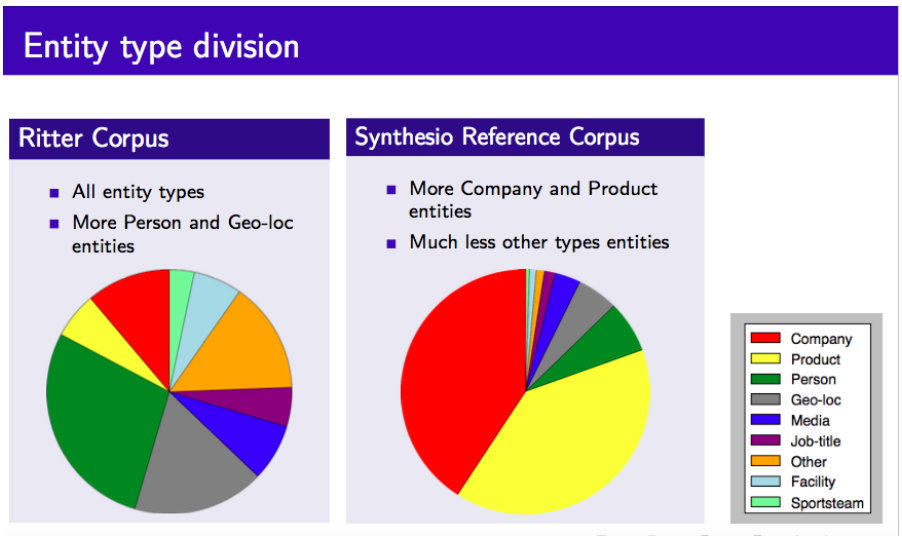

FIGURE 6.19: Named entity distribution in two datasets

The figure 6.19 shows the distribution of each type of named entities in both datasets. These two datasets are very different. The Synthesio dataset has more `Company` and `Product` named entities. These two types of named entities also have a lot of repetitions for one named entity.

named entity	occurrence	named entity (concurrent)	occurrence
Deezer	15	Spotify	28
Dunkin Donuts	13	Starbucks	37
		McDonalds	25
Mattel	87	Disney	13
Nissan	74	Toyota	8
Land Rover	13	Jeep	27
Total	340		

TABLE 6.16: Clients names and their concurrents

For example, in all labelled named entities of type `Company`, some named entities and their numbers of occurrences are shown in table 6.16. All these named entities are not in the training dataset (from (Ritter et al., 2011)), and the total number of their occurrences counts more than half of the labelled `Company` named entity (340 on 534). These named entity of type `Company` have never been in the training data from (Ritter et al., 2011), and the context from CRFs features is not sufficient to detect them.

6.3 Conclusion

In this chapter, we wanted to add normalization results as a new feature of the CRFs model from chapter 4 for Named Entity Recognition (NER) on the Synthesio dataset.

We first applied the non-standard words (NSW) and standard words (SW) classifier from chapter 5 to predict all tokens for the dataset from (Ritter et al., 2011) and for the Synthesio dataset. The percentages of detected NSW were 2.94% and 1.39% respectively.

When we used the classifier prediction as a feature (which value is either NSW or SW) in the CRFs model, the CRFs model succeeded to correctly label more named entities of type `Company`.

We then tried to normalize the text by proposing a normalized form for detected non-standard words (NSW) or for all tokens in the text.

We tested two normalization models.

The first normalization model was only trained on detected non-standard words (NSW) from the training data (from (Baldwin et al., 2015) and (Li and Liu, 2014)). This model was then only applied on NSW predicted by the NSW/SW classifier, and we added the proposed normalization candidate as feature value of the CRFs model. The detected SW have their original forms as feature value. The NER results by the CRFs model did not change because the generated CRFs model did not keep this feature in optimized set of features and weights because the CRFs model considered that this feature was not useful according to the labelled training data.

We tried then another normalization model which proposed a normalized form for all tokens. The feature value was the normalized form for NSW and the original form for SW. The NER results with the same CRFs model showed improvement for some types of named entities and the same results as before for other types.

Finally, analysis of the distribution of named entities of these two datasets showed that they are very different. That is the reason why the CRFs model trained on one dataset does not perform the same on the other dataset.

Having normalized texts as features of the CRF model allowed more named entities to be detected in the Synthesio corpus. However, some are still missing as examples shown in table 6.16.

Chapter 7

Conclusion and perspectives

This PHD thesis aimed to improve Named Entity Recognition (NER) results on User-Generated Content (UGC). In this chapter, we try to summarize our entire work and contributions regarding this theme.

This work is the result of a Cifre grant and our results are thus more practical than theoretical.

We analysed the differences between UGC and well-formed text (standard text) from a lexical, morpho-syntactic and syntactic point of view (chapter 2). We then created a multi-domain, multi-type dataset in English, and we labelled this dataset with eight types of named entities (chapter 3). The first results of the Named Entity Recognition (NER) task using Conditional Random fields (CRFs) on this dataset are presented in chapter 4.

Since the results were not satisfying enough, we tried two ways to improve them. The first method consisted in training a POS tagger on mixed data with User-Generated Content (UGC) and standard texts. This POS tagger was trained with a defined universal tagset suitable for all 80 languages that Synthesio works on. Then we used the predicted POS tags as a feature in our NER model with CRFs (chapter 4).

As mentioned in chapter 2, user-generated content (UGC) often contains spelling errors, abbreviations, etc. compared to standard text. In order to normalize a part of the UGC vocabulary in the Synthesio datatset, we first tried to unify some date, time, currency, quantity and volume expressions using regular expressions. We then created a Synthesio word2vec model after this first normalization (chapter 3).

After that, we tried to normalize non-standard words (NSW) by proposing a correct form for them. We first developed a convolutional neural network (CNN) model to classify standard words (SW) and non-standard words (NSW). We then developed a lexical normalization model that proposes a correct form to replace non-standard words (NSW) in the text. As for standard words, this proposition was itself (chapter 5). We then used the proposed

candidates as another feature in our NER model with CRFs to try to improve the NER results (chapter 6).

7.1 Summary of findings

Nowadays, more and more User-Generated Content (UGC) is present on the Internet. In this context, Natural Language Processing (NLP) tasks make it possible to analyze the mass of information with less human efforts. These NLP tools include tokenizers, part-of-speech (POS) taggers, chunkers, parsers and named entity recognizers. Many applications like sentiment analysis, social recommendations and advertising are based on these NLP tools.

This work is the result of a Cifre thesis developed within the company Synthesio. Synthesio works on social media data analysis and sentiment analysis. Its clients cover the cosmetics, automobile, music streaming domains. Synthesio provides text analysis for more than 80 languages all over the world.

Synthesio first extracts text data through the Internet that interest their clients, then tries to analyze these data in time period, geolocation, and language. Synthesio also developed a sentiment analyser with three values for each document: positive, negative and neutral.

The Named Entity Recognition (NER) task aimed to help them identify the topic of a text. The topic could be a company, a product, a person, a sports team, a music artist... according to the domains and need of their clients.

There are existing named entity recognisers trained on tweets but for Synthesio, the ressources of User-Generated Contents (UGC) are short texts like tweets (limited in 140 characters) and also long texts on forums. The definitions of entity types are also different from other named entity recognisers. That is why we should develop our own named entity recogniser.

There are already User-Generated Content (UGC) datasets in which named entities are annotated (Ritter et al., 2011) if we want to train a proper NER model on them. We only had to add a new type of named entity, the job title and merge some types of existing named entities (TV shows, movies and music artist into media) following Synthesio's named entity definitions.

Dataset creation and annotation

Since we wanted to work with real data for Synthesio, we first needed to create a dataset for evaluation.

We created a multi-domain, multi-type dataset in English using Synthesio data.

We chose five Synthesio clients: Deezer, Dunkin donuts, Land Rover, Mattel and Nissan from four different domains: music, street food, automobile and toys. For each domain, we extracted 50 long texts from discussion forums, websites and 50 short texts from Twitter. They were organized by domain and by text type (long text and short text). We tokenized the text with the Synthesio tokenizer, then we annotated these data following the recommendations of Ritter et al., 2011 and Synthesio's definitions of types of named entities. The human annotation was done with the same person that developed the NER model in CRFs. In order to test the model's performance, the tendency was to annotate the most named entity possible, that is, when we are not sure if a word sequence is a named entity (for example, "witch doctor" for named entity of type `Job-title`), we define it as a named entity.

These texts, whether they were long or short, were all User-Generated Content (UGC), so they contained a lot of noise and mistakes. Some texts were not syntactically correct, they were real pieces of data that Synthesio tried to analyse.

This dataset also contained some duplicated data since Synthesio kept all quotes in forum, all retweets from one tweet (beginning with "RT:" then following by the content of the original tweet) and these duplicated texts had different identifications.

This dataset was not a proper dataset for training, but it was a good beginning for an evaluation data.

POS tagger

The POS tagger's performance decreases on user-generated content as mentioned in (Gimpel et al., 2011) and in (Owoputi et al., 2013).

We developed a new POS tagger for UGC, with the objective to improve our NER model's performance.

This POS tagger is different from the existing ones. First, our goal is to improve NER results. We want to use the grammar category for each token as a feature. Secondly, the User-Generated Content (UGC) that we try to process contains a lot of syntax errors. Lastly, Synthesio wanted a POS tagger for all the eighty languages that they process, so the definition of tagset should be compatible to almost all human languages.

That is the reason why we do not need a detailled tagset like Penn Treebank (Marcus, Marcinkiewicz, and Santorini, 1993) (with forty-five different POS tags). So we defined a POS tagset with 17 tags, and a mapping rule to match the tagset of Penn Treebank (Marcus, Marcinkiewicz, and Santorini, 1993) and (Ritter et al., 2011) into these 17 tags.

We used a cross validation approach for validation on dataset in (Ritter et al., 2011). We found that the mixed training data with the Penn Tree Bank (Marcus, Marcinkiewicz, and Santorini, 1993) plus the training part of cross validation) works better than a model trained only on one dataset.

We can conclude that having more training data is the best option to reach the best performing model. But since we evaluate our model on tweet data in (Ritter et al., 2011), if the quantity of Penn Tree Bank data adding into training data exceeds some limit (in our experiment, that's nine times more sequences), the performance of the pos tagger will decrease.

Synthesio has already implemented this POS tagger as a component of its natural language processing (NLP) pipeline right after the tokenizer. This POS tagger can help Synthesio to find the object (topic) mentionned in the text and the corresponding sentiment (for sentiment analysing).

Lexical normalization for tweets

In chapter 2, we talked about the specificities of User-Generated Content (UGC) compared to standard texts, and traditional NLP tools' performances decline on UGC. If we want to continue to use these traditional NLP tools, we need a special preprocessing procedure for the noisy, non-canonical UGC. We tested lexical normalization model as a preprocessing step for UGC. This normalization procedure contains three levels.

The first two levels are about to create a vocabulary using unlabelled data. Synthesio processes a lot of semi-structured data. The meta-data of each document contains informations as date, source (urls), platform (like Facebook, Twitter) etc... We want to make use of this mass of unlabelled data.

The first level was morpho-lexical normalization. On this level, we tried to normalize the form in lowercase or uppercase. Tokens are replaced into their most frequent form according to a large Synthesio dataset. For example, for the token all letters in lowercase "nissan", the form "Nissan" is the most frequent form with almost 90%. But there are also other forms like "NISSAN", "nissan" even "NIssan". When we replace all the other forms by the most

frequent form "Nissan", we can limit our vocabulary size without changing any text meaning.

The second normalization step dealt with the lexical level. UGC and well-formed texts both contain non-standard words, numbers, abbreviations, dates, currency amounts, acronyms, etc. In addition, these texts contain a lot of unknown words and spelling errors.

When we tried to analyze a large quantity of data, we found out that there were tokens with similar tokens which express volume, date, currency, etc. For example, "3000" and "3k" have the same meaning: both of them refer to the number of three thousand. On the other hand, sometimes we do not take exact numbers into account. All numeric values can be normalized with a simple indication of quantity in the normalization stage. For example, when one text talks about "3m people" (3 million people) and another text talks about "five hundred people", they are both about some number of people that we can ignore the exact number. So if we normalize tokens "3m" and "five" in only one expression, that would limit the number of vocabulary.

As a result of the first two levels of lexical normalization, we trained a word2vec model with unannotated Synthesio data. In this word2vec model, tokens were represented by their most frequent forms (on the morpho-lexical level). Quantity, volume expression and date expression were unified into general labels like "quantity", "volume" and "date" etc.

The third normalization step also dealt with the lexical level. We tried to find non-standard words (NSW) to replace them with standard forms.

Non-standard words (NSW) contain out-of-vocabulary (OOV) words and in some cases, in-vocabulary words as showed in the NSW definition figure 2.10 of chapter 2.

Not all OOV words are non-standard words (NSW). For example, some named entity which are not present in a standard vocabulary (like "Obama") are OOV words but they are in standard form because we do not need to correct them and it is impossible to correct them) so they are standard words (SW).

Sometimes an in-vocabulary (IV) word is NSW, as "wit" in "I will go wit you". Here "wit" exists in standard English dictionary but it should be detected as non-standard word and be replaced by its standard form "with" according to this context.

Noises in a text are incomprehensible words or words which have no semantic meanings. They were considered as standard words (SW) because we

cannot find a standard form to replace the noisy tokens (like "hahahaha") for the following step.

The definition of non-standard words (NSW) makes it difficult to classify all tokens in a tokenized text into non-standard words (NSW) and standard words (SW).

We first developed a simple SVM model to classify NSW and SW in order to provide a baseline. Then we developed and tested more complex neural networks (NN) models: long short-term memory (LSTM) model and a convolutional neural networks (CNN) model.

This CNN model used character-level representation for the current token and two words before and after the current token as context window. We achieved a F1-measure of the NSW class at about 96% on cross validation with a mixed dataset from (Baldwin et al., 2015) and (Li and Liu, 2014). This score is a little less than that in Baldwin et al., 2015 (98% on F1-measure for NSW class).

We then developed a word normalization model proposing a correct form (in a defined vocabulary) to replace a NSW token in the text.

There were two possibilities to use the normalization model. We could train this normalization model only on NSW and evaluate on NSW words, which implied that all tokens had to be correctly classified on NSW and SW. We could also train this normalization model on NSW and SW. For SW, the correct form proposed by the normalization model was just its form itself in the text. We evaluated this normalization model on all tokens (both NSW and SW). For each case, we also tested the normalization model with the context (two words before and two words after) and without the context of the current token.

The normalization model took the sequence of all characters in the token as input. We also trained a character representation from all words in the vocabulary, to take into account relations between characters in one word. Then as output, the normalization model gave an in-vocabulary (IV) word as candidate proposed for the given token, or an "unknown" label for an out-of-vocabulary (OOV) word proposed. In this case, if the correction of the token was an OOV word, the proposed "unknown" label was considered as a correct answer, as if the normalization model had found the correct form.

As a conclusion, the normalization model trained only on NSW had more accuracy than the normalization model working for both NSW and SW (80% vs 73%). But for the correct forms on out-of-vocabulary (OOV) word, our

normalization model only returned an "unknown" label as it was trained to do.

Named entity recognition

The Named Entity Recognition (NER) task on user-generated content (UGC) was the main subject of this thesis.

Ritter et al., 2011 already showed that existing Named Entity Recognizers do not perform so well on User-Generated Content (UGC) because of spelling errors, noises and syntactic errors in the text. They also developed a NER tool trained on an annotated dataset of tweets.

However, we could not use this tool directly for Synthesio because Synthesio had a different set of named entities types. But based on this annotated dataset, Synthesio defined a subset of similar types as `geo-location`, `sportsteam`, `person`, `facility`, `company` and `product`. Among these types of named entities, `company`, `product` and `sportsteam` could be the ones of Synthesio's customers (potential customer in the future). Other types of named entities in the dataset of Ritter et al., 2011 like `music artists`, `TV shows` and `movies` are less likely to be Synthesio's client and Synthesio wants to merge these three classes into one: `media`. We also added one type of named entity `job title` for Synthesio. We kept the type `other` for other types of named entities like holidays, constellation names, etc.

Similar to Ritter et al., 2011, we found that our CRFs named entity recognition model in chapter 3 did not perform well on User-Generated Content (UGC). We tried to improve those results for the rest of the thesis.

The first method that we tried was to add part-of-speech tag as a feature in the CRFs model for NER.

The part-of-speech tag is important for the NER task. We included the POS tag as features in our CRFs model for NER task, but there were problems with the annotated POS tag in the Penn Treebank. On the one hand, there was a lot of out-of-vocabulary (OOV) words in our evaluation dataset (created by Synthesio), which were not present in the Penn Treebank dataset (we labelled these tokens with "None" label). On the other hand, there were 65 POS tags in the Penn Treebank's tagset and sometimes one token could have more than one possible POS tags in different contexts in the dataset (up to 7 possible POS tags). This feature was represented by 7 sub-features: the most frequent POS tag, the second frequent POS tag, etc. These 7 sub-features were then considered as independent features for all tokens.

Similarly to NER task, the POS tagging on user-generated content (UGC) declined compared to standard text.

We developed our own POS tagger trained on UGC, which predicted tokens with 17 possible POS tags from defined POS tagset (from chapter 3). We then used this POS tagger to first predict tokenized texts and then used the predicted POS tag as a CRF feature in our NER model.

As a conclusion, according to the results in chapter 4, the CRFs model with the results of our POS tagger was slightly better than the other models (with POS tag in Penn Treebank as feature) for some types of entities: `company`, `person` and some types of entities that do not have any correctly extracted entities: `media` and `sportsteam`.

Another method that we tried in order to improve the NER results was lexical normalization. We first created a large unannotated Synthesio dataset and we normalized uppercase/lowercase forms. Then we normalized some common expressions such as date, quantity, volume, etc. We then trained a word2vec model with this dataset and we obtained a vector representation of this dataset. We used this word2vec model as word representation for the following NSW/SW classifier and the word normalization model.

To add the normalized tokens as a feature in the CRFs model for the NER task, we classified NSW and SW tokens and developed a normalization model which could propose an in-vocabulary (IV) word to replace a NSW token. If the correction of the NSW token was an out-of-vocabulary word, the normalization model would label it "unknown".

According to the results in chapter 6, the NER model was slightly better with the correct form proposed by the normalization model as a CRF feature.

The NER result remained largely worse on Synthesio's reference evaluation dataset than on dataset of Ritter et al., 2011. On one hand, all of our CRF models are developed to maximise the cross validation on the dataset from Ritter et al., 2011 and we expected that the model would adapt to the domains of Synthesio dataset. The result means that the Synthesio dataset was too various and too different from Ritter et al., 2011 even though both are User-Generated Content (UGC). On the other hand, the Synthesio evaluation dataset was created to test our CRF models and this dataset was annotated by our CRF model developper. So the annotation was stricter and more complex.

7.2 Limitations and Perspective

This work aimed to help the company Synthesio with its natural language processing (NLP) task: the named entity recognition (NER).

Since the beginning, creating a client list and extracting only the corresponding tokens has not been an option because it was difficult to maintain the client list in all 80 language and that would only work for named entity types like company, product. It would be almost impossible to create a list of person names and a geo-location list. That is why we considered a machine learning method.

We first created an annotated dataset on Synthesio data and we evaluated our model on it. However, in that dataset, there were doubled texts and the vocabulary and contexts were very limited. For example, in the domain of "Nissan", we had already selected texts with the token "Nissan" (because we extracted texts for this client), but this token was absent in the training data and it appeared in almost all the texts in the domain, that is a missing named entity of all occurrence of the token "Nissan" (the company name for the named entity).

On the other hand, this dataset was very different from the one of Ritter et al., 2011, our CRF models known to be working fine during cross validation on the dataset from Ritter et al., 2011, did not perform as well as we expected. These two datasets were from different domains and even the dataset of Synthesio was from four different domains. For example, our CRF model never learnt that "Nissan" was a company. It is almost impossible for a CRF model to predict some token with its value and its context since the model has never learned that "Nissan" is a company.

For future work, Synthesio should create a proper training dataset which covers a maximum number of domains to try to train a proper CRF model. If this data is not sufficient, we can mix the Synthesio training dataset with the dataset in (Ritter et al., 2011) for domain adaptation as we did for POS tagger in chapter 3.

The lexical normalization part of this thesis aimed to improve the NER results. So we did not annotate Synthesio dataset with correct form for non-standard words (NSW) and we did not develop a real pipeline for this normalization task. We stopped at the point that the correction of a token could be an out-of-vocabulary (OOV) word by only labelling "unknown" as the proposed correct form.

The entire pipeline could be: first classify a token as non-standard word

(NSW) or standard word (SW). Then for SW, the model will propose its original form as correction. As for NSW, the model will propose an in-vocabulary (IV) word as proposition of correct form as we already did. After that, if the proposition is an out-of-vocabulary (OOV) word, the model will propose a sequence of characters (possible spellings) as corrections (as the test that we did in the chapter 5). Finally, if necessary, we could develop a post-treatment to match the proposed form with the nearest word in the vocabulary using edit distance.

We could evaluate our NSW/SW classifier and lexical normalization model with this dataset. There are less variances on the character level (with spelling variance) than the named entity level.

We only normalized texts on the morpho-syntactic level. There are already studies on phonetic level as in Xu, Xia, and Lee, 2015. They used the cmu pronouncing dictionary [1] to represent tokens on phonemes according to their pronunciation. For example, if the symbol `'f'` represents the pronunciation of the letter 'f', the combination of letters 'ph' will have the same representation since this combination pronounces the same as the letter 'f'. This is a new method to represent a given token. With this method, we can imagine that tokens variance like "thanks" with undefined number of 's' in the end like "thanksssssss" will have the same representation by pronunciation because the sequence of some repeating 's' is spelled the same as only one letter 's'.

[1] http://www.speech.cs.cmu.edu/cgi-bin/cmudict

Bibliography

Abeillé, Anne, Lionel Clément, and François Toussenel (2003). "Building a Treebank for French". In: *Treebanks: Building and Using Parsed Corpora*. Ed. by Anne Abeillé. Dordrecht: Springer Netherlands, pp. 165–187. ISBN: 978-94-010-0201-1. DOI: 10.1007/978-94-010-0201-1_10. URL: https://doi.org/10.1007/978-94-010-0201-1_10.

Akhtar, Md Shad, Utpal Kumar Sikdar, and Asif Ekbal (2015). "IITP: Hybrid Approach for Text Normalization in Twitter". In: *Proceedings of the Workshop on Noisy User-generated Text*. Beijing, China: Association for Computational Linguistics, pp. 106–110. DOI: 10.18653/v1/W15-4316. URL: http://aclweb.org/anthology/W15-4316.

Altman, N. S. (1992). "An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression". In: *The American Statistician* 46.3, pp. 175–185. DOI: 10.1080/00031305.1992.10475879. eprint: https://www.tandfonline.com/doi/pdf/10.1080/00031305.1992.10475879. URL: https://www.tandfonline.com/doi/abs/10.1080/00031305.1992.10475879.

Alvarado, Julio Cesar Salinas, Karin Verspoor, and Timothy Baldwin (2015). "Domain adaption of named entity recognition to support credit risk assessment". In: *Proceedings of the Australasian Language Technology Association Workshop 2015*, pp. 84–90.

Baldwin, Timothy et al. (2015). "Shared Tasks of the 2015 Workshop on Noisy User-generated Text: Twitter Lexical Normalization and Named Entity Recognition". In: *Proceedings of the Workshop on Noisy User-generated Text*. Beijing, China: Association for Computational Linguistics, pp. 126–135. DOI: 10.18653/v1/W15-4319. URL: http://aclweb.org/anthology/W15-4319.

Beckley, Russell (2015). "Bekli:A Simple Approach to Twitter Text Normalization." In: *Proceedings of the Workshop on Noisy User-generated Text*. Beijing, China: Association for Computational Linguistics, pp. 82–86. DOI: 10.18653/v1/W15-4312. URL: http://aclweb.org/anthology/W15-4312.

Ben-David, Shai et al. (May 2010). "A Theory of Learning from Different Domains". In: *Machine Learning* 79.1-2, pp. 151–175. ISSN: 0885-6125. DOI: 10.1007/s10994-009-5152-4. URL: https://doi.org/10.1007/s10994-009-5152-4.

Bender, Oliver, Franz Josef Och, and Hermann Ney (2003). "Maximum Entropy Models for Named Entity Recognition". In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*. CONLL '03. Edmonton, Canada: Association for Computational Linguistics, pp. 148–151. DOI: 10.3115/1119176.1119196. URL: https://doi.org/10.3115/1119176.1119196.

Berend, Gábor and Ervin Tasnádi (2015). "USZEGED: Correction Type-sensitive Normalization of English Tweets Using Efficiently Indexed n-gram Statistics". In: *Proceedings of the Workshop on Noisy User-generated Text*. Beijing, China: Association for Computational Linguistics, pp. 120–125. DOI: 10.18653/v1/W15-4318. URL: http://aclweb.org/anthology/W15-4318.

Blei, David M, Andrew Y Ng, and Michael I Jordan (2003). "Latent dirichlet allocation". In: *Journal of machine Learning research* 3.Jan, pp. 993–1022.

Bollen, Johan, Huina Mao, and Xiao-Jun Zeng (Oct. 2010). "Twitter Mood Predicts the Stock Market". In: *Journal of Computational Science*.

Bondy, J.A. and U.S.R Murty (2008). *Graph Theory*. 1st. Springer Publishing Company, Incorporated. ISBN: 1846289696.

Brown, Peter F. et al. (Dec. 1992). "Class-based N-gram Models of Natural Language". In: *Computational Linguistics* 18.4, pp. 467–479. ISSN: 0891-2017. URL: http://dl.acm.org/citation.cfm?id=176313.176316.

Bunescu, Razvan C. and Raymond J. Mooney (Dec. 2005). "A shortest path dependency kernel for relation extraction". English (US). In: *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*. Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, HLT/EMNLP 2005, Co-located with the 2005 Document Understanding Conference, DUC and the 9th International Workshop on Parsing Technologies, IWPT ; Conference date: 06-10-2005 Through 08-10-2005, pp. 724–731. URL: http://www.aclweb.org/anthology/H05-1091.

Byrne, Kate (2007). "Nested named entity recognition in historical archive text". In: *Semantic Computing, 2007. ICSC 2007. International Conference on*. IEEE, pp. 589–596.

Cano Basave, Amparo Elizabeth et al. (2013). "Making sense of microposts (#MSM2013) concept extraction challenge". In: *#MSM2013 : concept extraction challenge at Making Sense of Microposts 2013*. Ed. by Amparo E. Cano et al. CEUR workshop proceedings. CEUR-WS.org, pp. 1–15.

Cardoso, Pedro Dias and Anindya Roy (2016). "Sentiment Lexicon Creation using Continuous Latent Space and Neural Networks". In: *Proceedings of the 7th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pp. 37–42.

Cherry, Colin and Hongyu Guo (2015). "The unreasonable effectiveness of word representations for twitter named entity recognition". In: *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 735–745.

Chieu, Hai Leong and Hwee Tou Ng (2003). "Named Entity Recognition with a Maximum Entropy Approach". In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*. CONLL '03. Edmonton, Canada: Association for Computational Linguistics, pp. 160–163. DOI: 10.3115/1119176.1119199. URL: https://doi.org/10.3115/1119176.1119199.

Chinchor, N. and P. Robinson (1998). "Appendix E: MUC-7 Named Entity Task Definition (version 3.5)". In: *Seventh Message Understanding Conference (MUC-7): Proceedings of a Conference Held in Fairfax, Virginia, April 29 - May 1, 1998*. URL: http://www.aclweb.org/anthology/M98-1028.

Chinchor, Nancy (1999). "Overview of MUC-7". In: *Seventh Message Understanding Conference (MUC-7): Proceedings of a Conference Held in Fairfax, Virginia, April 29-May 1, 1998*.

Chomsky, Noam (1957). *Syntactic Structures*. The Hague: Mouton and Co.

Chrupała, Grzegorz (2014). "Normalizing tweets with edit scripts and recurrent neural embeddings". In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Baltimore, Maryland: Association for Computational Linguistics, pp. 680–686. DOI: 10.3115/v1/P14-2111. URL: http://aclweb.org/anthology/P14-2111.

Collobert, Ronan and Jason Weston (2008). "A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning". In: *Proceedings of the 25th International Conference on Machine Learning*. ICML '08. Helsinki, Finland: ACM, pp. 160–167. ISBN: 978-1-60558-205-4. DOI: 10.1145/1390156.1390177. URL: http://doi.acm.org/10.1145/1390156.1390177.

Collobert, Ronan et al. (Nov. 2011). "Natural Language Processing (Almost) from Scratch". In: *Journal of Machine Learning Research (2011)* 12, pp. 2493–2537. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=1953048.2078186.

Constant, Mathieu et al. (June 2011). "Intégrer des connaissances linguistiques dans un CRF : application à l'apprentissage d'un segmenteur-étiqueteur du français". In: *TALN*. Vol. 1. Montpellier, France, p. 321. URL: https://hal.archives-ouvertes.fr/hal-00620923.

Cortes, Corinna and Vladimir Vapnik (1995). "Support-vector networks". In: *Machine learning* 20.3, pp. 273–297.

Cunningham, Hamish et al. (2002). "GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications". In: *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*.

Curran, James R. and Stephen Clark (2003). "Language Independent NER Using a Maximum Entropy Tagger". In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*. CONLL '03. Edmonton, Canada: Association for Computational Linguistics, pp. 164–167. DOI: 10.3115/1119176.1119200. URL: https://doi.org/10.3115/1119176.1119200.

Denis, Pascal, Jason Baldridge, et al. (2009). "Global joint models for coreference resolution and named entity classification". In: *Procesamiento del Lenguaje Natural* 42.1, pp. 87–96. URL: https://hal.inria.fr/inria-00514302.

Derczynski, Leon R. A., Bin Yang, and Christian S. Jensen (2013). "Towards Context-aware Search and Analysis on Social Media Data". In: *Proceedings of the 16th International Conference on Extending Database Technology*. EDBT '13. Genoa, Italy: ACM, pp. 137–142. ISBN: 978-1-4503-1597-5. DOI: 10.1145/2452376.2452393. URL: http://doi.acm.org/10.1145/2452376.2452393.

Dias Cardoso, Pedro Miguel and Anindya Roy (2016). "Language Identification for Social Media: Short Messages and Transliteration". In: *Proceedings of the 25th International Conference Companion on World Wide Web*. WWW '16 Companion. Montréal, Québec, Canada: International World Wide Web Conferences Steering Committee, pp. 611–614. ISBN: 978-1-4503-4144-8. DOI: 10.1145/2872518.2890560. URL: https://doi.org/10.1145/2872518.2890560.

Doddington, George R et al. (2004). "The Automatic Content Extraction (ACE) Program-Tasks, Data, and Evaluation". In: *Proceedings of Language Resources and Evaluation Conference 2004*. Vol. 2, pp. 837–840.

Dupont, Yoann (Nov. 2017). "Structuration in named entities". Theses. Université Sorbonne Paris Cité. URL: https://tel.archives-ouvertes.fr/tel-01772268.

Durrett, Greg and Dan Klein (2014). "A joint model for entity analysis: Coreference, typing, and linking". In: *Transactions of the Association for Computational Linguistics* 2, pp. 477–490.

Ehrmann, Maud (June 2008). "Named entities, from Linguistics to NLP: Theoretical status and disambiguation methods". Theses. Paris Diderot University. URL: https://hal.archives-ouvertes.fr/tel-01639190.

Elman, Jeffrey L (1990). "Finding structure in time". In: *Cognitive science* 14.2, pp. 179–211.

Erp, Marieke Van, Giuseppe Rizzo, and Raphaël Troncy (May 2013). "Learning with the Web: Spotting named entities on the intersection of NERD and machine learning". In: *WWW 2013, 3rd International Workshop on Making Sense of Microposts (#MSM'13), Concept Extraction Challenge, May 13, 2013, Rio de Janeiro, Brazil*. Rio de Janeiro, BRAZIL. URL: http://www.eurecom.fr/publication/3968.

Evans, Richard (2003). "A framework for named entity recognition in the open domain". In: *In Proceedings of the Recent Advances in Natural Language Processing (RANLP*, pp. 137–144.

Finkel, Jenny Rose, Trond Grenager, and Christopher Manning (2005). "Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling". In: *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. ACL '05. Ann Arbor, Michigan: Association for Computational Linguistics, pp. 363–370. DOI: 10.3115/1219840.1219885. URL: https://doi.org/10.3115/1219840.1219885.

Finkel, Jenny Rose and Christopher D. Manning (2009). "Nested Named Entity Recognition". In: *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*. EMNLP '09. Singapore: Association for Computational Linguistics, pp. 141–150. ISBN: 978-1-932432-59-6. URL: http://dl.acm.org/citation.cfm?id=1699510.1699529.

Florian, Radu et al. (2003). "Named Entity Recognition Through Classifier Combination". In: *Proceedings of the Seventh Conference on Natural Language*

Learning at HLT-NAACL 2003 - Volume 4. CONLL '03. Edmonton, Canada: Association for Computational Linguistics, pp. 168–171. DOI: 10.3115/1119176.1119201. URL: https://doi.org/10.3115/1119176.1119201.

Forney, G David (1973). "The viterbi algorithm". In: *Proceedings of the Institute of Electrical and Electronics Engineers (IEEE)* 61.3, pp. 268–278.

Fort, Karën, Maud Ehrmann, and Adeline Nazarenko (2009). "Towards a Methodology for Named Entities Annotation". In: *Proceedings of the Third Linguistic Annotation Workshop*. ACL-IJCNLP '09. Suntec, Singapore: Association for Computational Linguistics, pp. 142–145. ISBN: 978-1-932432-52-7. URL: http://dl.acm.org/citation.cfm?id=1698381.1698406.

Foster, Jennifer et al. (2011). "#hardtoparse: POS Tagging and Parsing the Twitterverse". In: *AAAI 2011 Workshop On Analyzing Microtext*. United States, pp. 20–25. URL: https://hal.archives-ouvertes.fr/hal-00702445.

Garcia-Fernandez, Anne, Olivier Ferret, and Marco Dinarelli (2014). "Evaluation of Different Strategies for Domain Adaptation in Opinion Mining". In: *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. Ed. by Nicoletta Calzolari (Conference Chair) et al. Reykjavik, Iceland: European Language Resources Association (ELRA). ISBN: 978-2-9517408-8-4.

George, Carlisle and Jackie Scerri (June 2018). "Web 2.0 and User-Generated Content: Legal Challenges in the New Frontier". In: *Journal of Information, Law and Technology*.

Gimpel, Kevin et al. (2011). "Part-of-speech Tagging for Twitter: Annotation, Features, and Experiments". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*. HLT '11. Portland, Oregon: Association for Computational Linguistics, pp. 42–47. ISBN: 978-1-932432-88-6. URL: http://dl.acm.org/citation.cfm?id=2002736.2002747.

Glorot, Xavier and Yoshua Bengio (May 2010). "Understanding the difficulty of training deep feedforward neural networks". In: *JMLR W&CP: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2010)*. Vol. 9. Chia Laguna Resort, Sardinia, Italy, pp. 249–256.

Godin, Fréderic et al. (2015). "Multimedia Lab $@$ ACL WNUT NER Shared Task: Named Entity Recognition for Twitter Microposts using Distributed

Word Representations". In: *Proceedings of the Workshop on Noisy User-generated Text*. Beijing, China: Association for Computational Linguistics, pp. 146–153. DOI: 10.18653/v1/W15-4322. URL: http://aclweb.org/anthology/W15-4322.

Gouws, Stephan, Dirk Hovy, and Donald Metzler (2011). "Unsupervised Mining of Lexical Variants from Noisy Text". In: *Proceedings of the First Workshop on Unsupervised Learning in NLP*. EMNLP '11. Edinburgh, Scotland: Association for Computational Linguistics, pp. 82–90. ISBN: 978-1-937284-13-8. URL: http://dl.acm.org/citation.cfm?id=2140458.2140468.

Grefenstette, Edward et al. (2014). "A Deep Architecture for Semantic Parsing". In: *Proceedings of the ACL 2014 Workshop on Semantic Parsing*. Baltimore, MD: Association for Computational Linguistics, pp. 22–27. DOI: 10.3115/v1/W14-2405. URL: http://www.aclweb.org/anthology/W14-2405.

Grishman, Ralph and Beth Sundheim (1996). "Message Understanding Conference-6: A Brief History". In: *Proceedings of the 16th Conference on Computational Linguistics - Volume 1*. COLING '96. Copenhagen, Denmark: Association for Computational Linguistics, pp. 466–471. DOI: 10.3115/992628.992709. URL: https://doi.org/10.3115/992628.992709.

Guo, Honglei et al. (2009). "Domain Adaptation with Latent Semantic Association for Named Entity Recognition". In: *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. NAACL '09. Boulder, Colorado: Association for Computational Linguistics, pp. 281–289. ISBN: 978-1-932432-41-1. URL: http://dl.acm.org/citation.cfm?id=1620754.1620795.

Gupta, Vishal and Gurpreet Singh Lehal (2011). "Article: Named Entity Recognition for Punjabi Language Text Summarization". In: *International Journal of Computer Applications* 33.3, pp. 28–32.

Hajishirzi, Hannaneh et al. (2013). "Joint Coreference Resolution and Named-Entity Linking with Multi-Pass Sieves." In: *EMNLP*. ACL, pp. 289–299. ISBN: 978-1-937284-97-8. URL: http://dblp.uni-trier.de/db/conf/emnlp/emnlp2013.html#HajishirziZWZ13.

Hammerton, James (2003). "Named Entity Recognition with Long Short-term Memory". In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*. CONLL '03. Edmonton, Canada: Association for Computational Linguistics, pp. 172–175. DOI: 10.3115/

1119176.1119202. URL: https://doi.org/10.3115/1119176.1119202.

Han, Bo (2014). "Improving the utility of social media with Natural Language Processing". In: URL: http://hdl.handle.net/11343/41029.

Han, Bo and Timothy Baldwin (2011). "Lexical Normalisation of Short Text Messages: Makn Sens a #Twitter". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*. HLT '11. Portland, Oregon: Association for Computational Linguistics, pp. 368–378. ISBN: 978-1-932432-87-9. URL: http://dl.acm.org/citation.cfm?id=2002472.2002520.

Han, Sangdo et al. (2017). "Answer Ranking Based on Named Entity Types for Question Answering". In: *Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication*. IMCOM '17. Beppu, Japan: ACM, 71:1–71:4. ISBN: 978-1-4503-4888-1. DOI: 10.1145/3022227.3022297. URL: http://doi.acm.org/10.1145/3022227.3022297.

Hearst, Marti A. (1992). "Automatic Acquisition of Hyponyms from Large Text Corpora". In: *Proceedings of the 14th Conference on Computational Linguistics - Volume 2*. COLING '92. Nantes, France: Association for Computational Linguistics, pp. 539–545. DOI: 10.3115/992133.992154. URL: https://doi.org/10.3115/992133.992154.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory". In: *Neural computation* 9.8, pp. 1735–1780.

Hu, Yuheng et al. (2012). "What Were the Tweets About? Topical Associations between Public Events and Twitter Feeds". In: *Proceedings of the Sixth International AAAI Conference on Weblogs and Social Media (ICWSM)*.

Jaccard, Paul (1912). "The distribution of the flora in the alpine zone. 1". In: *New phytologist* 11.2, pp. 37–50.

Jiang, Long et al. (2011). "Target-dependent Twitter Sentiment Classification". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*. HLT '11. Portland, Oregon: Association for Computational Linguistics, pp. 151–160. ISBN: 978-1-932432-87-9. URL: http://dl.acm.org/citation.cfm?id=2002472.2002492.

Jin, Ning (2015). "NCSU-SAS-Ning: Candidate Generation and Feature Engineering for Supervised Lexical Normalization". In: *Proceedings of the Workshop on Noisy User-generated Text*. Beijing, China: Association for Computational Linguistics, pp. 87–92. DOI: 10.18653/v1/W15-4313. URL: http://aclweb.org/anthology/W15-4313.

Jonathan, Mark Przybocki et al. (1999). "Hub-4 Information Extraction Evaluation". In: *In Proceedings of the DARPA Broadcast News Workshop*. Morgan Kaufmann, pp. 13–18.

Kalchbrenner, Nal, Edward Grefenstette, and Phil Blunsom (2014). "A convolutional neural network for modelling sentences". In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Baltimore, Maryland: Association for Computational Linguistics, pp. 655–665. DOI: 10.3115/v1/P14-1062. URL: http://www.aclweb.org/anthology/P14-1062.

Kessler, Brett, Geoffrey Numberg, and Hinrich Schütze (1997). "Automatic Detection of Text Genre". In: *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*. ACL '98/EACL '98. Madrid, Spain: Association for Computational Linguistics, pp. 32–38. DOI: 10.3115/976909.979622. URL: https://doi.org/10.3115/976909.979622.

Kim, Yoon (2014). "Convolutional Neural Networks for Sentence Classification". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, pp. 1746–1751. DOI: 10.3115/v1/D14-1181. URL: http://www.aclweb.org/anthology/D14-1181.

Kim, Yoon et al. (2015). "Character-Aware Neural Language Models". In: *The Computing Research Repository (CoRR)* abs/1508.06615. arXiv: 1508.06615. URL: http://arxiv.org/abs/1508.06615.

Klein, Dan et al. (2003). "Named Entity Recognition with Character-level Models". In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*. CONLL '03. Edmonton, Canada: Association for Computational Linguistics, pp. 180–183. DOI: 10.3115/1119176.1119204. URL: https://doi.org/10.3115/1119176.1119204.

Kneser, Reinhard and Hermann Ney (1993). "Improved clustering techniques for class-based statistical language modelling". In: *Third European Conference on Speech Communication and Technology*.

Kong, Lingpeng et al. (2014). "A dependency parser for tweets". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1001–1012.

Kripke, Saul A (1972). "Naming and necessity". In: *Semantics of natural language*. Springer, pp. 253–355.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*, pp. 1097–1105.

Kumar, Aman et al. (2015). "Understanding Medical Named Entity Extraction in Clinical Notes". In:

Lafferty, John D., Andrew McCallum, and Fernando C. N. Pereira (2001). "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data". In: *Proceedings of the Eighteenth International Conference on Machine Learning*. ICML '01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., pp. 282–289. ISBN: 1-55860-778-1. URL: http://dl.acm.org/citation.cfm?id=645530.655813.

LeCun, Yann et al. (1989). "Generalization and network design strategies". In: *Connectionism in perspective*, pp. 143–155.

Leeman-Munk, Samuel, James Lester, and James Cox (2015). "NCSU_SAS_SAM: Deep Encoding and Reconstruction for Normalization of Noisy Text". In: *Proceedings of the Workshop on Noisy User-generated Text*. Beijing, China: Association for Computational Linguistics, pp. 154–161. DOI: 10.18653/v1/W15-4323. URL: http://aclweb.org/anthology/W15-4323.

Levenshtein, Vladimir I (1966). "Binary codes capable of correcting deletions, insertions, and reversals". In: *Soviet physics doklady*. Vol. 10. 8, pp. 707–710.

Li, Chen and Yang Liu (2014). "Improving Text Normalization via Unsupervised Model and Discriminative Reranking". In: *Proceedings of the ACL 2014 Student Research Workshop*. Baltimore, Maryland, USA: Association for Computational Linguistics, pp. 86–93. DOI: 10.3115/v1/P14-3012. URL: http://aclweb.org/anthology/P14-3012.

— (2015). "Improving Named Entity Recognition in Tweets via Detecting Non-Standard Words". In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China: Association for Computational Linguistics, pp. 929–938. DOI: 10.3115/v1/P15-1090. URL: http://aclweb.org/anthology/P15-1090.

Liang, Percy (2005). "Semi-supervised learning for natural language". PhD thesis. Massachusetts Institute of Technology.

Lilleberg, J., Y. Zhu, and Y. Zhang (2015). "Support vector machines and Word2vec for text classification with semantic features". In: *2015 IEEE 14th International Conference on Cognitive Informatics Cognitive Computing (ICCI*CC)*, pp. 136–140. DOI: 10.1109/ICCI-CC.2015.7259377.

Ling, Wang et al. (2015). "Two/Too Simple Adaptations of Word2Vec for Syntax Problems". In: *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Denver, Colorado: Association for Computational Linguistics, pp. 1299–1304. DOI: 10.3115/v1/N15-1142. URL: http://aclweb.org/anthology/N15-1142.

Liu, Fei, Fuliang Weng, and Xiao Jiang (2012). "A Broad-coverage Normalization System for Social Media Language". In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*. ACL '12. Jeju Island, Korea: Association for Computational Linguistics, pp. 1035–1044. URL: http://dl.acm.org/citation.cfm?id=2390524.2390662.

Liu, Fei et al. (2011a). "Insertion, Deletion, or Substitution?: Normalizing Text Messages Without Pre-categorization nor Supervision". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*. HLT '11. Portland, Oregon: Association for Computational Linguistics, pp. 71–76. ISBN: 978-1-932432-88-6. URL: http://dl.acm.org/citation.cfm?id=2002736.2002753.

Liu, Xiaohua et al. (2011b). "Recognizing named entities in tweets". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, pp. 359–367.

Liu, Xiaohua et al. (2012). "Joint Inference of Named Entity Recognition and Normalization for Tweets". In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*. ACL '12. Jeju Island, Korea: Association for Computational Linguistics, pp. 526–535. URL: http://dl.acm.org/citation.cfm?id=2390524.2390598.

Liu, Xiaohua et al. (2013). "Named entity recognition for tweets". In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 4.1, p. 3.

Luce, R. Duncan and Albert D. Perry (1949). "A method of matrix analysis of group structure". In: *Psychometrika* 14.2, pp. 95–116. ISSN: 1860-0980.

DOI: 10.1007/BF02289146. URL: https://doi.org/10.1007/BF02289146.

Marcus, Mitchell P, Mary Ann Marcinkiewicz, and Beatrice Santorini (1993). "Building a large annotated corpus of English: The Penn Treebank". In: *Computational linguistics* 19.2, pp. 313–330.

Marty, Patrick, Tian Tian, and Isabelle Tellier (Mar. 2014). "Extraction de propriétés de produits". In: *COnférence en Recherche d'Information et Applications (CORIA 2014)*. CORIA 2014. Nancy, France, pp. 121–136. URL: https://hal.archives-ouvertes.fr/hal-01473389.

Mayfield, James, Paul McNamee, and Christine Piatko (2003). "Named Entity Recognition Using Hundreds of Thousands of Features". In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*. CONLL '03. Edmonton, Canada: Association for Computational Linguistics, pp. 184–187. DOI: 10.3115/1119176.1119205. URL: https://doi.org/10.3115/1119176.1119205.

McCallum, Andrew and Wei Li (2003). "Early Results for Named Entity Recognition with Conditional Random Fields, Feature Induction and Web-enhanced Lexicons". In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*. CONLL '03. Edmonton, Canada: Association for Computational Linguistics, pp. 188–191. DOI: 10.3115/1119176.1119206. URL: https://doi.org/10.3115/1119176.1119206.

Merchant, Roberta, Mary Ellen Okurowski, and Nancy Chinchor (1996). "The Multilingual Entity Task (MET) Overview". In: *Proceedings of a Workshop on Held at Vienna, Virginia: May 6-8, 1996*. TIPSTER '96. Vienna, Virginia: Association for Computational Linguistics, pp. 445–447. DOI: 10.3115/1119018.1119075. URL: https://doi.org/10.3115/1119018.1119075.

Mikolov, Tomas, Wen-tau Yih, and Geoffrey Zweig (2013). "Linguistic Regularities in Continuous Space Word Representations." In: *HLT-NAACL*, pp. 746–751.

Mikolov, Tomas et al. (2013). "Distributed Representations of Words and Phrases and Their Compositionality". In: *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'13. Lake Tahoe, Nevada: Curran Associates Inc., pp. 3111–3119. URL: http://dl.acm.org/citation.cfm?id=2999792.2999959.

Miller, S., J. Guinness, and A. Zamanian (2004). "Name tagging with word clusters and discriminative training". In: *Proceedings of 2004 Human Language Technology conference / North American chapter of the Association for Computational Linguistics annual meeting*. Vol. 4.

Min, Wookhee and Bradford Mott (2015). "NCSU_SAS_WOOKHEE: A Deep Contextual Long-Short Term Memory Model for Text Normalization". In: *Proceedings of the Workshop on Noisy User-generated Text*. Beijing, China: Association for Computational Linguistics, pp. 111–119. DOI: 10.18653/v1/W15-4317. URL: http://aclweb.org/anthology/W15-4317.

Murthy, V. Rudra and Pushpak Bhattacharyya (2016). "A Deep Learning Solution to Named Entity Recognition". In: *17th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing2016)*.

Nadeau, David and Satoshi Sekine (2007). "A survey of named entity recognition and classification". In: *Linguisticae Investigationes* 30.1. Publisher: John Benjamins Publishing Company, pp. 3–26. URL: http://www.ingentaconnect.com/content/jbp/li/2007/00000030/00000001/art00002.

Nooralahzadeh, Farhad, Caroline Brun, and Claude Roux (2014). "Part of Speech Tagging for French Social Media Data". In: *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin, Ireland: Dublin City University and Association for Computational Linguistics, pp. 1764–1772. URL: http://www.aclweb.org/anthology/C14-1166.

Nouvel, Damien, Maud Ehrmann, and Sophie Rosset (2016). *Named Entities for Computational Linguistics*. URL: https://hal-inalco.archives-ouvertes.fr/hal-01359440.

N.V, Sobhana, Pabitra Mitra, and S.K. Ghosh (2010). "Article: Conditional Random Field Based Named Entity Recognition in Geological text". In: *International Journal of Computer Applications* 1.3. Published By Foundation of Computer Science, pp. 119–125.

O'Connor, Brendan T., Michel Krieger, and David Ahn (2010). "TweetMotif: Exploratory Search and Topic Summarization for Twitter". In: *In Proc. of AAAI Conference on Weblogs and Social*.

Owoputi, Olutobi et al. (2012). "Part-of-speech tagging for Twitter: Word clusters and other advances". In: *Technical Report CMU-ML-12-107*.

Owoputi, Olutobi et al. (2013). "Improved Part-of-Speech Tagging for Online Conversational Text with Word Clusters". In: *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Atlanta, Georgia: Association for Computational Linguistics, pp. 380–390. URL: http://www.aclweb.org/anthology/N13-1039.

Pandian, S. Lakshmana and T. V. Geetha (2009). "CRF Models for Tamil Part of Speech Tagging and Chunking". In: *Proceedings of the 22Nd International Conference on Computer Processing of Oriental Languages. Language Technology for the Knowledge-based Economy*. ICCPOL '09. Hong Kong: Springer-Verlag, pp. 11–22. ISBN: 978-3-642-00830-6. DOI: 10.1007/978-3-642-00831-3_2. URL: http://dx.doi.org/10.1007/978-3-642-00831-3_2.

Park, SoHyun et al. (2016). "Classifying Out-of-vocabulary Terms in a Domain-Specific Social Media Corpus". In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. Ed. by Nicoletta Calzolari (Conference Chair) et al. Portorož, Slovenia: European Language Resources Association (ELRA). ISBN: 978-2-9517408-9-1.

Pennell, Deana and Yang Liu (2011). "A Character-Level Machine Translation Approach for Normalization of SMS Abbreviations". In: *Proceedings of 5th International Joint Conference on Natural Language Processing*. Chiang Mai, Thailand: Asian Federation of Natural Language Processing, pp. 974–982. URL: http://aclweb.org/anthology/I11-1109.

Pennington, Jeffrey, Richard Socher, and Christopher D. Manning (2014). "GloVe: Global Vectors for Word Representation". In: *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543. URL: http://www.aclweb.org/anthology/D14-1162.

Petrov, Slav, Dipanjan Das, and Ryan McDonald (2012). "A Universal Part-of-Speech Tagset". In: *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*. Ed. by Nicoletta Calzolari (Conference Chair) et al. Istanbul, Turkey: European Language Resources Association (ELRA). ISBN: 978-2-9517408-7-7.

Petrović, Saša, Miles Osborne, and Victor Lavrenko (2010). "The edinburgh twitter corpus". In: *Proceedings of the NAACL HLT 2010 Workshop on Computational Linguistics in a World of Social Media*, pp. 25–26.

Plank, Barbara et al. (2014). "Adapting taggers to Twitter with not-so-distant supervision". English. In: *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Association for Computational Linguistics, pp. 1783–1792.

Poibeau, Thierry and Leila Kosseim (2001). "Proper Name Extraction from Non-Journalistic Texts". In: *Computational Linguistics in the Netherlands. Selected Papers from the Eleventh CLIN Meeting*. Ed. by W. Daelemans et al. Amsterdam/New York, pp. 144–157.

Powers, David M. W. (1998). "Applications and Explanations of Zipf's Law". In: *Proceedings of the Joint Conferences on New Methods in Language Processing and Computational Natural Language Learning*. NeMLaP3/CoNLL '98. Sydney, Australia: Association for Computational Linguistics, pp. 151–160. ISBN: 0-7258-0634-6. URL: http://dl.acm.org/citation.cfm?id=1603899.1603924.

Putthividhya, Duangmanee (Pew) and Junling Hu (2011). "Bootstrapped named entity recognition for product attribute extraction". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. EMNLP '11. Edinburgh, United Kingdom: Association for Computational Linguistics, pp. 1557–1567. ISBN: 978-1-937284-11-4. URL: http://dl.acm.org/citation.cfm?id=2145432.2145598.

Quimbaya, Alexandra Pomares et al. (2016). "Named Entity Recognition Over Electronic Health Records Through a Combined Dictionary-based Approach". In: *Procedia Computer Science* 100. International Conference on ENTERprise Information Systems/International Conference on Project MANagement/International Conference on Health and Social Care Information Systems and Technologies, CENTERIS/ProjMAN / HCist 2016, pp. 55–61. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2016.09.123. URL: http://www.sciencedirect.com/science/article/pii/S1877050916322906.

Quinlan, J. Ross (1986). "Induction of decision trees". In: *Machine learning* 1.1, pp. 81–106.

Rajaraman, Anand and Jeffrey David Ullman (2011). *Mining of Massive Datasets*. New York, NY, USA: Cambridge University Press. ISBN: 1107015359, 9781107015357.

Ramshaw, Lance A and Mitchell P Marcus (1999). "Text chunking using transformation-based learning". In: *Natural language processing using very large corpora*. Springer, pp. 157–176.

Ratinov, Lev and Dan Roth (2009). "Design Challenges and Misconceptions in Named Entity Recognition". In: *Proceedings of the Thirteenth Conference*

on Computational Natural Language Learning. CoNLL '09. Boulder, Colorado: Association for Computational Linguistics, pp. 147–155. ISBN: 978-1-932432-29-9. URL: http://dl.acm.org/citation.cfm?id=1596374.1596399.

Rau, L. F. (1991). "Extracting company names from text". In: *[1991] Proceedings. The Seventh IEEE Conference on Artificial Intelligence Application*. Vol. i, pp. 29–32. ISBN: 0-8186-2135-4. DOI: 10.1109/CAIA.1991.120841.

Raymond, Christian and Julien Fayolle (July 2010). "Reconnaissance robuste d'entités nommées sur de la parole transcrite automatiquement". Français. In: *TALN'10*. ATALA. Montréal, Québec, Canada. URL: http://hal.inria.fr/inria-00561732.

Ritter, Alan et al. (2011). "Named Entity Recognition in Tweets: An Experimental Study". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. EMNLP '11. Edinburgh, United Kingdom: Association for Computational Linguistics, pp. 1524–1534. ISBN: 978-1-937284-11-4. URL: http://dl.acm.org/citation.cfm?id=2145432.2145595.

Rong, Xin (2014). "word2vec parameter learning explained". In: *arXiv preprint arXiv:1411.2738*.

Rosenblatt, F. (1957). *The Perceptron, a Perceiving and Recognizing Automaton Project Para*. Report: Cornell Aeronautical Laboratory. Cornell Aeronautical Laboratory. URL: https://books.google.fr/books?id=P_XGPgAACAAJ.

Seddah, Djamé et al. (2012). "The French Social Media Bank: a treebank of noisy user generated content". In: *COLING 2012-24th International Conference on Computational Linguistics*.

Settles, Burr (2004). "Biomedical Named Entity Recognition Using Conditional Random Fields and Rich Feature Sets". In: *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and Its Applications*. JNLPBA '04. Geneva, Switzerland: Association for Computational Linguistics, pp. 104–107. URL: http://dl.acm.org/citation.cfm?id=1567594.1567618.

Sha, Fei and Fernando Pereira (2003). "Shallow Parsing with Conditional Random Fields". In: *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*. NAACL '03. Edmonton, Canada: Association for Computational Linguistics, pp. 134–141. DOI: 10.3115/1073445.1073473. URL: https://doi.org/10.3115/1073445.1073473.

Shen, Yelong et al. (2014). "Learning Semantic Representations Using Convolutional Neural Networks for Web Search". In: *Proceedings of the 23rd International Conference on World Wide Web*. WWW '14 Companion. Seoul, Korea: ACM, pp. 373–374. ISBN: 978-1-4503-2745-9. DOI: 10.1145/2567948.2577348. URL: http://doi.acm.org/10.1145/2567948.2577348.

Sienčnik, Scharolta Katharina (2015). "Adapting word2vec to named entity recognition". In: *Proceedings of the 20th nordic conference of computational linguistics, nodalida 2015, may 11-13, 2015, vilnius, lithuania*. 109. Linköping University Electronic Press, pp. 239–243.

Silfverberg, Miikka et al. (2014). "Part-of-Speech Tagging using Conditional Random Fields: Exploiting Sub-Label Dependencies for Improved Accuracy". In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Baltimore, Maryland: Association for Computational Linguistics, pp. 259–264. DOI: 10.3115/v1/P14-2043. URL: http://www.aclweb.org/anthology/P14-2043.

Sproat, Richard et al. (July 2001). "Normalization of Non-standard Words". In: *Comput. Speech Lang.* 15.3, pp. 287–333. ISSN: 0885-2308. DOI: 10.1006/csla.2001.0169. URL: http://dx.doi.org/10.1006/csla.2001.0169.

Sundheim, Beth M. (1995). "Overview of Results of the MUC-6 Evaluation". In: *Proceedings of the 6th Conference on Message Understanding*. MUC6 '95. Columbia, Maryland: Association for Computational Linguistics, pp. 13–31. ISBN: 1-55860-402-2. DOI: 10.3115/1072399.1072402. URL: https://doi.org/10.3115/1072399.1072402.

Supranovich, Dmitry and Viachaslau Patsepnia (2015). "Ihs_rd: Lexical normalization for english tweets". In: *Proceedings of the Workshop on Noisy User-generated Text*, pp. 78–81.

Surdeanu, Mihai (2013). "Overview of the TAC2013 Knowledge Base Population Evaluation: English Slot Filling and Temporal Slot Filling". In: *Proceedings of the Sixth Text Analysis Conference, TAC 2013, Gaithersburg, Maryland, USA, November 18-19, 2013*. URL: http://www.nist.gov/tac/publications/2013/additional.papers/KBP2013_English_and_Temporal_Slot_Filling_overview.TAC2013.proceedings.pdf.

Tarrade, Louise et al. (June 2017). *Typologies pour l'annotation de textes non standard en français*. TALN 2017. Poster. URL: https://hal.archives-ouvertes.fr/hal-01646442.

Tellier, Isabelle and Marc Tommasi (Jan. 2011). "Champs Markoviens Conditionnels pour l'extraction d'information". In: *Modèles probabilistes pour l'accès à l'information textuelle*. Ed. by E. Gaussier et F. Yvon, pp. 223–267.

Tian, Tian et al. (June 2015). "Etiquetage morpho-syntaxique de tweets avec des CRF". In: *TALN 2015*. Caen, France. URL: https://hal.archives-ouvertes.fr/hal-01473383.

Tian, Tian et al. (2016). "Domain Adaptation for Named Entity Recognition Using CRFs". In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. Ed. by Nicoletta Calzolari (Conference Chair) et al. Portorož, Slovenia: European Language Resources Association (ELRA). ISBN: 978-2-9517408-9-1.

Tjong Kim Sang, Erik F. (2002). "Introduction to the CoNLL-2002 Shared Task: Language-independent Named Entity Recognition". In: *Proceedings of the 6th Conference on Natural Language Learning - Volume 20*. COLING-02. Stroudsburg, PA, USA: Association for Computational Linguistics, pp. 1–4. DOI: 10.3115/1118853.1118877. URL: https://doi.org/10.3115/1118853.1118877.

Tjong Kim Sang, Erik F. and Fien De Meulder (July 2003). "Introduction to the CoNLL-2003 Shared Task: Language-independent Named Entity Recognition". In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*. CONLL '03. Edmonton, Canada: Association for Computational Linguistics, pp. 142–147. DOI: 10.3115/1119176.1119195. URL: https://doi.org/10.3115/1119176.1119195.

Toh, Zhiqiang, Bin Chen, and Jian Su (2015). "Improving twitter named entity recognition using word representations". In: *Proceedings of the Workshop on Noisy User-generated Text*, pp. 141–145.

Toutanova, Kristina and Christopher D. Manning (2000). "Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-speech Tagger". In: *Proceedings of the 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora: Held in Conjunction with the 38th Annual Meeting of the Association for Computational Linguistics - Volume 13*. EMNLP '00. Hong Kong: Association for Computational Linguistics, pp. 63–70. DOI: 10.3115/1117794.1117802. URL: https://doi.org/10.3115/1117794.1117802.

Tsuruoka, Yoshimasa, Jun'ichi Tsujii, and Sophia Ananiadou (2009). "Stochastic Gradient Descent Training for L1-regularized Log-linear Models with

Cumulative Penalty". In: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*. ACL '09. Suntec, Singapore: Association for Computational Linguistics, pp. 477–485. ISBN: 978-1-932432-45-9. URL: http://dl.acm.org/citation.cfm?id=1687878.1687946.

Wang, Jianhong et al. (2016). "Extracting Clinical entities and their assertions from Chinese Electronic Medical Records Based on Machine Learning". In: *3rd International Conference on Materials Engineering, Manufacturing Technology and Control (ICMEMTC 2016)*.

Whitelaw, Casey and Jon Patrick (2003). "Named Entity Recognition Using a Character-based Probabilistic Approach". In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*. CONLL '03. Edmonton, Canada: Association for Computational Linguistics, pp. 196–199. DOI: 10.3115/1119176.1119208. URL: https://doi.org/10.3115/1119176.1119208.

Xu, Ke, Yunqing Xia, and Chin-Hui Lee (2015). "Tweet Normalization with Syllables". In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China: Association for Computational Linguistics, pp. 920–928. URL: http://www.aclweb.org/anthology/P15-1089.

Yao, Yao et al. (2017). "Sensing spatial distribution of urban land use by integrating points-of-interest and Google Word2Vec model". In: *International Journal of Geographical Information Science* 31.4, pp. 825–848. DOI: 10.1080/13658816.2016.1244608. eprint: https://doi.org/10.1080/13658816.2016.1244608. URL: https://doi.org/10.1080/13658816.2016.1244608.

Yolchuyeva, Sevinj, Géza Németh, and Bálint Gyires-Tóth (2018). "Text normalization with convolutional neural networks". In: *International Journal of Speech Technology* 21.3, pp. 589–600. ISSN: 1572-8110. DOI: 10.1007/s10772-018-9521-x. URL: https://doi.org/10.1007/s10772-018-9521-x.

Yule, George (2010). *The Study of Language*. 4th ed. Cambridge University Press. DOI: 10.1017/CBO9780511757754.

Zhao, Zhehuan et al. (2017). "Disease named entity recognition from biomedical literature using a novel convolutional neural network". In: *BMC medical genomics* 10.5, p. 73.

Domain adaptation and model combination for the annotation of multi-source, multi-domain texts

The increasing mass of User-Generated Content (UGC) on the Internet means that people are now willing to comment, edit or share their opinions on different topics. This content is now the main ressource for sentiment analysis on the Internet.

Due to abbreviations, noise, spelling errors and all other problems with UGC, traditional Natural Language Processing (NLP) tools, including Named Entity Recognizers and part-of-speech (POS) taggers, perform poorly when compared to their usual results on canonical text (Ritter et al., 2011).

This thesis deals with Named Entity Recognition (NER) on some User-Generated Content (UGC). We have created an evaluation dataset including multi-domain and multi-sources texts. We then developed a Conditional Random Fields (CRFs) model trained on User-Generated Content (UGC).

In order to improve NER results in this context, we first developed a POS-tagger on UGC and used the predicted POS tags as a feature in the CRFs model. To turn UGC into canonical text, we also developed a normalization model using neural networks to propose a correct form for Non-Standard Words (NSW) in the UGC.

Keywords: domain adaptation, named entity recognition, machine learning, conditional random fields, neural networks

Adaptation au domaine et combinaison de modèles pour l'annotation de textes multi-sources et multi-domaines

Internet propose aujourd'hui aux utilisateurs de services en ligne de commenter, d'éditer et de partager leurs points de vue sur différents sujets de discussion. Ce type de contenu est maintenant devenu la ressource principale pour les analyses d'opinions sur Internet. Néanmoins, à cause des abréviations, du bruit, des fautes d'orthographe et toutes autres sortes de problèmes, les outils de traitements automatiques des langues, y compris les reconnaisseurs d'entités nommées et les étiqueteurs automatiques morpho-syntaxiques, ont des performances plus faibles que sur les textes bien-formés (Ritter et al., 2011).

Cette thèse a pour objet la reconnaissance d'entités nommées sur les contenus générés par les utilisateurs sur Internet. Nous avons établi un corpus d'évaluation avec des textes multi-sources et multi-domaines. Ensuite, nous avons développé un modèle de champs conditionnels aléatoires, entrainé sur un corpus annoté provenant des contenus générés par les utilisateurs.

Dans le but d'améliorer les résultats de la reconnaissance d'entités nommées, nous avons d'abord développé un étiqueteur morpho-syntaxique sur les contenus générés par les utilisateurs et nous avons utilisé les étiquettes prédites comme un attribut du modèle des champs conditionels aléatoire. Enfin, pour transformer les contenus générés par les utilisateurs en textes bien-formés, nous avons développé un modèle de normalisation lexicale basé sur des réseaux de neurones pour proposer une forme correcte pour les mots non-standard.

Mots-clés : adaptation au domaine, reconnaissance des entités nommées, apprentissage automatique, champs aléatoires conditionnels, réseaux de neurones

Ecole doctorale ED268, Langage et Langues, Université Sorbonne Nouvelle
Maison de la recherche, Bureau A006, 4 rue des Irlandais, 75005 PARIS