Improved Named Entity Recognition Through SVM-Based Combination

Vincent Labatut

Improved Named Entity Recognition Through SVM-Based Combination

Vincent Labatut
vlabatut@gsu.edu.tr
Computer Science Department, Galatasaray University, Istanbul, Turkey

11/12/2013

Abstract

Named Entity Extraction (NER) consists in identifying specific textual expressions, which represent various types of concepts: persons, locations, organizations, etc. It is an important part of natural language processing, because it is often used when building more advanced text-based tools, especially in the context of information extraction. Consequently, many NER tools are now available, designed to handle various sorts of texts, languages and entity types. A recent study on biographical texts showed the overall indices used to assess the performance of these tools hide the fact they can behave rather differently depending on the textual context, and could actually be complementary. In this work, we check this assumption by proposing two methods allowing to combine several NER tools: one relies on a voting process and the other is SVM-based. Both take advantage of a global text feature to guide the combination process. We extend an existing corpus to provide enough data for training and testing. We implement an open source flexible platform aiming at benchmarking NER tools. We apply our combination methods on a selection of NER tools, including state-of-the-art ones, as well as our custom tool specifically designed to process hyperlinked biographical texts. Our results show both proposed combination approaches outmatch the individual performance of all the considered standalone NER tools. Of the two, the SVM-based approach reaches the highest performance.

1 Introduction

Named entity recognition (NER) is generally considered as a subtask of information retrieval, consisting in identifying specific parts of a text for their semantic relation with some concepts of interest. The notion of named entity itself is ambiguous, in the sense it can be defined in several different ways [28]. The recognition task involves not only determining the position of the entity in the text, but also classifying it among several predefined entity types, such as Location, Organization or Person. The difficulty with NER comes from the fact one concept can take several textual forms, and one textual expression can be used to represent several different concepts.

NER is a an important and popular field, because it allows adding structure to textual data, which in turns is a prerequisite to build higher level information processing services [28]. Our work is directly related to this trait, since our long term goal is to take advantage of biographical articles to extract implicit social networks. NER constitutes a first step in this process, and we aim at using it to identify spatio-temporal events in this type of texts. More particularly, our focus is on the Wikipedia encyclopaedia, mainly for three reasons. First, it describes a very large number of persons and covers a wide spectrum of domains, which will allow us to explore different fields of activity (sports, arts, politics...). Second, it is available in many different languages, and we plan to use this to increase the quality of the extracted information. For instance, suppose we want to extract the network of members of the European parliament. Each one of them has a page on the English version of the

encyclopaedia, however it is very basic for most of them, with only a few lines. But the same article written in the native language of the representative is likely to be much more detailed. Third, a set of tools associated to Wikipedia, such as Freebase [8] or DBpedia [7], allow accessing a part of its content in a structured way, and even enriching it.

As shown in recent works [10, 38], the performance of a NER tool observed on a certain domain does not necessarily transpose well when applied to a different type of text. Most existing NER tools were trained or designed to handle news-related texts [21, 33, 15], so there is no guarantee they will be as good on biographical articles. This has been confirmed by our previous study comparing a selection of state-of-the-art NER tools, applied to a corpus of Wikipedia articles [5]. Their *overall* performances are lower than those obtained on classic corpora, and tend to be very similar from one tool to the other. However, a more thorough analysis reveals their behaviours are consistently different when considering separately various aspects of the performance. In particular, the entity type and article category have a strong effect on the performance of certain NER tools. Here, the category of an article corresponds to the area(s) of activity of the person it describes: arts, politics, sports, etc. (cf. Section 3.1). Differences in the number of entities and the distribution of entity types across categories partly explain the observed differences. But other factors intervene, such as the presence of various forms of ambiguities. For instance, artist biographies contain many artwork titles, which certain tools find difficult to handle. These findings are consistent with the explanations presented in the literature to justify the variations of performances between texts of different domains (newswires, emails, reports, etc.) [38].

Based on these observations, we make the assumption those NER tools can be considered as complementary, and we propose to combine them in order to improve the NER results. Our approach is based on the use of a Support Vector Machine (SVM), whose inputs are the outputs of standalone NER tools, as well as the article category, a global text feature we add to help the classification process. This approach is particularly relevant for us, because it is language independent, and it allows easily integrating additional metadata. Previous works have relied on such a stacking method, but we contribute further in the following ways. First, we use state-of-the-art existing NER tools as inputs, instead of custom-trained classifiers. This allows increasing the diversity of the behaviours to be combined by the SVM, and such diversity is considered to affect positively the performance of ensemble methods [26]. Second, we use a SVM to perform the combination, which was never done before in the context of NER, according to our knowledge. Existing combination-based methods usually rely on manually defined voting systems (cf. Section 5.1), which we use as a baseline to assess the performance of our own method. Third, we propose a new simple NER tool dedicated to the processing of Wikipedia articles, and study how it affects the combination process. Fourth, we significantly extend our corpus of annotated Wikipedia articles to evaluate all the mentioned NER tools. Fifth, we provide the community with a flexible open source Java platform for NER benchmarking, which can easily be extended to any NER tool or performance measure.

The rest of this article is organized as follows. In the next section, we review the standalone NER tools used as a basis for our combination process, and the measures used to assess their performance. In Section 3, we describe different variants of the proposed combination process, as well as our Wikipedia-specific NER tool. In Section 2.2, we introduce our corpus and evaluation platform, before presenting and commenting our results. Finally, Section 6 describes how our work could be extended, and presents some general perspectives.

2 Named Entity Extraction

In this section, we present the NER tools used in the rest of our work, as well as the methods designed to evaluate them. The latter point is of interest, because it is related to both the definition of our combination method (cf. Section 3.1) and to the comparison of NER tool performances given in Section 2.2.

2.1 Selected Tools

Selecting an appropriate NER tool is a difficult task, because they are numerous, and can rely on very different approaches. One can distinguish two families: rule-based approaches, which are based on the formal expression of linguistic knowledge, and machine-learning approaches, which use statistics to build models [27]. Note it is possible to combine both approaches, and most tools additionally integrate lists of words and expressions under the form of dictionaries or gazetteers.

For comparison purposes, we decided to apply the same tools we already used in our previous study [5]: Stanford Named Entity Recognizer [19], Illinois Named Entity Tagger [31], OpenCalais Web Service [36] and Alias-i LingPipe [3]. These were selected according to the following criteria: 1) publicly and freely available; 2) widespread use in the NER community; 3) can detect Location, Organization and Person entities; 4) able to process the English language. The first criterion is meant to exclude works not backed by any concrete software. The second one allows focusing on supposedly efficient tools. The third and fourth ones are related to the longer term objective of this work, which is to extract spatio-temporal events from Wikipedia biographical texts. In the present work, we added two other tools to this selection. First, Apache OpenNLP [4], because it fits these criteria. Second, Subee, which is a custom NER tool described later, in Section 3.2.

In the rest of this section, we present the selected third-party tools in more details. All of them are based on machine learning methods, possibly mixed with rules and/or dictionaries. Except OpenCalais, they are provided with one or several models, pre-trained on various corpora. Note most of these tools also allow training new models by using different corpora. We chose not to train them on our own corpus, because we wanted to study how well the predefined models perform when applied to texts of a different domain, and to test whether their combination improves the performance.

Stanford Named Entity Recognizer (SNER). This popular Java tool is based on linear chain conditional random fields [19]. It is provided with three predefined models for the English language, based on different corpora: CoNNL03 [33] for the first, MUC6 and MUC7 [21] for the second and all of them as well as ACE [15] for the third model. All three of them are able to recognize the targeted entity types (Location, Organization and Person) as well as others for the second model. Each of these three models exists in a plain version and in an augmented version, which includes distributional similarity features, i.e. additional data meant to improve performance. An additional option allows enabling or disabling case sensitivity. It is possible to train new models, possibly by taking advantage of existing dictionaries.

Illinois Named Entity Tagger (INET). This Java tool is based on several supervised learning methods: hidden Markov models, multi-layered neural networks and other statistical methods [31]. It also uses manually annotated dictionaries for lookup, and word clusters generated from unlabelled text to improve performance. A few word clusters and dictionaries are distributed with the tool, and it is possible to build new ones. Word clusters, models, output encoding schemes can be configured via a specific file. The tool is provided with two models trained on English texts from the CoNLL03 [33] and OntoNotes [23] corpora, respectively. Both can detect Location, Organization, Person entities, as well as other types. INET additionally allows training new models.

OpenCalais Web Service (OCWS). This tool takes the unusual form of a Web service [36]. Its use is free of charge, and a public API is available for developers. However, because it is a closed source commercial product, the nature of the internal processing it performs is unknown to us, and neither is the nature of the data used for its training. It can process English, French or Spanish raw or structured (XML/HTML) text. It supports 39 different types of entities, some of which are subsumed by the ones we target. For this reason, we associate several OCWS entity types to the same targeted type. The Person type is treated as such. A Location can be one amongst City, Continent, Country, ProvinceOrState and Region. An Organization can be of the OCWS types Company, MusicGroup or Organization. Note OCWS is able to perform other NLP-related tasks besides NER.

Alias-i LingPipe (AILP). Like OCWS, this software is commercial and can handle various other NLP tasks besides NER [3]. But this Java tool is open source, and a free license is available for academic use. It relies on n-gram character language models, trained through hidden Markov models and conditional random field methods. Three different models are provided for the English language. Two of them are dedicated to genetics-related texts, and are therefore of little interest to us. The third is built on the MUC6 corpus [21] and can detect Organization, Location and Person entities. Many aspects of the process, such as the chunking method, can be controlled via a configuration file.

Apache OpenNLP (APON). This Java open source tool developed by the Apache Foundation can handle various NLP-related tasks, including NER [4]. Entity detection is performed through a maximum entropy classifier, able to handle various features as inputs (n-gram, dictionary, etc.). Many models are proposed, dealing with 7 European languages including English, and focusing on specific entity types, including the ones we target (Location, Organization, Person). New models can be trained, too, using various combination of predefined or custom features.

2.2 Evaluation Methods

For a given text, the output of a NER tool is a list of entities and their associated types, and the ground truth takes the exact same form. In order to assess the tool performance, one basically wants to compare both lists. The most widespread approaches are based on the classic *Precision* and *Recall* measures. Let us first define the terms TP, FP and FN, corresponding to the numbers of *True Positives*, *False Positives* and *False Negatives*, respectively. A TP is an entity which was correctly detected. A FP corresponds to the case were the NER tool detects an entity where there is none. A FN occurs when an entity is not detected by the tool. The Precision and Recall measures can then be defined as $Pre = TP/(TP + FP)$ and $Rec = TP/(TP + FN)$, respectively. The Precision can be interpreted as the proportion of detected entities which are correct, whereas the Recall is the proportion of real entities which were correctly detected. Some authors prefer to use a single score, obtained by taking their harmonic mean, also called F-measure: $F = 2(Pre \cdot Rec)/(Pre + Rec)$.

However, different variants exist, depending on the goal and context of the NER task [29]. They differ mainly on how they define what they consider to be a correct entity. Concretely, these differences mainly concern two points: 1) how they handle the two aspects of entities (position and type) and 2) how they penalize partial matches. First, the correctness of an estimated entity can be assessed both spatially and typically. We consider it to be *spatially* correct if the NER tool managed to identify its position in the text. We say it is *typically* correct if the entity type assigned by the NER tool corresponds to the real one. Second, some evaluation approaches require to identify the exact position of the entity in the text (exact spatial match), whereas others reward even partial matches, i.e. a situation where the estimated entity spatially intersects with the actual one.

The simplest, and most widespread variant, is to consider an entity is a true positive if and only if both its position and type are exact. This is the case, for example, of the method adopted for the MUC conference series [21], which additionally considers only exact matches. It takes the form of a single performance measure, combining both typical and spatial aspects at once. On the contrary, in our previous work [5], we wanted to make a detailed comparative study of the performances of NER tools on biographical texts, so we adopted the opposite approach. Not only did we consider separately the spatial and typical aspects, but we also examined the Precision, Recall and F-measure independently for each type, as well as for each article category.

Our context in this article is different, because of our longer-term objective, which is to identify spatio-temporal events in a collection of texts. First, we are not interested in the precise limits of an entity in the sentence containing it, but rather by its simple presence and approximate position. For this reason, we consider partial spatial matches as very acceptable results. Second, when assessing the performance of the tested NER tools, we do not need to consider independently each entity type and article category anymore (this level of detail is superfluous). It is still interesting to consider separately position and type, though: if an entity is correctly detected, but with a wrong type, some post-process

might allow correcting the mistake. Note that, independently from performance assessment, we still use some of the detailed measures from [5]: these are considered as weights in the context of various voting processes described in Section 3.1.

In summary, when evaluating the NER tools in Section 4.2, we consider a detected entity to be a true positive even if it only partially matches the actual entity. Moreover, we consider position and type separately, in order to be able to clearly assess both these aspects of the NER performance. As an example, let us consider the following sentence: Nelson Mandela was born into the Madiba clan in Mvezo. Suppose the evaluated tool detects Mandela as a person and Mvezo as a location. Then, Mvezo is both a spatial and a typical true positive. However, the type of the estimated entity Mandela is correct, but its location only partially matches the actual entity, which would be Nelson Mandela. Still, as stated before, we consider it to be both a spatial and a typical match, too. Finally, the evaluated tool failed to identify Madiba clan at all, so this is both a spatial and a typical false negative.

3 Proposed Approach

We propose to modify the NER process in two distinct ways, in order to improve its overall performance when dealing with biographical texts. The first point consists in combining several standalone NER tools; we describe two main methods to perform this combination, with several variants. The second point is the definition of a new NER tool able to take advantage of the information encoded in Web pages under the form of hyperlinks.

3.1 Combination Process

The idea of combining several standalone NER tools is based on the observations made in our previous study [5], which highlight the fact some tools are better with certain entity types or article categories than others. Hence our assumption that an appropriate combination of their outputs might lead to an improved overall performance. We propose two ways of doing so: a naive method based on a voting system, and an SVM-based approach we expect to be more efficient. In this section, we first describe two pre-processing steps which are necessary for certain variants of both methods, and then the methods themselves.

Entity Groups. For a given article, the first step is to apply separately each one of the standalone NER tools, resulting in distinct sets of estimated entities. Each entity is described by its starting and ending positions, expressed as character offsets relatively to the whole text, and by its type. Comparing the types of the entities outputted by the various NER tools is straightforward, since they are basically symbols. However, comparing their boundaries is much less direct, because entities can contain several words, and partially overlap. The classic BIO approach consists in first segmenting the text in chunks (words), and use an additional label to indicate if a chunk is at the Beginning, Inside or Outside an entity. However, this requires performing sequential labelling, in order to preserve the continuity of the entities. We propose an alternative approach, based on what we call entity groups.

We define an *entity group* as a set of entities detected by different NER tools, such that all those entities match. As explained in Section 2.2, we accept partial matching, which means entities belonging to the same group overlap at least, and match exactly at best. An entity group can be considered as a consensual estimation of an actual entity, with partial disagreement between NER tools regarding the exact position and/or the type. For instance, let us consider again our example Nelson Mandela was born into the Madiba clan in Mvezo. Suppose we have 3 NER tools, considering respectively Madiba, Madiba clan and the Madiba clan as entities. Then this set of entities constitutes an entity group. The role of the combination methods is to treat each group, in order to get a final result regarding the existence of the supposed entity, and its position and type (if appropriate).

Article Categories. Because of the previously mentioned heterogeneity observed in the behaviour of the selected tools, it is necessary to take some context into account in order to get reasonably good results. In [5], the article categories were identified manually, which was appropriate since the goal was only to study how NER performance was influenced by this factor. But in the present work, the category is directly used during the NER process, so it must be identified in a non-supervised way. For this purpose, we defined a two-fold automatic method. It relies on 12 predefined categories, described in Table 1. Note we use twice as many categories as in our previous work, in order to perform a finer characterization of the articles. Moreover, categories are not mutually exclusive anymore, for the same reason.

Table 1: Categories of Articles

Academia	Architecture	Art	Business
Law	Medicine	Media	Military
Politics	Religion	Sports	Other

The first step in our process consists in analysing the first sentence of the considered article. In all Wikipedia biographies, it takes the following form: <First name Last name> (<Birth Date> – <Death Date>) is/was a <Occupation>. We defined a mapping from textual expressions describing occupations to our categories. An article is assigned to all the categories whose associated occupations appear in its first sentence. Although rare, it is possible that no category can be found at all, for instance because the concerned occupation is not in our map, or because the first sentence is exceptionally structured differently. In this case, we switch to our second step, which consists in looking up the considered Wikipedia page on Freebase [8]. Freebase is an online collaborative database containing data gathered from various public sources. What interests us in Freebase, is that each Wikipedia page is described in a structured way, including a set of normalized concepts. More particularly, some of these concepts are related to occupations. We therefore defined a second map aiming at converting a subset of Freebase concepts to our categories. If, for some reason, this second step also fails, then the article is associated only to the category *Others*. Note this category can also be assigned when the occupation could be identified during one of the two steps, but does not fit in our article classification.

Vote-based Method. With this combination method, our goal was to define a simple process to be used later as some kind of baseline, when assessing the performance of the SVM-based approach. It is constituted of three successive votes aiming at deciding of a given entity existence, location and type, respectively. Each entity group identified during the pre-processing step is processed iteratively.

During the first vote, each tool represented in the group is considered to vote for the existence of the entity, whereas each tool not represented votes against. We say a tool is *represented* in a group if one of the entities in this group was detected by this tool. If the vote leads to the conclusion the entity does not exist, the process ends here for the considered group. Otherwise, it continues with both other votes. The second vote aims at solving disagreements regarding the exact position of the entity. Each represented tool votes for the starting and ending positions it detected. Similarly, in the final vote, each represented tool votes for the type it detected. The majority positions and type are considered as the most consensual ones, and are outputted by our combining algorithm.

The simplest voting approach, uniform voting, consists in giving the same importance to each tool. We alternatively considered using different sets of weights during the voting process, in order to reflect the relative quality of the tools. These weights are based on the performance measures defined in [5] and presented in Section 2.2. We process them over our training set (cf. Section 4.1 for a description of our corpus). In the first vote (existence of the entity) we use the spatial Precision to weight the *for* votes, because a high value means few false positives, so one can trust such a tool when it detects an entity. On the contrary, we used the spatial Recall to weight the *against* votes, since a high value means few false negatives, so one can trust such a tool when it does not detect an

entity. We consider both partial and exact matches in both measures. For the second vote (entity position), we used the same principle, but this time with the spatial Precision and Recall processed using only exact matches. For the third vote (entity type), we used the typical Precision and Recall (i.e. processed by comparing only types, and not positions).

The article category was introduced in our voting process by considering one distinct weight for each category. Those are obtained by processing the previously mentioned Precision and Recall variants not for the whole training set, but only for subsets of articles from the considered category. Since we have 12 categories, 2 different possible weights for each vote (for or against) and 3 distinct voting steps, this amounts to a total of 72 category-wise weights for each tool. For comparison purposes, the results obtained with the three types of weights are commented in Section 4.2: uniform, overall and category-wise weights.

SVM-based Method. The voting process described in the previous section can be seen as a raw approximation of the optimal combination of NER tools. To get a better estimation, we alternatively propose training a multiclass SVM to perform the same task. It is possible to encode the information fetched to the SVM in various ways. We propose two partially different approaches, to be compared later in Section 4.

The first approach is relatively close to the voting process, in the sense the data is considered through the entity groups defined during the pre-processing step. For each NER tool, we use as many binary inputs as there are different entity types, so 3 in the case of this work (Organization, Location, Person). For a given entity, the input corresponding to the type estimated by the considered tool is set to $+1$, whereas the other inputs take the value -1. Otherwise, if the tool did not detect the entity at all, all inputs are set to -1. To encode article categories, we use as many inputs as there are categories, i.e. 12 in our case. Each category associated to the considered article receives the value $+1$, whereas the others are all -1. We combine 6 different NER tools if we count our own one (cf. Section 3.2), so our SVM has 30 inputs in total. The output is an integer among 4 different values, representing either the absence of any entity, or its type if an entity is deemed present. One may notice this method allows estimating both the existence and type of an entity, but not its exact position. For this matter, we proceed similarly to the voting-based approach: the NER tools vote for a starting and an ending position, and the majority ones are kept.

The limitation of this approach is that the SVM determines only the presence and type of the entities, but not their location. Therefore, we propose a second SVM approach which takes advantage of the BIO method described in Section 3.1 to perform this additional task. This time, the text is first segmented to get its constitutive chunks (word), which are then fetched to the SVM. To represent the BIO labels, we need 2 supplementary inputs for each tool, in addition to those already defined for the first approach. A word at the beginning or inside an entity is represented by the value $+1$ in one of these additional inputs, whereas a word located outside corresponds to -1 for both inputs. Like in the first approach, a specific value is used to output an absence of entity. To represent the fact a word belongs to an entity, we use two distinct values for each entity type: one for Beginning and one for Inside. So, we have a total of 7 possible output values. However, this process treats each word separately, which could have an impact on performance. To solve this, we propose another variant of this approach, in which the SVM is additionally fed the type and BIO label of the previous chunk. This is meant to help the tool handling entities spanning multiple words.

3.2 Simple URL-based Entity Extractor

We propose a very simple NER tool, called Subee (Simple URL-based Entity Extractor), for two reasons: to take advantage of a type of information ignored by traditional algorithms, and to highlight the fact a NER tool which is not very efficient on its own can still bring some performance improvement in the context of our combination process.

The particularity of the texts we process, besides the fact they are biographies, is that they are Wikipedia articles, hence Web pages. We therefore have access not only to the text itself, but also

to some additional information encoded using the HTML language. We are particularly interested by the hyperlinks, especially those located in the article itself (by opposition to those often listed as external references at the end of Wikipedia pages). These links have two characteristics which are very relevant to us: first, they are manually defined by the authors or editors of the article, so they can be considered as valuable manual annotations; second, they highlight objects of interest, which most often include named entities.

Traditional NER tools are designed to process plain text, so by definition they ignore this useful information. Our tool simply parse the HTML source code and analyses the existing hyperlinks. We focus only on the internal URLs, i.e. those directed at other Wikipedia pages. Note almost all hyperlinks found in an article body are internal. We then perform a first filtering, by discarding links whose label (i.e. textual content) does not match some typographical criteria. This text must not be purely numerical (to avoid dates) and must start with a capital letter (to focus on proper names). The remaining labels are considered as potential entities, and an additional test allows confirming or invalidating this status. For this matter, we first retrieve the URL pointed by the link, and fetch it to Freebase (cf. Section 3.1 for a description of Freebase). Freebase uses specific concepts to indicate the Wikipedia page corresponds to an organization, a location or a person (among others). It is therefore possible to simultaneously check if the expression is an entity and to retrieve its type when it is the case.

This process allows discarding most of the false positives, however it tends to be confused by demonyms. A *denomyn* is an adjective relative to a place, or the name given to its inhabitants. For instance, the demonyms of SPAIN are SPANISH and SPANIARD. In Wikipedia articles, most demonyms are hyperlinks pointing at the concerned place, and for this reason, our method mistakes them for location entities. To prevent this, we constituted a list of unambiguous demonyms and use it to discard any matching label.

Another issue is the high number of false negatives: a lot of entities are not associated to hyperlinks, and are missed by our method. However, one can remark that, when writing a Wikipedia article, the convention is to define a hyperlink only for the first occurrence of the concerned object [2]. An easy way of solving the problem therefore consists in looking for the other occurrences of the same expression in the rest of the text, and consider them as the same entity (if the original expression is supposed to be an entity itself, obviously). We also take advantage of another Wikipedia convention regarding the use of acronyms. The first time such an entity appears, its name is generally stated in full, followed by the corresponding acronym in parenthesis [1]. We look for occurrences of both the full name and acronym in the rest of the text. Finally, the last improvement concerns the name of the person described by the article of interest. The name of the subject of a biography appears many times, but never as a hyperlink, obviously (otherwise, the Web page would be pointing at itself). Hopefully, the title of the article corresponds to this name, so we can still identify the associated entities by searching for various combinations of last names and first names or initials in the text.

4 Experimental Evaluation

In this section, we first briefly present our platform and corpus. We then describe and comment the results obtained with the selected NER tools used independently and combined through the various proposed methods.

4.1 Corpus and Implementation

The corpora traditionally used to evaluate NER tools are mainly based on newswire texts: MUC6 [21], CoNLL [33], ACE [15], OntoNotes [23], etc. Several studies showed these are not reliable, in the sense the performance NER tools obtain on these data does not hold on different datasets. This was observed not only for datasets of different text types, such as emails or speech transcriptions [38], but also for datasets of the exact same type [10, 38]. Note the same observation was made for languages

other than English, such as Chinese [22]. Moreover, Balasuriya *et al.* showed standard NER tools performance is lower on Wikipedia texts [6], which highlights the need for a specific corpus.

For these reasons, we decided to constitute our own corpus. We chose to define it manually, and not to use an automatically method such as the one described in [30], in order to closely fit our needs and control precisely the meaning of the annotated entities. The corpus was initially created by randomly picking articles, but a bias was later introduced towards the categories constituting our main interests in terms of future applications (social network extraction), as shown in Table 2. Note one article can have several categories, since one person can be active in several domains during his life. The selected articles were initially annotated by two different persons, then a second pass was performed by a third member of our team, and a third pass was finally performed collectively. This corpus was used in our previous work [5], but since then, we extended it from 247 to 408 articles (only Wikipedia biographies). This approximately amounts to $500,000$ words and $45,000$ Location, Organization and Person entities. The revised and completed corpus is freely available online[1], including the original Wikipedia pages.

Table 2: Distribution of Categories in the Corpus

Academia	98	Architecture	13
Art	79	Business	13
Law	53	Medicine	6
Media	43	Military	20
Politics	176	Religion	5
Sports	31	Other	10

In order to implement our propositions, we used Nerwip, the Java platform we created for our previous work [5]. The original version already contained classes allowing to apply most of the selected tools and assess their performances. We completed it with the ability to handle OpenNLP, process the MUC performance measures, and implemented our Subee NER tool and our two combination methods in a compatible way. For the SVM-based method, we used the widespread LIBSVM library [9], with a RBF kernel. LIBSVM uses the one-against-one strategy for multiclass problems, i.e. one SVM is trained to discriminate between each pair of classes. The different NER tools can be applied in a parallel way, which means our platform is as fast as the slower tool: the combination process itself is extremely fast, and the processing time it requires is negligible when compared to the NER tools. The complete platform is open source and freely available online[2]. It is flexible enough to allow integrating any other NER tool and performance measure, therefore it constitutes a good basis to benchmark NER tools (be it on Wikipedia texts or not).

4.2 Results and Discussion

We first focus on the performances of the standalone tools, before considering the combination methods. Two-thirds of the corpus were used for training the combination methods, the remaining third was used for testing all tools. Note the third-party tools were not trained on those data. When several models or parameters were available, we selected the one giving the best results obtained on the training set, in terms of F-measure.

Standalone Tools. We start with the 5 third-party tools. Table 3 shows the Precision, Recall and F-measure obtained on the testing set. As expected, the performances are significantly lower than what can be observed in the literature when the same algorithms are applied on the corpora for which they were trained. According to the F-measure, INET and SNER are the best tools, with OCWS as a close third. AILP and APON are behind, especially in terms of typical performance.

[1] http://dx.doi.org/10.6084/m9.figshare.1289791
[2] https://github.com/CompNet/Nerwip

Table 3: Performances of the third-party tools

	Best Previous Model						Nerwip Model					
	Position			Type			Position			Type		
Tool	Pre	Rec	F	Pre	Rec	F	Pre	Rec	F	Pre	Rec	F
AILP	0.78	0.86	0.82	0.52	0.57	0.54	0.88	0.82	0.85	0.79	0.73	0.76
APON	0.89	0.48	0.62	0.73	0.40	0.52	0.78	0.70	0.73	0.59	0.53	0.56
SNER	0.87	0.88	0.88	0.69	0.69	0.69	0.90	0.91	0.90	0.81	0.82	0.82
INET	0.88	0.88	0.88	0.75	0.74	0.75	0.94	0.89	0.91	0.87	0.83	0.85
OCWS	0.93	0.76	0.84	0.76	0.61	0.68	–	–	–	–	–	–

Table 4 presents the performances obtained by Subee. We assessed its performance for all combinations of the different features described in Section 3.2, but we present only the selection of the obtained results, for space matters. The first row corresponds to the simplest variant, consisting in using only hyperlinks. As expected, the Precision values are high, but the Recall values are low due to the large number of false negatives: remember only the first occurrence of an entity is hyperlinked in Wikipedia articles (cf. Section 3.2). In the second row, the search for additional occurrences is enabled, allowing to decrease the number of false negatives while keeping a relatively high Precision. In the third row, we add the processing of the article title, aiming at looking for entities corresponding to the person which constitutes the topic of the article. This slightly increases the Precision, and the increase in Recall is even higher. The fourth row displays the performances obtained when additionally enabling the processing of acronyms, which has mainly a positive effect on type processing. Finally, the last feature is the processing of demonyms, which slightly improves Precision for both location and type. Subee displays the best performance in terms of spatial Precision, which confirms the assumption that many hyperlinks correspond to entities. However, its overall performance is largely decreased by its low Recall value, which means a large number of entities are not associated to hyperlinks at all.

Table 4: Performances of Subee

				Position			Type		
Occurrences	**Title**	**Acronyms**	**Demonyms**	Pre	Rec	F	Pre	Rec	F
×	×	×	×	0.95	0.26	0.41	0.76	0.21	0.33
✓	×	×	×	0.92	0.38	0.54	0.72	0.30	0.42
✓	✓	×	×	0.94	0.58	0.72	0.80	0.49	0.61
✓	✓	✓	×	0.94	0.58	0.72	0.83	0.52	0.64
✓	✓	✓	✓	0.96	0.58	0.72	0.84	0.52	0.64

Relatively to the other standalone (third-party) tools, Subee is clearly in the lower tier, just above APON. It is worth noticing our goal with Subee was precisely to define a tool reaching a high Precision, in order to see if a combination method was able to take advantage of this characteristic by using such an input when appropriate and discarding it the rest of the time. So, the low Recall, even if not desirable, was expected and does not constitute a problem in the context of our work.

Combination Methods. Let us now comment the results obtained when combining the NER tools. Table 5 presents the performance obtained for the vote-based approach. The first column represents the type of weights: uniform, overall (processed over the whole corpus) or category-wise. The second column indicates whether Subee was used or not.

It is worth noticing that, independently from the type of combination, the obtained F-measure are always higher than those of the best standalone tools. In terms of Precision, the vote-based approach is second only to Subee, and in terms of Recall, it is just behind INET for locations and reaches the best score for types. Another important observation is that the different variants of this approach all

have very similar performances. The types of weights we tested do not have any observable effect on any measure, while the use of Subee positively affects the performance, but only marginally.

Table 5: Overall Performances of the Vote-Based Approach

		Best Previous Model						**Nerwip Model**					
		Position			**Type**			**Position**			**Type**		
Weights	**Subee**	Pre	Rec	F	Pre	Rec	F	Pre	Rec	F	Pre	Rec	F
Uniform	×	0.94	0.84	0.89	0.80	0.72	0.76	0.94	0.91	0.93	0.86	0.84	0.85
Overall	×	0.94	0.84	0.89	0.80	0.72	0.76	0.94	0.91	0.93	0.87	0.84	0.86
Category	×	0.94	0.84	0.89	0.80	0.72	0.76	0.94	0.91	0.93	0.87	0.84	0.86
Uniform	✓	0.94	0.87	0.90	0.81	0.76	0.78	0.94	0.93	0.93	0.86	0.85	0.86
Overall	✓	0.94	0.87	0.90	0.81	0.75	0.78	0.94	0.92	0.93	0.87	0.85	0.86
Category	✓	0.94	0.87	0.90	0.81	0.75	0.78	0.94	0.92	0.93	0.87	0.85	0.86

Table 6 presents the results of the SVM-based approach. Again, we did not include all possible combinations of features in the table, we instead focused on the most interesting results. The first column (Position) corresponds to the method used to determine the position of the entities: entity groups with uniform weights (Uniform), with overall weights (Overall), with category-wise weights (Category), chunking and considering only the current word (Current), and finally chunking and considering both current and previous words (Previous). The second column (Subee) indicates if Subee was used, and the third one (Categories) shows whether or not article categories were used as an additional input of the SVM.

The best version of our SVM-based approach equals or outperforms all the individual tools, for all three measures (Precision, Recall and F-measure) in both performance aspects (entity position and type). It also outperforms the best variant of our vote-based method, especially regarding type detection. Like for the vote-based approach, we observe no effect from the type of weights used to combine entity groups (three first rows of the table). Using the BIO approach instead does not bring clearly better results (rows 4 and 5). When using only the current word, it is even slightly inferior, whereas using both the current and previous words leads to results comparable to those obtained with the entity group approach. However, the performance of the BIO method is improved when using Subee (row 6), which is not the case when positions are decided based on entity groups. Finally, including categories in the SVM inputs (row 7) allows slightly improving all measures, especially Recall, which confirms the interest of using such a global feature.

Table 6: Overall Performances of the SVM-Based Approach

			Best Previous Model						**Nerwip Model**					
			Position			**Type**			**Position**			**Type**		
Position	**Subee**	**Categories**	Pre	Rec	F	Pre	Rec	F	Pre	Rec	F	Pre	Rec	F
Uniform	×	×	0.94	0.85	0.89	0.82	0.74	0.78	0.93	0.92	0.93	0.87	0.85	0.86
Overall	×	×	0.94	0.85	0.89	0.82	0.74	0.78	0.93	0.92	0.93	0.87	0.85	0.86
Category	×	×	0.94	0.85	0.89	0.82	0.74	0.78	0.93	0.92	0.92	0.87	0.85	0.86
Category	✓	×	0.94	0.86	0.90	0.83	0.75	0.79	0.93	0.91	0.92	0.87	0.85	0.86
Category	✓	✓	0.94	0.87	0.90	0.83	0.77	0.80	0.93	0.92	0.92	0.87	0.85	0.86
Current	×	×	0.92	0.86	0.89	0.80	0.75	0.77	0.92	0.91	0.92	0.85	0.85	0.85
Previous	×	×	0.95	0.83	0.89	0.82	0.73	0.77	0.93	0.92	0.92	0.86	0.85	0.86
Previous	✓	×	0.96	0.85	0.90	0.84	0.75	0.79	0.93	0.91	0.92	0.87	0.85	0.86
Previous	✓	✓	0.96	0.87	0.91	0.85	0.77	0.81	0.93	0.91	0.92	0.87	0.85	0.86

Instead of considering the performance at the level of the whole testing set, it is also possible to study it by category or by type. In other words, one can process the same measures by focusing

only on a certain category or type. In our previous study, we used this approach to show NER tools do not behave similarly even if their overall performances were very close [5]. In other terms, their performances are not homogeneous: some tools are better on certain types or categories. An interesting effect of the combination of NER tools is that the performances observed at the level of the types or categories are smoothed compared to those of the standalone NER tools. The overall performance is increased because the combination allows to make it more homogeneous over entity types and article categories.

5 Related Works

Our work is related to two topics of the NER community. First, the use of several classifiers to improve NER performance, since we propose several methods to combine the outputs of multiple NER tools. Second, works taking advantage of Wikipedia, because we focus on the analysis of Wikipedia articles, use it directly to define our own NER tool Subee (Section 3.2) and take advantage of Freebase (which is largely based on Wikipedia) to identify entity types.

5.1 NER Tools Combination

Combining several classifiers to improve the overall performance is a popular strategy used in the machine learning domain. Since NER can be described as a classification problem, it is not surprising to see the same approach was applied to solve this problem, in various ways.

In certain works, an additional classifier is used to post-process the output of a single NER-trained classifier, and improve its results. For instance, in [11], Collins compares the use of a boosting and a perceptron classifiers, when applied to the output of a single maximum-entropy tagger (as well as additional features), in order to increase the spatial accuracy of entity detection. By comparison, we also use an additional feature (article category), but our primary goal is to combine several NER tools.

Some works rely on ensemble methods to combine the outputs of multiple instances of the same learner. In [34], Szarvas *et al.* apply boosting to the C4.5 decision tree learning algorithm. Five different trees are trained on various sets of text features, and combined with a $3/5$ qualified majority vote. The method of Lucarelli *et al.* is a composite approach relying on 4 consecutive analyses of the text. The two last ones are performed by two pairs of combined SVMs, whose training is performed through active learning (the algorithm requires manual annotation for instances likely to improve its performance). Among other points, our work differs mainly with these approaches in the fact our objective is to combine very heterogeneous NER tools.

Other works study the combination of several different classifiers through various voting mechanisms. Some approaches assign the same weight to all considered classifiers (uniform voting), and keep the majority decision as in [25]. Other approaches proceed similarly, but use a qualified majority instead [18]. In the approach proposed by Dimililer *et al.* [14], a classifier is not always allowed to vote, depending on the entity type. This classifier selection is performed through a genetic algorithm applied on the training data. In other cases, classifier importance can vary in the voting process. In [35], Thao *et al.* use the Precision measure as a vote weight to combine the output of heterogeneous classifiers. They consider the overall measure obtained on their training set, and they also propose to modulate the weights depending on the detected entity type. Florian *et al.* focus on statistical classifiers only [20]. This allows them taking advantage of the probability distributions they produce, in order to estimate specific vote weights for each classifier and type. Some authors estimate those weights through optimization methods, such as genetic algorithms [17] or simulated annealing [16]. By comparison, we also propose to use various voting mechanisms rather similar to those used in the literature (cf. Section 3.1). One difference is the use of an additional feature (article category) to modulate the vote weights. Moreover, we propose another approach fully based on SVM, which does not rely on the votes (at least in the above sense), since SVMLIB uses the one-against-one strategy to handle multiclass problems (cf. Section 4.1).

Some authors rely on procedural approaches rather than weights. In [40], an *ad hoc* rule-based procedure is designed to combine the outputs of several statistical classifiers trained on Chinese text, leading to better results than with uniform voting. In [39], Vlachos trains two different classifiers to recognize biomedical entities such as genes. He notices one classifier is better to detect new entities, i.e. entities absent from the training set. Our own SVM-based method is related to these procedural approaches. However, instead of designing rules manually, it is automatically trained from the data. Vlachos' idea of using additional information to improve the combination can also be related to the use of categories in both our voting and SVM-based methods. However, the way this information is integrated to the combining process is completely different.

Our method differ from all the cited works by other aspects. First, we mostly use already existing NER tools, when all other approaches rely only on custom tools, taking the form of classifiers trained for NER. This generally means those classifiers are all fetched the same features and are trained on the exact same data. This is likely to decrease the diversity of their behaviour, which in turns tends to lower the overall performance when applying ensemble approaches. On the contrary, the tools we used are provided with various models obtained on various corpora (or combinations of corpora), rely on various processing principles (both rule-based and machine learning), and use various text features. Second, we include a global feature in the combination process (article category), when existing approaches consider only the classifiers outputs.

5.2 Wikipedia-Based Approaches

There are strong links of various natures between NER and Wikipedia (WP). In [37], Toral & Munoz consider this encyclopaedia as a source to automatically extract gazetteers and dictionaries to be used during a NER task. For a given word or expression, they analyse the text of the corresponding WP page, in order to determine if it is a named entity, as well as to identify its type. It is worth noticing they particularly focus on the first sentence of the article, as we did to automatically derive the category of an article (cf. Section 3.1).

Other works use WP to deliver additional features to classifiers trained for NER. When processing the text of interest, candidate expressions are identified, corresponding to potential entities. Various WP-based calculations are then performed to characterize these expressions and improve the NER process. The method proposed by Kazama & Torisawa [24] focuses exclusively on the first sentence of the article possibly associated to a candidate expression. In [12], Cucerzan uses not only the WP article itself, but also metadata such as redirection pages and WP article groups. Interestingly, to evaluate his method, he applies it to a set of WP pages, and adds WP hyperlinks to the detected entities. He then compares the result with the actual hyperlinks of the original WP articles. The assumption he implicitly makes is that manually defined hyperlinks appearing in WP articles can highlight named entities. Our NER tool Subee (cf. Section 3.2) is based on the same principle. However, its functioning is very different (and voluntarily much simpler), and its purpose as well, since we process only WP articles. This work is extended by Dakka & Cucerzan [13], under the form of a classifier able to determine the entity associated to a WP page.

Nothman *et al.* [30] propose to use WP to automatically constitute NER corpora for both training and testing. They first define a classifier to associate a single entity to each one of the considered WP pages, using again first sentences and article groups. Then, they take advantage of hyperlinks (as we did in Subee) to associate entity types to text expressions. A rule-based method allows them to identify subsequent occurrences of the same entity in the rest of the article (as mentioned in Section 3.2, hyperlinks are defined only for the first occurrence in WP articles). The main differences with Subee are the goal (we aim at defining a NER tool, not a corpus) and the fact we use Freebase to identify entity types.

A similar approach was designed by Richman & Schone [32] to generate multilingual corpora. They first analyse a collection of English WP pages, in order to identify their entity types. They then switch to another language, and proceed like Nothman *et al.*, with an extra step: when a hyperlink is detected in the processed text, they identify the targeted page and retrieve its English counterpart,

in order to use its entity type. The transition between languages is performed primarily using WP interwiki links, i.e. manually defined connections between WP articles in different languages, but describing the same entity. This highlights the interest of using WP as a base for deriving NER tools adapted to multilingual purposes. We plan extending our own NER tool Subee in a similar way.

6 Conclusion

In this work, we described a method to improve the performance of NER tools when applied to biographic texts, and more particularly to Wikipedia articles. It consists in combining existing state-of-the-art standalone NER tools, as well as a custom tool we designed specifically to process Wikipedia articles (or more generally, text containing hyperlinks). We proposed two different combination methods (plus some variants), and compared them experimentally on a corpus constituted by ourselves. It turns out both methods obtain better results than the NER tools they are built upon, when these are considered individually. Our most promising results are obtained with the SVM-based approach, when it both takes advantage of the global feature we introduced and is used jointly with our custom NER tool.

One of the main advantages of our approach is that it can be used to integrate heterogeneous NER algorithms in a single tool, since it does not rely on any assumption regarding their functioning. Moreover, it allows using tools one cannot train for practical reasons. This was noticeably the case, in this work, of the OpenCalais Web Service, whose source code is closed. Another interesting point is that the combination process itself is language- and domain-independent: our method could be applied to process any type of text, provided the NER tools used as inputs are able to treat it.

However, some limitations remain. The first one concerns the training time of the SVM, which is rather long (several hours on a standard workstation). This point could be improved by considering faster classifiers, but it remains to be seen if the result of the combination would be better than the results obtained with some standalone NER tool. Note the detection time, on the contrary, depends only on how fast the considered NER tools are. Second, we were expecting the results of the SVM-based approach to be more clearly better than those of the vote-based one. We plan to introduce other features in the classification process, as we did with the category of the processed article, to improve the SVM-based method further. In the longer term, we will be studying how our tool performs in a multilingual context. In particular, we plan on taking advantage of the fact Wikipedia pages exist in several linguistic versions one could use as complementary sources for NER.

References

[1] Wikipedia:manual of style/abbreviations, 2013. http://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style/Abbreviations.

[2] Wikipedia:manual of style/linking, 2013. http://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style/Linking.

[3] Alias-i. LingPipe 4.1.0, 2008. http://alias-i.com/lingpipe.

[4] Apache Software Foundation. Apache OpenNLP, 2011. http://opennlp.apache.org/index.html.

[5] S. Atdağ and V. Labatut. A comparison of named entity recognition tools applied to biographical texts. In *2nd International Conference on Systems and Computer Science*, pages 228–233, 2013.

[6] D. Balasuriya, N. Ringland, J. Nothman, T. Murphy, and J. R. Curran. Named entity recognition in wikipedia. In *People's Web - Workshop on The People's Web Meets NLP: Collaboratively Constructed Semantic Resources*, pages 10–18, 2009.

[7] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. Dbpedia - a crystallization point for the web of data. *Web Semantics*, 7(3):154–165, 2009.

[8] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *ACM SIGMOD international conference on Management of data*, pages 1247–1250, 2008.

[9] C.-C. Chang and C.-J. Lin. Libsvm : a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27, 2011.

[10] M. Ciaramita and Y. Altun. Named-entity recognition in novel domains with external lexical knowledge. In *Advances in Structured Learning for Text and Speech Processing Workshop - Advances in Neural Information Processing Systems*, 2005.

[11] M. Collins. Ranking algorithms for named-entity extraction: boosting and the voted perceptron. In *40th Annual Meeting on Association for Computational Linguistics*, pages 489–496, 2002.

[12] S. Cucerzan. Large-scale named entity disambiguation based on wikipedia data. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 708–716, 2007.

[13] W. Dakka and S. Cucerzan. Augmenting wikipedia with named entity tags. In *3rd International Joint Conference on Natural Language Processing*, volume 1, pages 545–552, 2008.

[14] N. Dimililer, E. Varoğlu, and H. Altınçay. Vote-based classifier selection for biomedical ner using genetic algorithms. *Lecture Notes in Computer Science*, 4478:202–209, 2007.

[15] G. Doddington, A. Mitchell, M. Przybocki, L. Ramshaw, S. Strassel, and R. Weischedel. The Automatic Content Extraction (ACE) program: Tasks, data, and evaluation. In *4th International Conference on Language Resources and Evaluation*, 2004.

[16] A. Ekbal and S. Saha. A multiobjective simulated annealing approach for classifier ensemble: Named entity recognition in indian languages as case studies. *Expert Systems with Applications*, 38(12):14760–14772, 2011.

[17] A. Ekbal, S. Saha, and C. S. Garbe. Multiobjective optimization approach for named entity recognition. *Lecture Notes in Computer Science*, 6230:52–63, 2010.

[18] R. Farkas, G. Szarvasy, and A. Kocsor. Named entity recognition for hungarian using various machine learning algorithms. *Acta Cybernetica*, 17:633–646, 2006.

[19] J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by Gibbs sampling. In *43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370, 2005.

[20] R. Florian, A. Ittycheriah, H. Jing, and T. Zhang. Named entity recognition through classifier combination. In *7th Conference on Natural Language Learning*, volume 4, pages 168–171, 2003.

[21] R. Grishman and B. Sundheim. Message Understanding Conference-6: a brief history. In *16th Conference on Computational Linguistics*, pages 466–471, 1996.

[22] H. L. Guo, L. Zhang, and Z. Su. Empirical study on the performance stability of named entity recognition model across domains. In *Conference on Empirical Methods in Natural Language Processing*, pages 509–516, 2006.

[23] E. Hovy, M. Marcus, M. Palmer, L. Ramshaw, and R. Weischedel. OntoNotes: The 90% solution. In *HLT/NAACL*, 2006.

[24] J. Kazama and K. Torisawa. Exploiting wikipedia as external knowledge for named entity recognition. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 698–707, 2007.

[25] Z. Kozareva, O. Ferrández, A. Montoyo, R. Muñoz, and A. Suárez. Combining data-driven systems for improving named entity recognition. *Lecture Notes in Computer Science*, 3513:80–90, 2005.

[26] L. I. Kuncheva and C. J. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, 51(2):181–207, 2003.

[27] A. Mansouri, L. Suriani Affendey, and A. Mamat. Named entity recognition approaches. *International Journal of Computer Science and Network Security*, 8(2):339–344, 2008.

[28] M. Marrero, J. Urbano, S. Sánchez-Cuadrado, J. Morato, and J. M. Gómez-Berbís. Named entity recognition: Fallacies, challenges and opportunities. *Computer Standards and Interfaces*, in press, 2013.

[29] D. Nadeau and S. Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

[30] J. Nothman, J. R. Curran, and T. Murphy. Transforming wikipedia into named entity training data. In *Australasian Language Technology Workshop*, page 9, 2008.

[31] L. Ratinov and D. Roth. Design challenges and misconceptions in named entity recognition. In *13th Conference on Computational Natural Language Learning*, pages 147–155, 2009.

[32] A. E. Richman and P. Schone. Mining wiki resources for multilingual named entity recognition. In *ACL*, pages 1–9, 2008.

[33] E. F. T. K. Sang and F. de Meulder. Introduction to the CoNLL-2003 shared task: language-independent named entity recognition. In *7th conference on Natural Language Learning*, 2003.

[34] G. Szarvas, R. Farkas, and A. Kocsor. A multilingual named entity recognition system using boosting and c4.5 decision tree learning algorithms. *Lecture Notes in Computer Science*, 4265:267–278, 2006.

[35] P. T. X. Thao, T. Q. Tri, D. Dien, and N. Collier. Named entity recognition in vietnamese using classifier voting. *ACM Transactions on Asian Language Information Processing*, 6(4):3, 2007.

[36] Thomson Reuters. Calais Web service, 2008. http://www.opencalais.com/.

[37] A. Toral and R. Munoz. A proposal to automatically build and maintain gazetteers for named entity recognition by using wikipedia. In *11th Conference of the European Chapter of the Association for Computational Linguistics - Workshop on New Text*, pages 56–61, 2006.

[38] M. Vilain, J. Su, and S. Lubar. Entity extraction is a boring solved problem: or is it? In *Conference of the North American Chapter of the Association for Computational Linguistics*, pages 181–184, 2007.

[39] A. Vlachos. Evaluating and combining biomedical named entity recognition systems. In *Workshop on BioNLP 2007: Biological, Translational, and Clinical Language Processing*, pages 199–206, 2007.

[40] C.-W. Wu, S.-Y. Jan, R. T.-H. Tsai, and W.-L. Hsu. On using ensemble methods for chinese named entity recognition. In *5th SIGHAN Workshop on Chinese Language Processing*, pages 142–145, 2006.