Extracting tags from large raw texts using End-to-End memory networks

Feras Al Kassar, Frédéric Armetta

Extracting tags from large raw texts using End-to-End memory networks

Feras Al Kassar, Frédéric Armetta

Université de Lyon, CNRS
Université Lyon 1, LIRIS, UMR5205, F-69622, France

19-09-2017

Abstract

Recently, new approaches based on Deep Learning have demonstrated good capacities to manage Natural Language Processing problems. In this paper, after selecting the End-to-End Memory Networks approach for its ability to efficiently capture context meanings, we study its behavior when facing large semantic problems (large texts, large vocabulary sets) and apply it to automatically extract tags from a website. A new data set is proposed, results and parameters are discussed. We show that the so formed system can capture the correct tags most of the time, and can be an efficient and advantageous way to complement other approaches because of its ability for generalization and semantic abstraction.

1 Introduction

To automatically extract the meaning of a web page or a raw text is still a deep challenge. Rule-based approaches which have been applied for years are efficient but can't manage easily diversity, and suffer from hand-made drawbacks like the difficulty to model a rich word built of complex interacting semantical concepts. Natural Language Processing (NLP) based on Deep Learning have recently been applied to capture the meaning of texts, and to build inferences so that it could be possible, as a long term goal, to have a full and natural discussion with such a system (Li et al. (2016)).

In this paper, after covering some of the deep learning methods to address NLP problem's, we select a promising *End-to-End Memory Networks* approach for its ability to capture the meaning of complex words associations. As presented in section 2, this approach has been designed to preserve the memory of texts acquired.

For our study, we apply the approach to a large problem, so that we can see how the *End-to-End Memory Networks* can be applied to the Web. We choose to study long texts (biographies), with a large set of words leading to a high memory and computing consumption. The section 4 introduces and motivates the elaboration of a new dataset based on biographies proposed by the website `http://biography.com`. Experimental results and parameters are then discussed in section 5. Section 6 concludes and introduces some perspectives.

2 State of the art and positioning

Capturing the meaning of texts requires to capture the meaning of words, and the meaning of word associations. Words can be efficiently represented by a dedicated embedding. World associations can be captured considering their sequentiality.

Recurrent neural networks (RNN), can be applied to capture the order between words. In this case, the network is progressively fed, word after word. The network tends to forget long relation between words for large texts. Long Short Term Memories (LSTM, Hochreiter and Schmidhuber

(1997)) allows to moderate this limitation. On their side, Memory Networks can be fed by an aggregation of words in one step. Based a some state of the art comparison results, we can discuss the ways to address the problem, and settle on Memory Networks for their ability to manage efficiently word embeddings and complex semantic contexts.

2.1 Word embeddings

The word embeddings aim to find a representation for strings in a high dimensional space. The general idea is to learn appropriate vectors for each of the available words as presented in Mikolov et al. (2013). Doing so, a distributed representation for words in a vectorial space manifests good properties as a base to learn the languages: similar words will be placed close to each other. The learning is really rich, it will not just be about the semantic (equation 1) but it will also be about syntactic (equation 2).

$$Vec(\text{“}Italy\text{”}) + Vec(\text{“}Rome\text{”}) - Vec(\text{“}France\text{”}) = Vec(\text{“}Paris\text{”}) \tag{1}$$

$$Vec(\text{“}run\text{”}) + Vec(\text{“}running\text{”}) - Vec(\text{“}walk\text{”}) = Vec(\text{“}walking\text{”}) \tag{2}$$

The computational and memory consumption for embeddings highly rely on the size of the dictionary. It is possible to limit the number of the words in the vocabulary. In this way we reduce the size of the network and reach better computational results. So for example the repetition words like “a”, “an” or “the” will not provide a valuable information to understand the sentence and can be removed. We can apply a filter applying the following probability to discard the meaningless words, where the function f stands for the frequency of the word.

$$P(W_i) = 1 - \sqrt{\frac{t}{f(W_t)}} \tag{3}$$

Starting from a word embedding, one can then complete the approach to capture the meaning of the sentences or complete texts.

2.2 RNN and LSTM

Recurrent Neural Networks (RNN) contain loops that allow to feed the network step-by-step (word-by-word for an NLP problem). Nevertheless, long-term dependencies are not well supported because of the gradient vanishing problem (Pascanu et al. (2012)). LSTM (Long Short Term Memory) partially solved this problem by using a forgetting factor. They allow to protect data to remember, and allow to forget the useless ones.

2.3 Memory Networks and End-to-End Memory Networks

One other way to consider memory for sentences is to agglomerate sentences directly in selected slots. This is what has been proposed in Weston et al. (2014). The first Memory Networks proposal requires a layer-by-layer learning. An extension of this work called *End-to-End memory networks* allows a full back propagation for the network with no additional costs as presented in Sukhbaatar et al. (2015).

The general idea relies on imitating the human brain memory. The human memory retrieves some data from its memory thanks to global contexts completed by stimuli or events. End-to-End memory networks follow the same idea, i.e., when we write the current experience (a story) inside the memory followed by an event (a query), the appropriated knowledge is propagated to the end of the memory network thanks to the neural network well-known ability to generalize past experiences.

2.4 Comparative results

Lets consider the experiments proposed in Hill et al. (2015) and and Weston et al. (2015) which propose answering question models experimented on short stories. The first one was about using the question-answer formulation while focusing on predicting the missing word inside the question. The dataset was formed by some children books shaped sequentially using the following pattern: 20

sentences were used as a story, the next sentence was altered (one word was removed) and used as a question and so on.

The size of the vocabulary set is about 53.000, with a fixed number of sentences inside the memory (20 sentences). We can see comparative results, somehow equivalent, between LSTM and End to End memory network in table 1.

Table 1: Children's Books experiments, Hill et al. (2015)

LSTM	65%
End-to-End Memory Network	67%

For the second experiments proposed in Weston et al. (2015), stories involve rooms, people and objects. The query refers to the places of the objects. For this problem, the semantic acquisition has to be refined in order to apply inferences and deduce the position for objects. This dataset manipulates a small number of words (around 20 words), the size of the sentences is also limited (7 words and small story containing less than 20 sentences). We can see comparative between LSTM and Memory Network in table 2.

Table 2: Toy Tasks experiments, Weston et al. (2015)

LSTM	49%
Memory Network	93%

We can notice that the End-to-End memory networks are working better to extract the meaning of specific objects than predicting words. That is why we choose to experiment and test this approach for automatic extraction of tags. In order to study the scale sensibility for the approach, we will focus on a larger vocabulary set size with a larger amount of sentences standing for the context to capture.

3 End-to-End Memory Networks description

In this section, we will detail the one layer inference mechanism of end-to-end memory networks. The global approach allows to cover the network more than one time to compute complex inferences but is not used nor detailed here. Indeed, for the tag capturing problem, additional layers hasn't provided any significant gain in our experiments (see Sukhbaatar et al. (2015) for a complete description of the approach), the context can be captured with only one covering of the network.

The model aims to define continuous representations for each sentences in the stories and for the queries, then this representation is processed to generate at the output the answer of the query, as presented in figure 1. The learning is supervised and will propagate the error back through the network in order to apply network weights corrections.

3.1 Sentence representation

The memory relies in slots for each of the sentences of the story $x_1, x_2, ..., x_n$. Every x_i will be converted to a memory vector m_i of dimension d computed by embedding each x_i in a continuous space using an embedding matrix A (of size $d \times V$, with V set as the size of the dictionary). In the same manner, the query q will be embedded by a matrix B with the same dimensions to obtain the internal state u.

The temporal context for sentences is really important to catch. Each sentence is represented by the sum of pondered word embeddings: m_i will be $m_i = \sum_j l_j . A x_{ij}$ where l_j is a column vector with the structure $l_{kj} = (1 - j/J) - (k/d)(1 - 2j/J)$ with J being the number of words in the sentence, and d the dimension of the embedding. The same representation is used for questions, memory inputs and memory outputs.

An other more precise way to refine the sentence contexts stands on modifying the memory vector $m_i = \sum_j A x_{ij} + T_A(i)$ where the ith row of a special matrix T_A encodes temporal information. And in the same way $c_i = \sum_j C x_{ij} + T_C(i)$. Both T_A and T_C are learned during training.

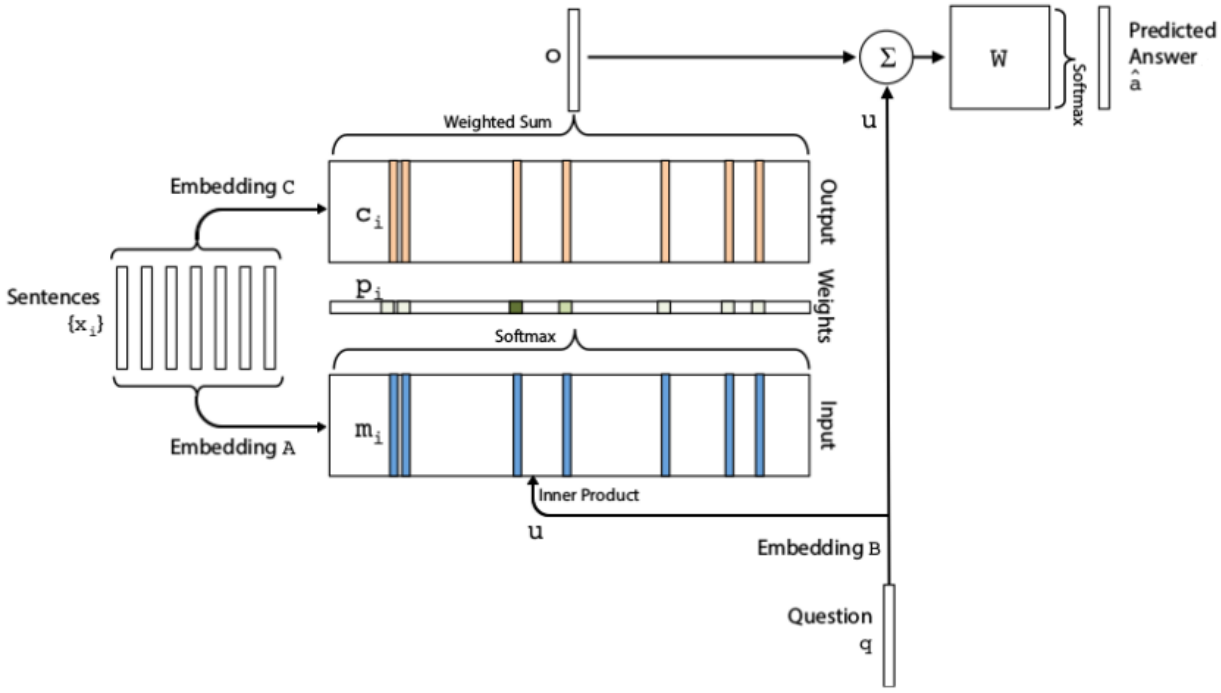

Figure 1: First part of the End to End Memory Network extracted from Sukhbaatar et al. (2015)

3.2 Propagation of the meaning through the network

We can compute the matching between u and every memory m_i by taking the *softmax* of the inner product, as described by equation 4.

$$P_i = Softmax(u^T m_i) \tag{4}$$

After that every inputs x_i will be embedded by an other matrix C with the same dimensions of A and B, then it will produce c_i for every input. The output o is the sum over the transformed inputs c_i weighted by the probability vector from the input, as presented in equation 5.

$$o = \sum_i P_i C_i \tag{5}$$

One can understand that the embeddings A and B are tuned for question-sentence correlation, while the embedding C is used to extract the meaning from previously selected relevant sentences from the story or context.

At the end, to generate the final prediction, we apply a *softmax* function to the sum of the output vector o with the question embedding u then passed through a final weight matrix W of size $V \times d$ (see equation 6).

$$\hat{a} = Softmax(W(o + u)) \tag{6}$$

4 Selected data set

4.1 Biography.com

Biography is a website containing over 7.000 profiles for famous people in many domains like Politics, Cinema, History, Sport, etc. The information provided by the website is daily updated and every person profile has a picture, tags and a raw text formed by titled paragraphs. As an example, the figure 2 shows the first part of the profile of Victor Hugo, a famous French author.

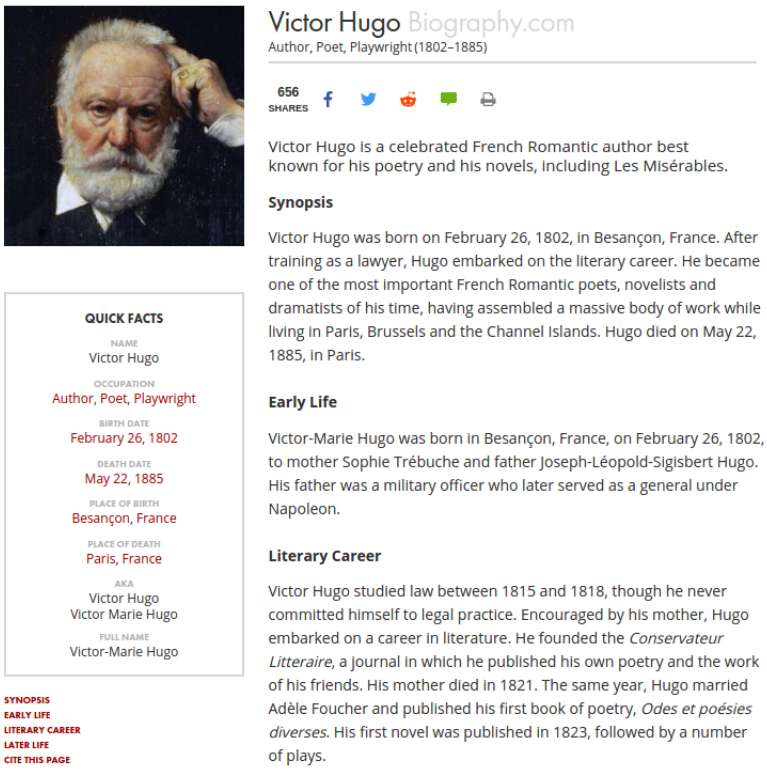

Figure 2: Victor Hugo profile in Biography.com

The supervision for the learning has been possible thanks to the tags extracted from the website. To do so, we automatically generate a question, like What is Victor Hugos occupation? (for each person), we educate the network so that it can infer the good answer and capture the relevant parts of the raw text.

4.2 Extracting and Preparing the data

To extract the data from the website we used python with Selenium library to simulate a browser, this allows to request every article and extract the article with the tags. We extracted 6000 profiles. We then divided the articles into sentences using NLTK library in python and we deleted the repeated words. The longest article has 410 sentences, the average of the provided sentences is 49. The vocabulary size is 54.928 and the longest sentence has 88 words.

4.3 Hardware

We used a strong graphic card (NVIDIA TITAN X where GPU Architecture: Pascal, Frame buffer: 12 GB G5X, Memory Speed: 10 Gbps,Boost Clock Actual: 1531 MHz) to make the Tensorflow (specific machine/deep learning library which allows GPU computing) processing as efficient as possible.

We use 4500 profiles for the training with one question for every profile and 1500 profiles for testing. The training took about half an hour for each of our experiments.

5 Experimental results

The goal of this implementation is to test the limits of the end-to-end memory network and apply it on a complex and large data set extracted from a website, observing the results and evaluating the difficulties to reach the best solution. This algorithm needs to tune many parameters (the memory size, the embedding dimensions and the number of epochs).

Table 3: Result 1

Epochs	Embedding Dimensions	Memory Size	Result
20	50	10	0.545
20	50	50	0.62
20	50	100	0.635
20	50	150	0.59
20	50	200	**0.65**
20	50	250	0.61
20	50	300	0.625
20	50	350	0.615
20	50	400	0.64

Table 4: Result 2

Epochs	Embedding Dimensions	Memory Size	Result
40	50	10	0.56
40	50	50	0.62
40	50	100	0.65
40	50	150	**0.655**
40	50	200	0.63
40	50	250	0.605
40	50	300	0.63
40	50	350	0.62
40	50	400	0.60

We tried two strategies to deal with the parameters: the first one was to vary values for the memory size, the number of epochs and the embedding dimensions. The memory loads be the whole story, so each time the story is longer than the memory the model only keep the last sentences. We used 4500 profiles for the training with one question for every profile and 1500 profiles for the testing (table 3 and table 4).

Next we studied the influence of the embedding dimensions. Increasing the embedding dimensions and memory size causes the program to run out of memory. We can notice also that increasing the embedding size will not enhance the results quality, when the best result in this testing was with embedding dimensions 100 and memory size 100 with a mean result of 0.685/1 information retrieving.

In order to check the validity of our hyper-parameters, we applied genetic algorithms (Mitchell (1998)). The best result was 0.665 with the embedding dimensions set to 144, the memory size set to 86 and 20 epochs. The parameters for the genetic algorithm was a population size of 35, 10 generations, the embedding dimensions allowed was set between 10 and 450 and the memory size allowed between 10 and 400.

We can compare our results with the children's books presented in section 2.4, even if it is not the same problem: our work is about extracting information from the a raw text from a long story (the longest story has 410 sentences) with unrestricted sentences size (the longest sentence has 88 words), while the children's books focus on predicting the words with short story 20 sentences. Nevertheless,

Table 5: Result 3

Epochs	Embedding Dimensions	Memory Size	Result
20	100	50	0.64
20	100	100	**0.685**
20	100	150	0.65
20	100	200	0.645
20	100	250	ME

Table 6: Result 4

Epochs	Embedding Dimensions	Memory Size	Result
20	150	50	0.665
20	150	100	0.66
20	150	150	ME

both of them are using a large vocabulary size (around 50.000) and use End-to-End memory networks (table 7).

Table 7: Children's books and Biography profiles

	The goal	Memory Size	Embedding Dimensions	Vocabulries	Results
Children's Book	Predict the missing word	20	N.C.	53.000	65%
Biography profiles	Extract the occupation	100	100	54.000	68%

To have a semantic deep view of the results, let us see some of the predictions. For example: Rielle Hunter has no occupation for her in the biography but the model predicted her as a queen, with deep looking we can see she is married with John Edwards whose occupation was an U.S. representative. So the system probably exploited the relation between them, inferring on the way to defined her relation with her husband.

6 Conclusion and perspectives

In this work, we are interested in studying deep learning abilities to capture semantic inferred tags from a website. Our motivation is first to identify an approach able to learn large contexts or texts, while using large set of vocabularies. We show that memory networks manifest good properties for this kind of problems. Our results show that, for the selected tag retrieving problem, the system doesn't suffer so much from the large problems we tackle and associated memory sizes. It can succeed in identifying relevant parts of large texts used by its inference process for more than 65% of the cases. Some keywords express close meanings (poet, author, writer, etc.) and because the system is looking for an exact matching, probably success should be at a slighter higher score that what is reported here. Nevertheless, these results are already really encouraging and can be really useful to complement a rule based approach or other statistical approaches. The application of the network to a webpage is really quick, so that we can envisage for further works to apply the network for enhanced queries on the web. Many applications can be considered to make benefits from this new tool for natural language processing (social comments meaning extraction, product recommendation, etc.).

Acknowledgements

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan X Pascal GPU used for this research. This project is the result of a cooperation between the LIRIS laboratory and the Deal On company (`http://www.dealon.fr`).

References

Hill, F., A. Bordes, S. Chopra, and J. Weston (2015). The goldilocks principle: Reading children's books with explicit memory representations. *CoRR abs/1511.02301*.

Hochreiter, S. and J. Schmidhuber (1997, November). Long short-term memory. *Neural Comput. 9*(9), 1735–1780.

Li, J., A. H. Miller, S. Chopra, M. Ranzato, and J. Weston (2016). Dialogue learning with human-in-the-loop. *CoRR abs/1611.09823*.

Mikolov, T., I. Sutskever, K. Chen, G. S. Corrado, and J. Dean (2013). Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 26*, pp. 3111–3119. Curran Associates, Inc.

Mitchell, M. (1998). *An Introduction to Genetic Algorithms*. Cambridge, MA, USA: MIT Press.

Pascanu, R., T. Mikolov, and Y. Bengio (2012). Understanding the exploding gradient problem. *CoRR abs/1211.5063*.

Sukhbaatar, S., A. Szlam, J. Weston, and R. Fergus (2015). End-to-end memory networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, NIPS'15, Cambridge, MA, USA, pp. 2440–2448. MIT Press.

Weston, J., A. Bordes, S. Chopra, and T. Mikolov (2015). Towards ai-complete question answering: A set of prerequisite toy tasks. *CoRR abs/1502.05698*.

Weston, J., S. Chopra, and A. Bordes (2014). Memory networks. *CoRR abs/1410.3916*.