CS221 Final Report
Machine reading from a contextualized data source: sports reports

Nathanael Romano and Daniel Levy

December 11, 2015

Abstract

Machine reading is the exciting challenge to build systems capable of reading natural language documents and most of all understand them. This project focuses on this problem but within a very specific scope: answering natural language questions about highly contextualized documents, in this case: sport game reports. To be more specific, this will be applied on soccer-related articles from the website ESPN. We tackle this problem by building a system capable of finding the answer to specific questions, without any world knowledge, only focusing on sentence structure and context.

1 Presentation

1.1 Problem

The high-level problem statement is straightforward: we want to build a system capable of reading a news article about a soccer game, and then answer factoid questions about the game.

1.2 Expected behavior

Given an article about a soccer game, we want it to be able to answer a natural language question about the game, such as *Who scored the first goal for FC Barcelona?*. The system should only be using the text of an article about the game[1]. In the aforementioned article, the system should be able to answer *Javi Guerra scored the first goal.*

In practice, once it is trained, the system should present a text interface, allowing the user to enter an article url. The user should then be able to input questions in natural language, to which the system would answer.

1.3 Evaluation

We used two methods to evaluate the system's quality.

- a **human** method, where a person actually uses the system interactively by asking questions about game. This method allows us to insightfully diagnose and evaluate errors.
- a **systematic** method, where the answers to our set of questions are comparated to the real answer, on a labeled test set of articles. This method allows us to compare different models.

2 Model

2.1 Anonymisation

A first key element in our approach is the anonymisation of entities in the article (players, teams, stadiums, etc.).

As stated earlier, we want the system to be independant of world knowledge, and rather to focus on actually learning to read the document. We believe that this is particularly true in the case of answering questions about sports reports. For example, for any given game, if the question is "Who scored the first goal?", a simple system using a n-gram model might be enclined to answer

[1] Such as the one the following link: `http://www.espnfc.us/spanish-primera-division/match/433905/barcelona-rayo-vallecano/report`.

"Lionel Messi", as this answer probably occurs a lot in the dataset. On the other hand, it would be impossible for it to detect a goal from a player that has never played before.

Thus, when training our system, and when trying to answer questions, we anynonymze the data, replacing any entity (i.e. player, team, stadium, referee, etc.) by a token of the form 'entX'.

2.2 Key variables and notations

Let's introduce the key variables in this problem. We denote q a given question (e.g. *Who scored the first goal?*), a the correct answer to this query, and w an entity, candidate answer to the question.. We denote d the document for a given question (i.e. the article it is in), and $c(w)$ the *context* in which a possible answer w is, i.e. the sentence in which it is contained. We also denote $c(d, w)$ the set of contexts in which w occurs in d.

2.3 Objective

When formalized with those variables, we model the problem as finding the most likely answer among candidate answers, given the query, the article, and the candidate answer's context:

$$w^* = \arg\max_w P(a = w \mid q, d)$$

Our goal is to formalize an algorithm to estimate this likelihood.

3 Approach and algorithms

Figure 1 shows a schematic of our iteration process, summarizing the various approaches.

3.1 Entity-centric model

3.1.1 Motivation

The starting point of our project was that we noticed that there are clear patterns in the way journalists phrase their sentences.
As such, for each word, we performed a logistic regression to estimate:

$$P(a = w \mid c(w))$$

.

We can notice that this probability doesn't take into account the document but only the context of a word.

This quantifies the probability of a word being the answer, given that he appears in a certain context.

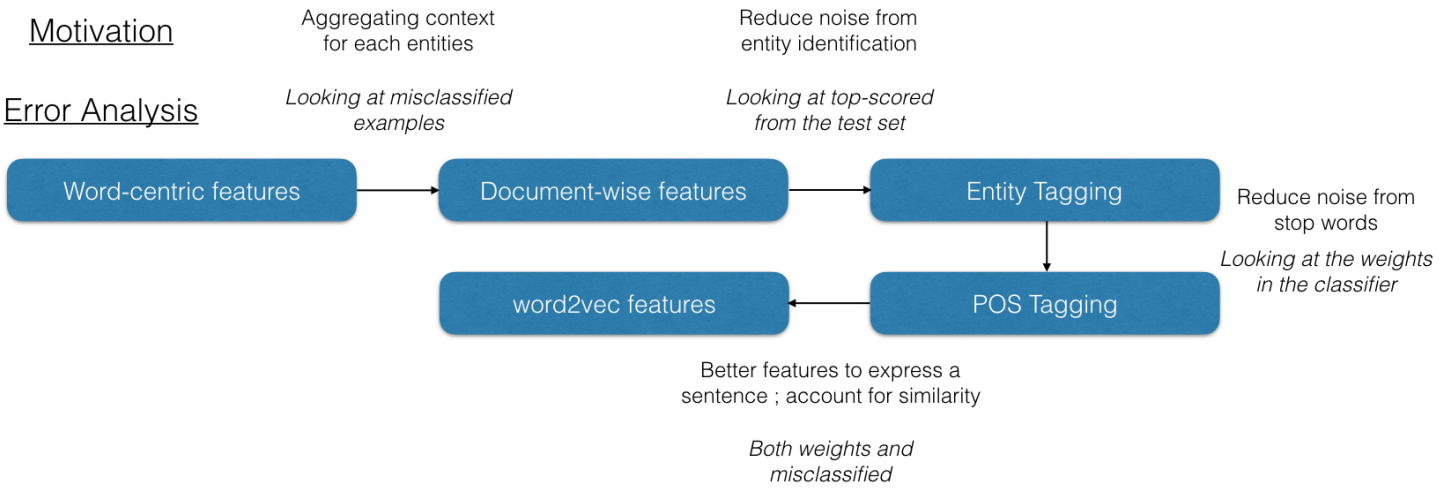

Figure 1: Sucessive approaches

This approach was motivated by the need to capture specific phrasing and had the benefit to artificially increase the size of our data. Indeed, we have a row of data for every word in every article.

For now, we defined a context of a word w by the word before and the word after, i.e., if w appears in $w_1 w w_2, c(w) = (w_1; w_2)$ represented in a sparse manner. To be more specific, the sentence *The United captain has received criticism for his performances recently.* would yield the following feature vector for $w =$ *United*:

$$\phi(w) = \{\text{before\_the} : 1,\ \text{after\_captain} : 1\}$$

3.1.2 Error Analysis

We ran this method on a preliminary set of 22 articles with the query "Who won?". The main issue with this algorithm is that it didn't capture whether an entity was the answer but more the fact that the entity was a team or not (in 77% of the testing examples, the two teams were in the top 3 scores).

The logistic regression also outputted probabilities in a very similar range so any given word could propulse one entity to the top score (even by a very slight margin) and thus made it very sensitive to noise.

3.2 Document-wise features

3.2.1 Motivation

To try and even out the noise, we decided to build document-wise features instead of simply entity-centric ones. Meaning that an entity e is represented by a vector of every context it appears in (still represented sparsly). This feature vector is then labelled 0 or 1 depending on whether the entity is the answer.

We are now estimating:

$$P(w = a|q, c_1(w), \ldots, c_n(w))$$

where $c_1(w), \ldots, c_n(w)$ are the contexts in which the entity appears.

To make a prediction, we just return :

$$w^* = \arg\max_w P(w = a|q, c_1(w), \ldots, c_n(w))$$

In this case, for a structure in the form $w_1w_2ww_3w_4$ we used both $c(w) = (w_1; w_2)$ and $c(w) = (w_1, w_2; w_3, w_4)$.

3.2.2 Error Analysis

We trained this algorithm on more articles (around a 50) and tested it on a test set of 40. While the success rate was better, there were still some inconsistency with the entity tagging who led to a lot of noise being recognized as entities (start of sentences, scores...). To improve upon that, we implemented a more performant entity tagger.

We also noticed that stop words' ("the","in","a","into"...) weights were the most negative which didn't seem natural as they should be "neutral-information" instead of proof of not being the answer. Our answer was to introduce a "Part-of-speech tagging" component, mostly to get rid of stop words.

3.3 Entity Recognizer and POS Tagging

3.3.1 Motivation

The entity recognition was based on `Stanford NER Tagger` which given a tokenized sentence tag the entities. We then performed a union-find like algorithm to reconstruct the similar entities (e.g. Christiano Ronaldo, Ronaldo and Christiano).

We also got rid of stop-words using a preset list.

3.3.2 Error Analysis

Upon error analysis, we noticed that some phrasing weren't properly captured, mostly because the determining factor could happen at the end of the sentence

(e.g. "Barcelona, against all odds, won the game."). Also, as in "sentiment analysis", the negative phrasing wasn't properly detected. This called for an entity featurization that could capture the whole sentence.

3.4 word2vec

3.4.1 Motivation

Word2Vec is an algorithm to obtain a mapping from each word to a vector (called an embedding) based on a corpus. The algorithm uses neural networks and a skip-gram model (explained in [3]) to learn if a n-gram is likely ; a side-effect of that is that the system learn the embeddings which make "semantic" sense. The classic example is that similar words will be close (for example "France" and "England") as well as capture relationships ("Paris" - "France" = "London" - "England").

Even in our small corpus, such relationships were captured: indeed the most similar to "Messi" - "Barcelona" - "Cristiano" was "Madrid".

3.4.2 Error Analysis

Our corpus still seems a bit too small and as such might overfit some strange embeddings. For example, when computing the cosine similarity of the word "player", the top-3 words are "same","becoming" and "the" which doesn't make semantic sense.

4 Infrastructure

Besides the algorithms per se, a key issue in our problem is building a proper infrastructure. Indeed, we need a consequent amount of data for our system to be efficient, and this data should be labeled (i.e. we should have the correct answer to the questions). As no public processed dataset of news article is available, and it is realy tidious to manually fetch and label a large set of articles, we built a consequent infrastructure to do so.

4.1 Scraping

The data we are using consists of game reports from the website espn.com, as the one in the link provided above. To build our dataset, we chose a set of questions, and an instance of this data set is the article text and the answer to all those questions.

To obtain the dataset, we built a system capable of scraping articles from the ESPN website, and fetching the answer from the Live Commentary section of the article. This system allowed us to automatically get a large number of articles.

4.2 Processing

4.2.1 Entity Tagging

We used the `Stanford NER` from the NLP team to be able to automatically tag entities. The tagger will output a list of pairs (token, tag) given a tokenized sentence. The NER will either associate the label 0 or a more precise one: Location, Organization or Person.

We didn't make use of the content of the tag as it could sometimes be subject to noise (*Paris Saint-Germain* could be either a Location or an Organization for example) but we used it to regroup successive entities ('Bayern' 'Munich' to 'Bayern Munich').

4.2.2 Entity reconstruction

After tagging all the entities, we implemented an algorithm to reconstruct. Indeed, if the article contains 'Cristiano', 'Ronaldo', and 'Cristiano Ronaldo', the system should be able to understand that this is a different wording of the same entities. We also regrouped entities and acronyms of entities ('Paris Saint-Germain' and 'PSG' or 'FC Barcelona' and 'FCB').

This was performed using a Union-Find like algorithm and a simple similarity function.

4.2.3 Stop words and scores

As explained earlier, we removed stop words and scores to reduce noise in the logistic regression's weights. Since speech tagging libraries can get the code really slow, we manually coded the stop word and scores removal, using respectively a hard coded list of stop words, and a regular expression.

4.2.4 word2vec

To benefit from a better featurization vector, we used the original `word2vec` C program (see [3] for the derivation) by Tomas Mikolov. We were able to run the scripts on our corpus of 1500 articles and store the embeddings.

4.3 Learning

For the learning algorithms, we used the library `LIBLINEAR`[2], which offers the advantages of being extremely fast, and being able to handle sparse feature vectors represented as dictionnaries. We used a logistic regression, with a ridge regularization of 1 (we did not focus on tuning this hyper-parameter, as this was not the focus of our project).

[2]The library and its documentation are available on https://www.csie.ntu.edu.tw/ cjlin/liblinear/

4.4 Interaction

Our system is highly interactive as a user can scrape, train and test it (with natural language questions) using mostly the command line.

5 Dataset

5.1 Questions

We only used this set of questions:

- Who lost?
- Who won?
- Who scored the nth goal?

Note that the two first questions are the ones with the most training articles (as the ones with the third template might be not relevant for some games).

5.2 Games

We scraped three datasets of different size. Each dataset consists of a training, development and test set, each of which contains articles and the corresponding answer (cf Appendix A for an example).

The *Small* dataset was used in the first iterations, to conduct experiments on the data. The *Big* dataset was used to develop the more advanced algorithms. Lastly, the *Huge* dataset was used to train and test our last algorithm. Those datasets are summarized in the following table.

Dataset	Training	Development	Test	Total
"Small"	51	42	29	122
"Big"	323	65	65	453
"Huge"	1052	0	229	1281

The "Huge" datasets corresponds to 11 months of game reports. In practice, a larger set could be scraped, but the *Named Entity Recognition* is making training really long.

6 Results

- When training on the small dataset, we obtained the following results:
  - the skip-gram model with a window of 2 words achieved an accuracy of 17.7%

  - the skip-gram model with a window of 4 words achied an accuracy of 18.3%
  - the word2vec model achieved an accuracy of 11.2%

- When training on the big dataset, we obtained the following results:
  - the skip-gram model with a window of 2 words achieved an accuracy of 32.3%
  - the skip-gram model with a window of 4 words achieved an accuracy of 36.4%
  - the word2vec model achieved an accuracy of 46.0%

We can see that the word2vec model performs very poorly on the small dataset, but his accuracy increases significantly with the training set size. As its dimensionnality is very high (it creates vectors of length 200), it needs a huge amount of data to perform an efficient fit.

To finish, we trained the word2vec model on our 'huge' dataset. We got an accuracy of 56.9%. This accuracy is way better than the ones obtained on the small dataset. It appears clearly that the accuracy is still widely increasing with the data and as such more data would improve the system. It is also interesting to note that the two teams are in most cases in the top 3 scores.

As a sidenote, when looking closely at the results from the automated tests, there were some occurences of the team's nickname being returned (e.g. 'Toffees' for Everton). While this can't be detected by our test error assessment, it may slightly reduce our accuracy (cf Appendix B).

7 Literature review

Information retrieval has always been a very discussed subject in the context of AI. In 1969 (see [8]), Robert Simmons already saw the opportunity for machines to be able to read and answer questions. This subject has always been a hot research topic (see [2], [7]). While there have been achievements here and there, it seems that, as it is the case in many subjects across AI, Neural Networks has allowed machine reading to make a leap forward.

While previous systems aimed at building a knowledge base and have the system make deduction on what it knows, recent papers have tried to better retain information from reading a text. This lead to the Memory Networks [4] from Facebook AI. The idea is quite similar to a LSTM Networks but with an emphasis on being able to have a better "memory". These Memory networks were able to outperform both RNN and LSTM, being able to solve problem up to difficulty 5 (The subject of the question has been subject to 5 actions).

A recent paper from Google DeepMind [1] also underlined the necessity for the capacity of learning information while reading and updating what has been learned when new information comes in. This led to Attention-based deep neural networks. Their algorithm is based on 2-layered deep LSTMs. This paper also completely eliminates the bias of any world knowledge as they even anonymize the entities.

8 Conclusion

The results of our testing are really encouraging as we can achieve almost 60% accuracy which best our baseline by a fairly large margin. This project was particularly enriching as we had to model the problem, which lead to several iteration of the model: entity-centric, document-wise, word2vec... as well as solve it. It also allowed us to get an overview of useful tools for NLP (how to scrape a website with Python, NLTK, Stanford NER Tagger, word2vec...).

At the end of the project, it seems very likely that an approach using neural nets with more data would be the way to achieve significantly better results, but we didn't favor this path to begin with because it involved a lot of tools unrelated to the course. If we were to continue working on this project, we would experiment with Neural Networks (and more specifically LSTM) and optimize the word embeddings jointly with the classification task.

References

[1] Hermann K., Kočiský T., Grefenstette E., Espeholt L., Kay W., Suleyman M., Blunsom P., 2015, *Teaching Machines to Read and Comprehend*

[2] Stephanie Strassel S., Dan Adams D., Henry Goldberg H., Jonathan Herr J., Ron Keesing R., Daniel Oblinger D., Heather Simpson H., Robert Schrag R., Jonathan Wright J., *The DARPA Machine Reading Program - Encouraging Linguistic and Reasoning Research with a Series of Reading Tasks*

[3] Yoav Goldberg, Omer Levy, 2014, *word2vec Explained: Deriving Mikolov et al.'s Negative-Sampling Word-Embedding Method*

[4] Jason Weston, Sumit Chopra, Antoine Bordes, 2015, *Memory Networks*

[5] Razvan C. Bunescu, Raymond J. Mooney, *Subsequence Kernels for Relation Extraction*

[6] Nguyen Bach, Sameer Badaskar, *A Review of Relation Extraction*

[7] Christopher Manning, *Text-based question answering systems*

[8] Robert Simmons. 1969. *Natural Language Question Answering Systems*

[9] Wen-tau Yih and Xiaodong He and Christopher Meek. 2014. *Semantic Parsing for Single-Relation Question Answering*, Proceedings of ACL.

Appendix A: Instance in the dataset

The following is an instance in the dataset[3].

Text

Wayne Rooney scored his first goal at Goodison Park for eight years as Manchester United coasted to a 3-0 victory on an emotional afternoon for Everton.

The news of Howard Kendall's death prior to kick-off led to a subdued atmosphere inside Goodison Park and the current crop of Everton players were unable to replicate the kind of performances seen under the club's greatest manager.

Non-existent marking from the hosts allowed Morgan Schneiderlin and Ander Herrera to put United 2-0 up inside 22 minutes and Rooney wrapped up the win in the second half.

The United captain has received criticism for his performances recently, but he looked like the Rooney of old, beating the offside trap and rolling the ball past Tim Howard to score at the home of his boyhood club for just the third time.

It was also his first league goal on the road since November last year.

All four corners of the ground took part in a minute's applause for Kendall, who oversaw Everton's most successful period in the mid-1980s, before kick-off.

A hushed atmosphere then descended upon Goodison Park. United looked nervous early on. Phil Jones, making his first start of the season, and Chris Smalling, were struggling to handle Romelu Lukaku.

The Belgian almost closed down David de Gea when he was sold short by two backpasses in the first six minutes.

The nervousness subsided in the 17th minute, when United took the lead. The hosts made it easy for the visitors. Herrera knocked the ball back to Juan Mata from a short corner Everton were not expecting. Steven Naismith's clearance fell to Marcos Rojo, who worked the ball to Schneiderlin via Chris Smalling's chest and he placed the ball into the far corner for his first United goal.

United were 2-0 up four minutes later, and again Everton's shoddy marking was to blame. Jon Moss played advantage after Seamus Coleman clattered into Anthony Martial and the ball fell to Rojo, who swung in a deep cross to find Herrera unmarked and he nodded in.

United played sensibly, keeping the ball rather than going for the jugular.

They seemed intent to play keep ball, but a clever run behind from Rooney gave them a chance to bag a third just before the break. The skipper squared to Martial, but he scuffed his shot.

Other than a fierce shot from Ross Barkley, which De Gea tipped over, Everton rarely threatened.

Roberto Martinez brought Arouna Kone on for Naismith at half-time and Van Gaal replaced Mata with Jesse Lingard.

The England Under-21 international made a telling impact, putting Barkley off his shot after he drifted into the box.

[3]The article can be found on http://www.espnfc.com/barclays-premier-league/match/422580/everton-manchester-united/report

Everton looked more threatening with Kone and Lukaku up front. Rojo entered the book for taking down the Belgian and Bastian Schweinsteiger also saw yellow for pulling Barkley back.

De Gea then denied Lukaku a certain goal by saving with his left foot. Van Gaal was unhappy with his team and the referee. He argued with fourth official Robert Madley after a decision did not go United's way.

The Dutchman was sat happily clapping in his seat moments later, however thanks to Rooney's goal.

Schneiderlin received Phil Jagielka's wayward pass and nudged the ball to Herrera, who slipped Rooney in and he beat Howard to end a run of three successive defeats for United at Goodison Park.

Questions and answers

- Who won? *Manchester United*
- Who lost? *Everton*
- Who scored the first goal? *Morgan Schneiderlin*
- Who scored the second goal? *Ander Herrera*
- Who scored the third goal? *Wayne Rooney*

Appendix B: Sample input and output (interactive mode)

```
Which URL? Enter stop to quit.
http://www.espnfc.us/spanish-primera-division/match/433884/sporting-gijon-malaga/report
Accuracy = 0% (0/8) (classification)
Query: Who won?
I think the answer is [u'Sporting Gijon'].
It is actually Sporting de Gijón.
Top 3 [[u'Primera Division'], [u'Sergio Alvarez'], [u'Sporting Gijon']]
Which URL? Enter stop to quit.
http://www.espnfc.us/barclays-premier-league/match/422563/everton-sunderland/report
Accuracy = 4.7619% (1/21) (classification)
Query: Who won?
I think the answer is [u'Toffees'].
It is actually Everton.
Top 3 [[u'Sunderland Arouna Kone', u'Kone', u'Sunderland'], [u'Tim Howard'], [u'Toffees']]
Which URL? Enter stop to quit.
```

Figure 2: Interactive mode