Semantic Relation Extraction and Classification in Scientific Papers

Erik Kristian Janezic

Faculty of computer and information science, Ljubljana

Abstract. In this project we explore the task of semantic relation classification and extraction from abstracts of scientific papers. The project is based on the task presented by SemEval 2018 []. Nowadays the scientific field is in a great need of smart algorithms which could intelligently parse knowledge related texts of human origin and enrich the obtained facts. For example this kind of algorithms could be used to build semantic networks of scientific facts in real time, as new data is produced and new findings are published.This need is mainly present due to the flood of information produced on the daily basis in all spheres of scientific research, and the inability of normal humans to comprehend the overall global picture of the data. In this work some of the approaches that have already been successful in this field.

Keywords: Relation extraction, LSTM, Semantic fingerprint

1 Introduction

The scientific community today has the luxury to have access to state of the art sensors and automated machinery, which enables us to observe phenomena natural phenomena in ways we couldn't imagine even 50 years ago. The development of scientific machinery and sensors was rapid in the last half century, it was and still is mostly fueled by parallel and equally fast development of computer technology. New technologies such as nano pore DNA sequencing enable scientists to produce huge amounts of data and for lower price than ever before, this leads to more scientific findings and more knowledge created in some time period than ever before. Contrary to blossoming state of scientific equipment, the tools to process and meaningfully connect and interpret the global picture of the data didn't manage to keep up with the time.

We think that two main problems in scientific research today are that the researchers can not follow the bleeding edge progression of their respective fields due to the amount of new findings proposed every day and that the scientific fields are usually highly intertwined among each other, meaning that solutions to some problem might already be found in some field while someone might still be struggling in the dark. To address this and speed up scientific progression of humanity even further we need to start thinking of developing intelligent algorithms that could parse human knowledge aggregated to date, and

incorporate new findings in to its representation in real time. We could then use these intelligent överseers"to provide researchers with the feedback of general state of knowledge concerning their field, giving researchers a better idea of how to proceed in their experiments or how their work fits in the global picture. They could also serve as agents which process complex scientific queries to help scientists find solutions from a broader area than they are specialized for.

Since it is clear to us that the major source of new knowledge and information today are scientific papers, it seems reasonable that we will need to extract different higher order semantic concepts from them sooner or later, if we want to build such intelligent semantic knowledge networks. The task of *Semantic Relation Extraction and Classification in Scientific Papers* presented at SemEval 2018 seemed like a good place to start and we will describe the task, data and our approach in the following chapters. But first we will look at some previous work that has been done in the area of relation extraction and relation classification.

2 Related work

2.1 Relation classification

Traditional approaches in relation classification tasks rely on hand crafted features proposed by linguists and domain experts. Although these methods have improved over the years they still suffer from over relying on selected features which leads to error propagation. Good traditional systems are also costly to build, since you need people to manually design and select features to be used [].

Lately data driven approaches have taken the spotlight over traditional methods mainly because they do not suffer from aforementioned issues. Most widely used in this category are models which use convolutional neural network (CNN) or recurrent neural networks (RNN) or a combination of both[]. These approaches were mainly made possible with the development of quality word embeddings, which capture contextual information of individual words in a vector representation. CNNs can then use these vectors/word embeddings to find local semantic patterns in specific texts and use them to classify relations. Similarly RNNs (mostly long short term memory LSTM) models can use word embeddings to classify sequences of words. Each relation can be represented as a sequence of words before the first entity, between the two entities and after the second entity. These words are then mapped to their corresponding embeddings and a sequence of embeddings can then be used to classify relations. It has been shown that LSTM models with certain augmentations (feature aggregation and position indication) out-preform CNN models, especially when dealing with long distance relation patterns.

2.2 Relation extraction

Approaches used in relation extraction problems can be roughly classified into one of 5 method families:

- Hand-built patterns
- Bootstrapping methods
- Supervised methods
- Unsupervised methods
- Distant supervision

Hand-built pattern approach relies on finding all sentences with a specific relation in a corpus, then by analysing the words in relations proximity we can define patterns that imply this specific type of relation. Interesting work employing this approach was done in 1999 by Berland and Charniak for extracting meronym (part-whole) relations []. As in relation classification hand build approaches suffer from similar drawbacks in relation extraction as well. We would need to define patterns for every type of relation there is, these patterns would be hard to maintain and update, they are very domain specific and usually need domain expert intervention, which is costly and time consuming.

Bootstrapping approaches address the issue of lack of annotated relation data. They make use of ßeed"relations to search for relation patterns in unannotated text. The entities from a seed relation are used to find examples of of co-occurrence of both entities in the unannotated text and a new pattern is extracted from found matches. Similarly relation patter from the seed relation can be used to find different types of entities that are part of this relation pattern. The bootstrapping cycle is illustrated in figure 1. A good example of bootstrapping is the Snowball algorithm proposed by Agichtein and Gravano in 2000 []. They used it to extract *organization - location* type of relations from the text. The main issues that trouble bootstrapping are semantic drift at each iteration, they have a lot of parameters to be tuned, have no probabilistic interpretation (we don't know how good the extracted patterns are) and usually their performance is not that good.

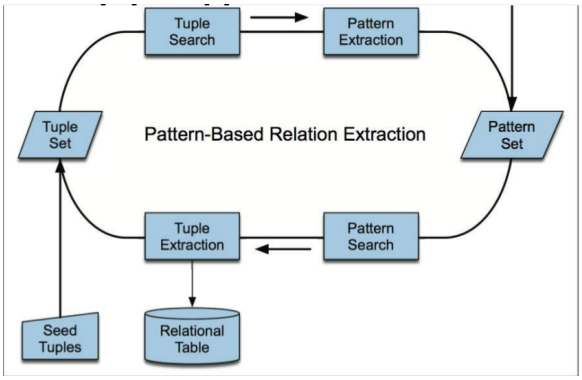


Fig. 1: Schematic representation of a bootstrapping cycle

Supervised methods heavily rely on labeled training data which is one of the main drawbacks. However they can achieve high accuracy for some relation types. The process in supervised approaches starts by defining the relation types we want to extract, then we need to collect a lot of annotated training data which can be combined with bootstrapping. Next we design relationship features, such as bag of words representations of words or dependency-tree paths between entities. We can then use different multi-class classifiers to extract relations of specified types.

The opposite extreme of supervised methods are unsupervised methods where we extract relations data purely from unannotated corpora. The foundational idea of unsupervised methods is to use low level word features, such as POS tags, to find entities (nouns) in the corpus and then extract sections of the text between entities and in their proximity to formulate simplified relation patterns. Basic supervised methods such as TextRunner [] suffer from incoherent and uninformative relation extractions and inability to detect synonymous or ambiguous relations. More advanced systems such as Yaos unsupervised relation discovery with Sense disambiguation can handle most of the mentioned [].

The distant supervision approach combines some of the advantages of the afore mentioned approaches. It incorporates the features of supervised methods via relation databases such as FreeBase and the features of unsupervised methods to use the relation database to extract training examples from corpora. Various classification algorithms in association with rich syntactical features can then be used to predict relation types of extracted relations[].

2.3 Task

This project is an attempt to solve some of the tasks presented in the SemEval 2018 *Semantic Relation Extraction and Classification in Scientific Papers* task. We were given two sets of 350 scientific article abstracts, of which one is manually annotated and the other is automatically annotated. The second set is meant to test your model on noisy data, however in our work we will initially only focus on the manually annotated dataset. Due to the lack of time we will also just focus on relation classification at this point, but we will present ideas for relation extraction as well.

The relation classification space is defined on six types of relations:

- **USAGE** is an asymmetrical relation. It holds between two entities X and Y, where, for example:
  X is used for Y
  X is a method used to perform a task Y
  X is a tool used to process data Y
  X is a type of information/representation of information used by/in a system Y)
- **RESULT** is an asymmetrical relation. It holds between two entities X and Y, where, for example:
  X gives as a result Y (where Y is typically a measure of evaluation)

X yields Y (where Y is an improvement or decrease)
a feature of a system or a phenomenon X yields Y (where Y is an improvement or decrease)

- **MODEL-FEATURE** is an asymmetrical relation. It holds between two entities X and Y, where, for example:
  X is a feature/an observed characteristic of Y
  X is a model of Y
  X is a tag(set) used to represent Y

- **PART-WHOLE** is an asymmetrical relation. It holds between two entities X and Y, where, for example:
  X is a part, a component of Y
  X is found in Y
  Y is built from/composed of X

- **TOPIC** is an asymmetrical relation. It holds between two entities X and Y, where, for example:
  X deals with topic Y
  X (author, paper) puts forward Y (an idea, an approach)

- **COMPARE** is a symmetrical relation. It holds between two entities X and Y, where:
  X is compared to Y (e.g. two systems, two feature sets or two results)

Relation data in the abstract dataset is highly unbalanced which can be seen in the table below:

USAGE	RESULT	MODEL-FEATURE	PART-WHOLE	TOPIC	COMPARE
483	72	326	234	18	95

Table 1: Distribution of relation types

When designing our classifier we will need to introduce some bias to account for less represented relations.

2.4 Data preprocessing

SemEval provides the data in two separate files. One contains article abstracts in the XML format, where each abstract and entities within them are given unique IDs. The other file contains all the relations found in the abstract, where each relation composed of a relation type and a tuple IDs of entities involved in a relation. Some relations have an extra feature in the tuple which indicates in the directional relationship between the entities within a relation (for example PART-WHOLE(car,door,REVERSE) means that door is a part of a car and not the other way around).

We decided to convert textual data to a sequence of custom objects, which will later on make it possible for us to augment local information about the surrounding text for each single word or entity inside an abstract. Each article is converted to *classes.Article*, which in addition to raw text includes a sequence of *classes.Word* and *classes.Entity* objects. Each of these objects can be assigned additional features such as POS tag,lemma, WordNet representation or its corresponding word embedding.

2.5 Approach

2.6 Long short term memory

LSTM networks have proven really efficient in the field of sequence classification and sequence prediction. Classical RNNs suffer from a vanishing gradient problem which becomes really apparent in just few time steps, LSTMs however have this issue resolved. The core idea in LSTM is the cell state which allows gradients to flow almost freely or altered only slightly by carefully controlled gates.

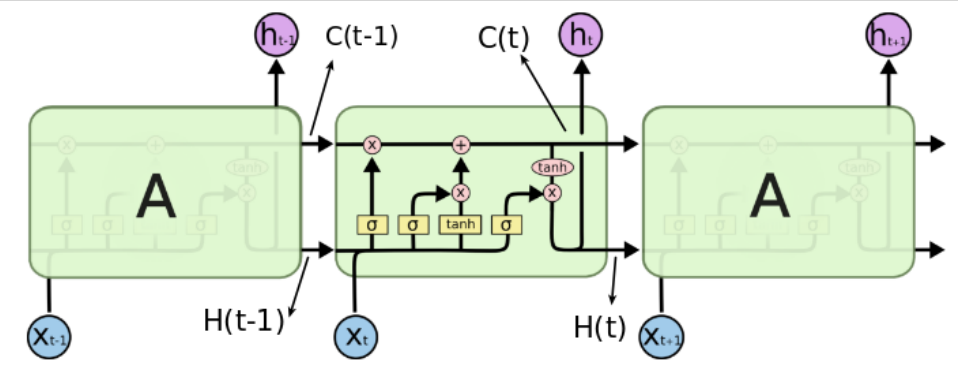


Fig. 2: Schematic representation of a unrolled LSTM

In figure 2 the cell state is marked with $C(t)$ where t is a time step, it is modified by only a multiplicative gate and additive gate. The multiplicative gate has a role of a forget gate, and the additive gate is responsible for unaffected propagation of gradients along the time steps. The forget gate operates in such way that it accepts outputs of a sigmoid function evaluated on a input vector x_t concatenated with the vector of previous hidden state h_{t-1}. The sigmoid layer outputs a number between 0 and 1 where 0 has a representation of forgetting and a leaves the cell state unaffected. In the next step the state C_{t-1} is updated to C_t via sigmoid gate which decides what values we will update in the new cell state and a tanh layer which produces the new values to be updated. The product of these two layers is then point wise added to produce C_t. The output

of the LSTM is always the hidden state of the last time step. The hidden state H_t is produced by selecting features to be updated from an old hidden state H_{t-1}, done by passing int through sigmoid layer, then this vector is point wise multiplied with the outputs of tanh layer which processes the newly created C_t described in the previous step.

Model For implementation of our LSTM model we used PyTorch framework, which provides tools to build dynamic computation graphs, classes to accumulate gradients and readily available optimization algorithms to execute variants of stochastic gradient descent.

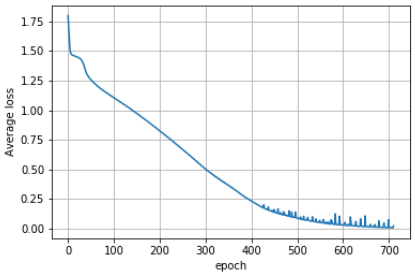
To be able to use LSTM principles in our model we first needed to decide how will we represent individual words which will be fed to the LSTM. WE decided to use pre-trained word embeddings GloVe (Global vectors for word representations)[]. We decided for the GloVe embeddings because they mostly used Wikipedia dumps as their training corpus, which is appropriate for our task since Wikipedia has quite some science related entries. There are embeddings of 3 different dimensions provided by GloVe;$1*50$,$1*100$,$1*300$. We experimented with all dimensions but observed little to no difference in performance, so we decided to use $1*50$ dimensional embeddings.

The training sequences were produced by concatenating the surroundings of entities involved in a relation. We took 5 words before and 5 words after each entity and concatenated all the words, including the entities, into a single string. String was normalized and non alphabetic characters were removed. To produce the embedding sequence we mapped each word in the produced string to its corresponding word embedding. Some words in our data set were really specific and topic related and were not found in the 6 billion Wiki token set. We decided to exclude those words from the embedded sequence production.

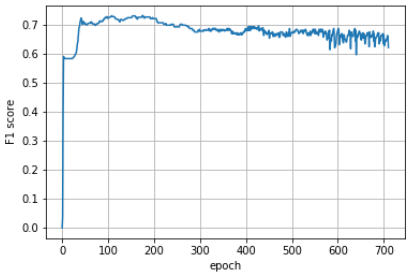
Our model is a 1 layered LSTM with a 64 dimensional hidden state and $1*50$ dimensional input. We only use the output of the last time step for predicting relation type, and we obtain relation type scores by passing last hidden state through a log-softmax layer. We need softmax logarithms because we are using negative log likelihood loss function (NLL loss) for learning. Since our training data is extremely unbalanced we use weighted NLL loss. A vector of relation type frequencies is provided as a meta argument to the NLL loss. This tells the objective function to take loss of rarer classes more serious than from common classes.

Results The data was initially split in training set with 0.8 of all examples and testing set with 0.2 of all examples.

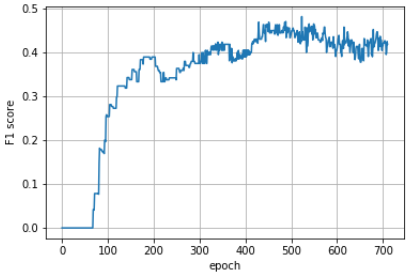
As defined by SemEval task, we use F1 metric for individual relations to evaluate the performance of the model. The results are presented in figure 3 below.



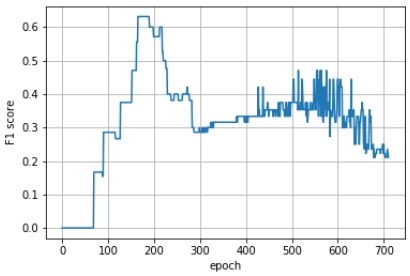
(a) Average loss



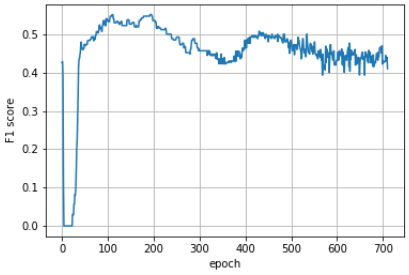
(b) F1 score for USAGE relation



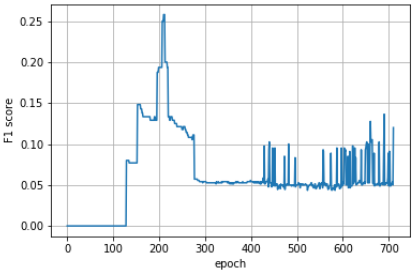
(c) F1 score for PART-WHOLE relation



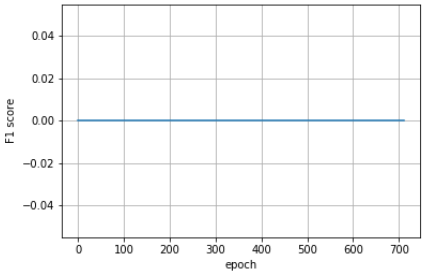
(d) F1 score for RESULT relation



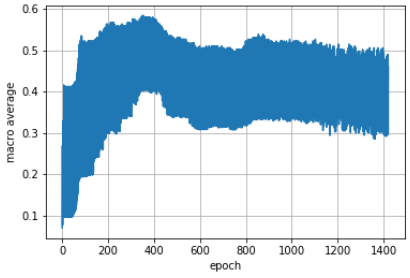
(e) F1 score for MODEL-FEATURE relation



(f) F1 score for COMPARE relation



(g) F1 score for TOPIC relation



(h) Macro average of F1 scoreS

Fig. 3: Performance of LSTM classifier with $1 * 50$ dimensional embedding vectors and single LSTM layer

The results show us that the model is partially successful in classifying USAGE, PART-WHOLE, MODEL-FEATURE and RESULT relations. It has the poorest performance on classifying TOPIC relations. We believe that this is due to only 18 examples of TOPIC relations in the dataset, which prevents the model to make relevant generalizations.

2.7 Cortical IO

Our next approach was more experimental in nature and was intended to explore ideas proposed by companies Numenta and Cortical.io. They propose sparse distributed representations of text to work with hierarchical temporal memory in semantic related NLP tasks.

BOM DOPISAL CIM HITREJE

2.8 Future plans

Preliminary results with LSTMs show promise in the future we want to test how they perform with extra semantic and syntactic features as it was shown in [Relation Classification: CNN or RNN?] that encoding distance from entities for each word, significantly improves the classification accuracy. Other source of syntactic/semantic information that we want incorporate in the LSTM networks is WordNet. We will try to find an efficient ways to represent word groups such as synonymy,hyponymy and hypernymy, and WordNet word descriptions and include this information in our prediction model.

The other area where we want to explore our options are the ideas presented by Numenta and Cortical.io. Their approach is definitely interesting, although it didn't show much promise in our initial tests. But we must take into account that we have taken really basic attributes to create the retina classification filters, namely 30 most frequent words that occur in the proximity of each relation type. We will need to explore what are our options to select words, which separate classes the most. We are also interested in implementing our own sparsely distributed representations of words and hierarchical temporal memory models to process them.

3 Conclusions

Relation classification and extraction from scientific texts presents a worthy challenge for modern day NLP systems. It is also an important step towards independent machine text understanding and intelligent unsupervised knowledge extractor systems. Our results show promise for future work with LSTMs as relation classifiers. There is still much room to improvement, and we need to compare them with some more traditional/linguistic models.

Experimenting with Cortical.io 's APIs grabbed our attention and interest to experiment with proposed ideas and maybe make our own implementation of the system. Their ideas seem to work really well for calculating similarities between

two texts, finding key words etc. and we don't see why these features couldn't be used with other models to improve the combined performance.

4 References

BOM DOPISAL CIM HITREJE

All links were last followed on October 5, 2014.