Relation Extraction

Or Zinger 200687572

Matan Vetzler 314883943

Abstract:

One of the complex problems in the NLP field is extraction of entities and relations between entities.
A lot of approaches to deal with the mentioned task exist, currently most of them try to solve the problem with neural networks models approach, especially with RNNs architectures and other sequential data models, which gets to high levels of success, although in contrast to 'other' tasks in the field of NLP which are solved much better with ML and DL, a lot of companies still using the approach of hard-coded rules based decision making.
Relations extraction is considered a hard task due to the fact that we need to figure and find the most significant patterns which will lead to identify the correct relation between entities.
Other hard task is to identify the appropriate root entities in a sentence.

We chose to try and identify the relation "Live_In" between two entities in a sentence based on the data which was provided.
Our case is a binary classification problem, where we need to find the 'pattern' and rules of the relation 'Live_In' against all other existing relations.

Our model:

We chose to implement two models in order to attack the problem from couple of sides and to investigate how efficient each method try to solve the problem of relation extraction, both are machine learning – deep learning approach. The first is just a simple linear classifier that use a binary features vector and the other one is a neural network in a form of a Bi-LSTM for that hard task.

Logistic Regression model:

This model is trained based on features vectors, which are constructed based on specific features we think are important and provide a lot of information in aspect of entities relations.
We trained a Logistic Regression model with the following parameters:

```
solver='lbfgs', multi_class='multinomial', tol=0.01, random_state=0
```

for prediction as we experienced with the MEMM task.
We built the binary features vectors based on three parts that consist of syntax vector, between word vector and entities themselves vector.

Syntax vector features -

This vector based on the shortest path between 2 root entities.
We built a dependencies tree based on the given sentence, find the shortest path between the entities, and extract from it the dependencies and POS of the nodes.
For example:

'American Airlines, a unit of AMR, immediately matched the move, spokesman Tim Wagner said'

provides the following features vector:

['nsubj', 'ccomp', 'ROOT', 'nsubj']

Between vector features -

This vector based on features we can extract from the span between the entities:

- POS of the words between the entities
- Number of entities exist between the target entities

We chose those features because we think that they provide us a lot of information on some significant distinct patterns between the entities.

It's more neural network approach to extract from them the hidden features, but in the end the Logistic Regression model shows successful results.

Entities themselves vector-

This vector based on entities themselves, their behavior and patterns.

We thought about their types, concatenate of their types, and more:

- Entities type
- Entities head POS
- Entities head dependencies
- Entities concatenate of iob (inside-outside-begin entity)

We build functions to assembly all vectors together and insert it to a simple logistic regression model.

Neural Network Model:

The other approach we have tried is to build a neural network model, like RNN to handle this sequence of words and extract the relation between them.

We have tried two models – one is based on BiLSTM as encoder and above of it an Attention layer, connected to another Acceptor. We chose acceptor because we need a fixed size vector to insert to a MLP layer

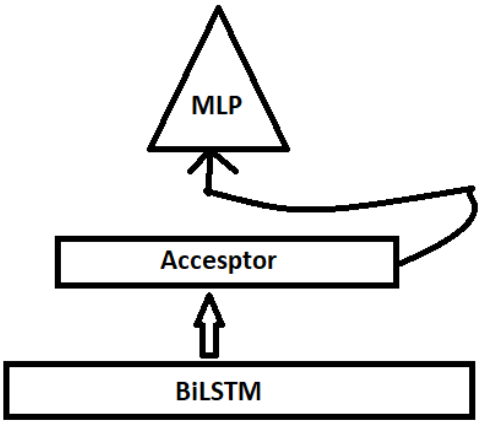
above of all. The second is based on BiLSTM as encoder and above of it an LSTM layer, again, as acceptor connected to a MLP layer.

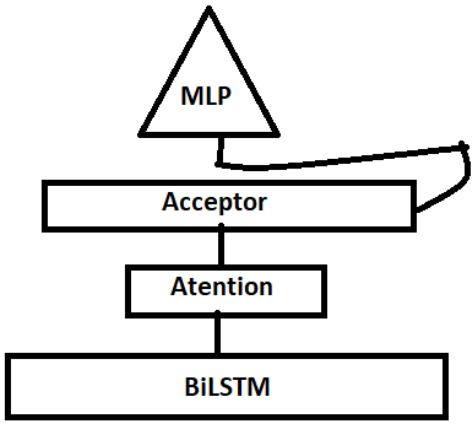
The data set consist of the given dataset “TRAIN.Annotations”, but we got another dataset called “Wikipedia.train”:

```
url=http://en.wikipedia.org/wiki/Charles_Darwin
In recognition of Darwin's pre-eminence, he was buried in <a href="/wiki/Westminste

url=http://en.wikipedia.org/wiki/John_Quincy_Adams
Adams's most important contributions to American history came before and after his

url=http://en.wikipedia.org/wiki/LaWanda_Page
<b>LaWanda Page</b> (<a href="/wiki/October_19" title="October 19" relation="birth_

Born in <a href="/wiki/Randolph_County%2C_North_Carolina" title="Randolph County, N

url=http://en.wikipedia.org/wiki/Aldous_Huxley
<b>Aldous Leonard Huxley</b> (<a href="/wiki/July_26" title="July 26" relation="bir

url=http://en.wikipedia.org/wiki/Martin_Luther_King,_Jr.
In <a href="/wiki/1953" title="1953">1953</a>, King became the pastor of the <a hre

url=http://en.wikipedia.org/wiki/Edward_Bulwer-Lytton,_1st_Baron_Lytton
<b>Edward George Bulwer-Lytton, 1st Baron Lytton</b> (<a href="/wiki/May_25" title=

url=http://en.wikipedia.org/wiki/LaWanda_Page
She was born in <a href="/wiki/Cleveland%2C_Ohio" title="Cleveland, Ohio" relation=
```

On this special dataset we build a parser to handle with it, i.e. we extract the PERSON entity from the “URL” substring and the <relation> tab that point on some relation, and the second entity from the <title> tab close to <relation> tab. We chose the relation “birth_place” to be the “Live_In” twin in this dataset. Finally, we got around 3000 and more examples, and this is enough for the model to learning.

Here are the two models' architectures:





After train the model, we got around 0.3% loss on the train. It seems that the "simple" model of BiLSTM + LSTM as much better than the BiLSTM + Attention + Decoder. We don't know why is that, but we can suppose that the loss wasn't calculated correctly due to the acceptor loss (the attention layer wasn't calculated per cell in the decoder like it suppose to, because we wanted to consider the MLP loss, and backpropged it to the network.

Parameters we chose where dim 32 for representing embedded sentence and use word2vec to represent them.

Error Analysis:

We can see that biggest problem of our implementation is differences between “spacy” library ability to extract entities and the given annotations. “Spacy”, often, return an entities that are partly similar to the given entities annotations or not at all (the given annotations aren’t in “spacy” entities list).

It hurts that a lot of recall and precisions are low.

Results: 2,118.6258

Logistic Regression model

Relation	Train Rec	Train Prec	Train F1	Dev Rec	Dev Prec	Dev F1
Live_In	64.42	40.81	49.96	58.22	36.39	44.78

Neural Network model

Relation	Train Rec	Train Prec	Train F1	Dev Rec	Dev Prec	Dev F1
Live_In				70%	25%	38%