Amos Maimon 301236287
Tamir Moshiashvili 316131259

Assignment 4: Relation Extraction

Report

Description of our system

In the past three weeks we tried many methods, machine-learning system and rule-based system, but, unfortunately, we had to give up on some of them because they did not work as we expected. The system we delivered is machine-learning approach. The model is train with xgboost's XGBClassifier model.

Since we have seen that there is a certain similarity in entities between different relations, e.g.: 'live_in' & 'work_for' are both use person for OBJ1 and GPE for OBJ2, we decided to train the model on all of the relations in the train corpus, so it could distinguish between such similarities. In predict time we return only the pairs of entities that was predicted with the relation of 'live_in'.

We have started to use the following features, but later we dropped some of them to get better performance in F1, recall, and prediction.

Features:

- NER
- POS
- DEP
- IOB
- prefixes and suffixes of the different phrases
- behavior of tokens before and after the candidate-phrase, for example, their tags (POS, DEP, the word it-self)
- lemmas
- distance between the two objects
- capitalized and titled
- upper and lower case

Finally, the chosen features used in our program was determined by checking the behavior of the classifier on the given features.
We also tried to optimize the classifier's success by playing with the parameters of the model and finally set the best ones we have found.

Our system being trained on the corpus with xgboost's XGBClassifier and with some of the features that mention above. After the model will write two files to later use – feature map file, and model file. Our model can later be loaded from these files and predict on other files without the need to re-train.

Amos Maimon 301236287
Tamir Moshiashvili 316131259

Error analysis

As mentioned above, we had many dead-end roads in our process. We deal with a lot of problems and challenges in the way. We will try to describe as many as we can remember from these.

Spacy entry text problem

We found that many entities from spacy have text that have a small and annoying differences between the annotation file result. As a result, many examples appeared in a slight change as both FN and FP. Examples:

- "Mrs. Higgins" return from spacy as "Higgins"
- "Umbria" return from spacy as "Umbria province"
- "Hakawati Theatre" return from spacy as "the Hakawati Theatre"

This kind of mistakes was decease our recall and our prediction rates, also in training-time this caused us to miss about half of the relations in the train.annotation file. We tried many methods to reduce the recurrence of these errors. We tried to use both of spacy.ents and spacy.noun_chunks to maximize entities group for the process sentence. We tried to use a black-list of the words that spacy put inside of the entity (e.g. 'the'). In the end we decided that this type of error is not really a mistake of our model, because the answer is still correct for the pair of entities, so we stopped comparing the text of the entities and started to search for the best containment between the answers and the prediction.

Spacy tagging mistakes

We found that spacy tagged wrongly some fields we used in our model. e.g. some entities that their "ent_type" should be 'GPE' tagged with tags like 'PRESON' or 'NORP'. Because we think the 'ent_type' is the most effective feature for 'live in' relation, these mistakes are critical for our model performance. We tried to use Wikipedia API to double check spacy tagging. It was a trade-off with the time that the model runs, because every query for Wikipedia have a delay time to get back an answer. We failed to achieve a major improvement in the tagging correctness to justify this delay. Because of this - we dropped this approach.

After we tried to use other tools, and also failed to achieve a major improvement with them, we decided that our model would handle this mistakes by itself.

Relation	Dev Recall	Dev Prec	Dev F1	Train Recall	Train Prec	Train F1
Live in	0.5954	0.3305	0.4250	0.6538	0.4594	0.5396