Natural Language Annotation

Entity Linking for Queries

Students:

Felix Zhu
15-907-835
@student.ethz.ch

Jakob Hasse
15-907-835
hassej@student.ethz.ch

Qin Wang
15-944-424
qwang@student.ethz.ch

Robert T
15-907-835
@student.ethz.ch

May 26, 2016

1 Approaches

The following approaches are implemented in our projects.
SMAPH-2, **Simple Heuristics Method** and **LSTM-CRF based method**.

1.1 SMAPH-2

We implemented the SMAPH-2 annotator mentioned in the paper [1]. However, we found that we had way too many bindings and the gathering of the reults took far too long. To proof the work we provide the files in the package.

1.2 Simple Heuristics Based Approach

This approach takes the spotted entities from the wikisense-api and tries to bind them to the query text. As heuristic the commonness provided by the spotting URL was used, since we had lot of cases in which the gold standard was no annotation but the spotting actually provided entities. Also, to match the entities to the exact mention in the query the Edit-Distance between the wikipedia title of the entity and the matching query part for each possible position in the query was used.

1.3 LSTM-CRF Based Approach

Named entity recognition tasks and entity linking tasks are challenging, most state-of-art approaches rely heavily on carefully constructed features and domain-specific knowledge. To some extent, this is because only a very limited amount of supervised training data are available for most languages. However, there are still lots of posibilities and processes on the neural side. A research group (Sandeep Subramanian et al. 2016) recently published their work on nueral architectures for named entity recognition and claimed their models obtain state-of-art performance without resorting to any language specific knowledge. Our LSTM-CRF based approach is strongly inspired by this paper and did modifications for our specific problem.
Our intuition is to use a neural model to capture the way a sequence of words form a named entity, we believe word vectors pretrained from wikipedia corpus somehow contain the information to identify an entity because words inside a named entitiy should have higher scores in the co-occurrence matrix, where the word vectors are generated from. In order to avoid overfitting and ask the model to learn the way words form an entity instead of remembering all entites appreared, we also introduce a training data generation method, which enlarges our training data significantly and improves our result somehow.

1.3.1 LSTM-CRF model

Our LSTM-CRF takes a sequence of word-vectors from pre-processed queries and generate a BIO label for every word-vector. It is based on Subramanian's LSTM-CRF model and their Python-Theano codes are used. Long short term memory network is a model that is widely used for sequential data. Unlike RNNs, it is designed to have the ability to capture long-range dependencies. This specific characteristic is achieved by using several gates that control the proportion of the input to give to the memory cell, and the proportion from the previous state to forget. In our project, we use a bidirectional LSTM to decide the tag of a word based on information from both sides. Above the LSTM layers, a conditional random field layer is used to model tagging decisions jointly.

1.3.2 Data pre-processing

Before translating our sequence of words into word-vectors, there are lots of things we can do to improve the final result.
Wash queries: We wash our queries by lowercasing all words, deleting non alphanum'- characters and split 's 'd 'll etc.
Infer spaces: Some queries are given without spaces, which could be a big problem because the search engine cannot autocorrect this type of queries, and even worse, we will lose lots of information because we cannot translate a long long word into its word-vector representation. In order to infer spaces, we use a list

of words by frequency, and use dynamic programming to infer spaces. However, in order to avoid breaking misspelled words into small parts, we also introduce a local misspelling detection function. Now we are able to deal with input like this "Thisisasuperlongquerywithsomemispppelling".
Spelling correction: A wiki word list from the Glove.100d.txt is generated, which is trained from the wikipedia corpus. In addition, a common misspelled word list is used. If any word is not in the list or in the second list, it will be sent to search engines and be replaced either by the auto correction result or the most similar one in the bold text.
Translate: 100d Word-vectors pretrained from wikipedia corpus by Stanford GloVe are used.

1.3.3 Training data generation

In order to avoid learning per-entity features, we enlarge our dataset by replacing known named entities with new wiki titles. For example, the query "washington county minnesota public housing" and its bindings may generate "Barack Obama TV set" and its related bindings. By doing so, we slightly improved our results, but also significantly made the training time longer. In our specific case where we do not have access to GPU clusters, we sample words from the list of wiki titles that overlaps with any word in the final test set and limit the size of the generated training set to shorten the training time.

1.3.4 Piggyback entity linking

Inspired by the SMAPH method, we also use several sources including search engine results and wikiMedia APIs to link our entities. We first try to directly find the result from wikiMedia API, if a wikipedia title matches our entity literally and without ambiguity, we then link it. Otherwise, we will search the entity in bing search engine with prefix 'language: en site:en.wikipedia.org'. If any wikipedia page title/abstract overlaps with our entity, we link it. If none of the previous steps managed to link the entity, we will delete this entity as a correction of the LSTM-CRF model.

1.3.5 Interface

We find that even if we have the correct NER and linking, it is still possible that we gain no points. This is because the mapping from original queries to corrected ones is not a simple process, spaces may be added inside a word, tokens can be changed because of misspelling. We introduce a function BIO2Queries to map entity BIOs to final start-end numbers by greedily compare the edit distances.

2 Conclusion

Results from LSTM-CRF model:
C2W mac-P/R/F1: 0.694/0.486/0.460 mic-P/R/F1: 0.555/0.447/0.495 TP/FP/FN: 183/147/226 std-P/R/F1: 0.400/0.427/0.409
A2W-SAM mac-P/R/F1: 0.645/0.448/0.419 mic-P/R/F1: 0.491/0.411/0.447 TP/FP/FN: 168/174/241 std-P/R/F1: 0.424/0.432/0.412

The final score is not significantly higher than the baseline. The heuristics based approach is too simple to provide higher scores while the LSTM-CRF based approach may still don't have enough training data. Further improvement are discussed in the next sction.

3 Further improvement

We believe the LSTM-CRF results can be better if we did the training data generation more carefully by replacing entities with new entities in the same domain. For example, replace "Barack Obama" with "Thomas Muller" instead of "TV set".
Ensemble methods should be the final layer in a competition to further improve precision. However, due to the limit of time and computing power, we did not implement this. In addition, we trained our LSTM model

on local laptop for a maximum of five hours and choose the one with best performance on the dev set. It might be possible to gain better performance if we trained for longer.

4 References

[1] Marco Cornolti, Paolo Ferragina, Massimiliano Ciaramita, Stefan Rd, Hinrich Schtze, *A Piggyback System for Joint Entity Mention Detection and Linking in Web Queries*, IW3C2, 2016.

[2] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, Chris Dyer, *Neural Architectures for Named Entity Recognition*, arXiv, 2016.