OpenIE for Slot Filling at TAC KBP 2017 - System Description

Samuel Broscheit **Kiril Gashteovski** **Martin Achenbach**
Data and Web Science Group at the University of Mannheim
{ k.gashteovski, broscheit } @uni-mannheim.de
machenbach@mail.uni-mannheim.de

Abstract

Open Information Extraction (OIE) extracts triples, i.e. subjects, relations and objects, from natural language text (Banko et al., 2007). Our goal was to explore the usefullness of OIE for slot filling, for which we used the OIE system MinIE (Gashteovski et al., 2017).

In our approach, we learn embeddings for triples using 400 million OIE triples extracted from two large corpora, i.e. the New York Times corpus (Sandhaus, 2008) and Wikipedia. We also use the OIE system on the slot filling data, i.e. the TAC KBP corpus, to create candidates triples. For a given query, the candidate triples are then scored by a classifier that receives the query subject and predicate as well as the candidate triple represented by the embeddings.

1 Introduction

Slot filling is the task of, given a query that consists of a query entity and a relation, to find possible answers that complement the query. E.g. for query entity "Bill Gates" and the relation "holds shares in" we want to find all the companies that complement this query. In the TAC KBP slot filling task the challenge is to extract the answers from a given corpus for a predefined set of relations.

The challenge that comes with this task is that even very simple relations like "is born in" can come in a huge variety of formulations in natural text. This makes it difficult for rule-based approaches as well as for supervised methods to gain ground in this task. This is why an obvious approach is a to use combination of both and bootstrap more training data. Open Information Extraction (OIE) is the task of extracting schema-free relations and their arguments in the form of triples or n-ary tuples from natural language text in unsupervised manner. We exploit the fact that we can learn embeddings for triples from an Open Information Extraction system, which provides us feature representations for the elements of triples, and also, by using similarity operations for vectors, also a means to represent the similarity between triples. Therefore the task is reduced to learning to align the OIE embeddings to the predefined set of relations.

2 MinIE

Consider the sentence *"AT&T, which is based in Dallas, is a telecommunication company"*, an OIE system aims to extract triples such as:

- ("AT&T"; "is based in"; "Dallas")
- ("AT&T"; "is"; "telecommunication company")

A common problem with OIE systems is that both the relations and the arguments can be overly specific. The assumption is that the underlined words in the following example can be dropped without damaging the basic information carried in the triple.

- ("The great Paul Dirac"; "worked jointly with"; "Boris Podolsky")
- $\rightarrow$ ("Paul Dirac"; "worked with"; "Boris Podolsky")

MinIE is an OIE system that tackles this challenge, by (1) identifying semantic information and preserving it as semantic annotations; (2) removing words that are considered overly-specific, but preserving them as meta data of the triple (Gashteovski et al., 2017). This process is called minimization.

2.1 Semantic annotations

In MinIE, the following semantic annotations are attributed to each triple:

Polarity: determines whether a triple is positive (+) or negative (–) and rewrites it:
("Bill Gates"; "is not CEO of"; "Microsoft") $\Rightarrow$
("Bill Gates"; "is CEO of"; "Microsoft") (–)
Modality: determines whether a triple is certainty (CT) or a possibility (PS):
("Aliens"; "may have landed in"; "Area 51") $\Rightarrow$
("Aliens"; "have landed in"; "Area 51") (PS)
Attribution: annotates the source (and its polarity/modality value) providing the information carried in the triple:
("Orly Taitz"; "said that"; "Barack Obama may have been born in Kenya") $\Rightarrow$
("Barack Obama"; "have been born in"; "Kenya")
Factuality: (+, PS)
Attribution: (Orly Taitz, +, CT)
Quantities: phrases expressing some sort of quantities. For example, the phrases *"9 cats"*, *"all cats"*,

"almost about 100 cats" are all rewritten as *"QUANT cats"*.

2.2 Minimizing modes

After the triples have been semantically annotated, MinIE employs different levels to minimize relations. Here we used the **MinIE-D (Dictionary mode)** which drops those words, that modify noun phrases (adjectives, adverbs, etc.) but keeps patterns that are found in a dictionary of multi-word expressions. It also filters out extractions containing subordinate clauses and drops words which are considered "safe to drop" (e.g. determiners, adverbs modifying the relation) (Gashteovski et al., 2017).

2.3 Preprocessing

MinIE uses ClausIE (Del Corro and Gemulla, 2013) as an underlying system, which produces high precision/recall extractions. MinIE's preprocessing pipeline annotates POS tags, NER tags, dependency parse using CoreNLP (Manning et al., 2014).

3 Slot Filling Model

3.1 Triple Embeddings

In our approach, we learned embeddings for triples using 400 million OIE triples. For training the embeddings, each OIE relation was represented as a concatenation of its lemmas, e.g. *is member of* was transformed into the token *rel:be_member_of*. The arguments, i.e. subjects and objects, were represented with the prefixes "subj" and "obj" respectively. Their words are not concatenated, unless they form a named entity. The triple embeddings were trained with CBOW (Mikolov et al., 2013) using a maximal context window over the whole triple, and no distance discount and feature size 200. We prune very long relations and tokens with a count lower than 3, which leaves us with a vocabulary size of approximately 13 million entries.

Our goal was to use the embeddings as a means to align OIE relations to TAC-SF relations in an unsupervised manner. For each relation from the TAC queries we manually selected a representative OIE relation. For example, for *per:charges* we chose *rel:be_charge_with*. As can be seen in Table 1 the embeddings indeed capture different lexical variations of the *per:top_member_employee_of* relation, which we mapped to *rel:be_president_of*.

Because of cases in which we only found the inverse of the TAC-SF relation, e.g. for *gpe:employees_or_members* we chose *rel:be_member_of*, but then use a binary indicator that signals an inversed mapping.

To map a subject, relation or object string to our embedding vocabulary, we use a search index over the vocabulary. This is mostly useful for relations, because in this way we can find embeddings that might have slight lexical variations. For example, for the string

rel:be president of
rel:be chairman of
rel:be vice president of
rel:be former president of
rel:be founder of
rel:be director of
rel:be head of
rel:be executive vice president of
rel:be former chairman of
rel:be chief executive of
rel:be vice-president of

Table 1: Nearest neighbours for the relation embedding rel:be president of

"sell minority stake in company to" we find the embedding *rel:sell_minority_stake_to*

3.2 Training data

We used the assessments of the TAC 2016 slot filling data as training data. First we process the source data with MinIE. Then apply a simple entity disambiguation where we use the anchors from links within Wikipedia to disambiguate strings to an entity.

After processing the source data with MinIE we aligned the triples from MinIE with the training data. For this, we aligned the filler provenance from the assessed answers with the object from a triple using the text offsets. The query subject, on the other hand, had to string match with the triple subject in either its raw surface form or the disambiguated form. To get more coverage we align answers with triples not only when they exactly match with their boundaries but also if they only overlap.

3.3 Model

For a given query $q = (s_q, r_q)$, with query subject s_q and query relation r_q, and a candidate triple $c = (s_c, r_c, o_c)$, with subject s_c, relation r_c and object o_c, we want to decide if the candidate triple provides a slot filler.

Because we wanted to exploit the negative examples in the training data we did not cast this as a multi class classification over the candidate triples. Also, because of the small size of the training data, we did not attempt to learn a compatibility of the query and the candidate triple. Therefore we reduce this to a binary prediction, that, given some compatibility between the query and the candidate triple computed by a function $\phi(q, c)$, if this triple provides a slot filler, or not. However, the TAC-SF relations actually share a fine grained set of slot filler types, e.g. $T_+ = \{city, country, state\_or\_province, gpe, org, per, noun, date, number, website\}$ which had a very uneven distribution in our training data. We found that casting the prediction as a multi class prediction over the slot filler types improved the classifier performance, most likely because it is easier to capture the statistics of the slot filler types in our training data. Therefore, we used the set T_+ as target labels and extended it with a negative class $T_- = \{-\}$, i.e. $T = T_+ \cup T_-$. Therefore our classification could be formalized as

$$\arg\max_{t \in T} \ p(y = t|\phi((s_q, r_q), (s_c, r_c, o_c)))$$

Let $emb(s)$ be the function that returns the embedding x_s for a string s from either one of s_q, r_q, s_c, r_c, or o_c. Let $a \circ b$ be the element wise product of two vectors a and b, and let $sim(a, b) = a \cdot b/(\|a\| * \|b\|)$ be the cosine similarity.

The inputs for the classifier are:

- Features for the compatibility between query and triple, that we manually selected through ablation
  - $sim(emb(s_q), emb(s_c))$
  - $sim(emb(s_q), emb(o_c))$
  - $sim(emb(r_q), emb(r_c))$
  - $sim(emb(r_q), emb(o_c))$
  - $emb(s_q) \circ emb(r_q)$
  - $emb(r_q) \circ emb(r_c)$
  - $emb(r_q) \circ emb(o_c)$
- A one-hot encoding of the query relation
- A one-hot encoding of the query slot filler types

For the relations that have a number, website or date as slot filler type, the object vector is a binary vector of the same size as the embeddings, which is 1, when a heuristic classifies it as date, number or website, else 0.

We trained a multilayer perceptron with rectifier linear units as activations and a final softmax non-linearity after the last layer. We did a grid search over the number of hidden layers, hidden layer size (64, 128, 196, 256), activations (sigmoid, tanh, rectifier linear units). We found that 3 layers with hidden layer size 128 and rectifier linear units as activations to be optimal. The weights were initialized with Xavier normal initialization (Glorot and Bengio, 2010). We used Adadelta (Zeiler, 2012) to reduce the negative log-likelihood loss and dropout with p=0.3 as regularization. To increase precision and to exploit the whole training data we used ensembling with models that were trained on different splits of the data with 4-fold cross validation.

During training and prediction, the classifier is presented with the triple in its natural occurrence (subject, relation, object) and also the inverse (object, relation, subject). During training, depending on the direction of the representative triple we assign the wrongly flipped triple the negative class. During prediction for the submission we map the predicted classes to: GPE, PER, ORG and STRING and discard predictions with low confidence.

On the validation data the classifiers achieved 67.9% accuracy (+/-2.9% std) after training.

4 Error analysis

There were many factors contributing to our low results for this challenge. We have identified several types of errors that we found in the results:

4.1 OIE

Quantities: for training the embeddings, we have used MinIE-D as-is, without replacing the quantities with their original phrases.

Coreference resolution: many of the errors that we observed were caused by coreference. MinIE does not have coreference resolution in its pipeline. We will investigate if running coreference resolution upfront on the whole documents will improve the results.

Dependency parse errors: MinIE uses dependency parse in its pipeline. An error in the dependency parse propagates to an error in the extractions. Long or unusual sentences pose a problem and especially the discussion forums contain many difficult sentences.

NER errors: MinIE used NER for some of the extraction it makes. This means that if the NEs are missed, these extractions are either not produced at all, or the NE spans are wrong.

Some patterns not covered by MinIE: we have observed several syntactic patterns that would have produced nice results for some queries. These were patterns that MinIE currently doesn't cover. If they are established to be generic enough, they can be used on top of any OIE system for producing more high-precision extractions.

4.2 Model

Training data: We definitely had too few training data. We could observe that we had an acceptable performance on classes for which we had the most data. We could tackle this by improving our method to align the training data with the MinIE triples. In general we have to consider also other means to create more training examples. Also, as we did not tune the preprocessing pipeline for coverage and did not use entity linking, due to time constraints, our coverage estimated on the 2016 slot filling data was very low: 17.39% on hop0 queries and 21.36% on hop1 queries with a lenient boundary match.

Training method: Explore different models, so we train over all of the candidate triples.

4.3 General

Entity linking: within our pipeline we haven't implemented entity linking, which should improve the results.

Inferred information: some answers that our current setup does not cover, are answers for which the relation between the query entity and the answer has to be assumed.

Metadata as answers: we have observed that in some

Team	AP
Stanford	21.86
UNIST SAIL	14.71
Zhejiang University	10.68
ours	2.89

Table 2: Results for Slot Filling; reported in average precision

of the queries, the answers are in the meta-data.

Easy cases: By examining the errors our system made we found many easier cases, e.g. the relation *org:alternate_names* that can be easily handled without a model. Also, due to a preprocessing error we did not train our final system on numbers and date slot filler types.

5 Conclusion

There could be huge amounts of relation types in natural language text. The biggest advantage of the OIE extractions is that they are not bound to relational schemes, nor to certain argument types, which can provide very diverse training data. These can be boosted by supervised methods for a specific task or specific relation types of interest.

This was our first entry to the slot filling task of TAC KBP. Our system suffered from the initial loss of coverage so our rank is very low, see Table 2. We hypothesize that by addressing the issues from the error analysis, we can significantly improve the results.

References

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, IJCAI'07, pages 2670–2676. http://dl.acm.org/citation.cfm?id=1625275.1625705.

Luciano Del Corro and Rainer Gemulla. 2013. Clausie: clause-based open information extraction. In *Proceedings of the 22nd international conference on World Wide Web*. ACM, pages 355–366.

Kiril Gashteovski, Rainer Gemulla, and Luciano Del Corro. 2017. Minie: Minimizing facts in open information extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 2620–2630.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10). Society for Artificial Intelligence and Statistics*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*. pages 55–60. http://www.aclweb.org/anthology/P/P14/P14-5010.

Tomas Mikolov, Kai Chen, G.s Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space 2013.

Evan Sandhaus. 2008. The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia* 6(12):e26752.

Matthew D. Zeiler. 2012. Adadelta: An adaptive learning rate method. *CoRR* abs/1212.5701.