Saarland University Spoken Language Systems at the Slot Filling Task of TAC KBP 2010

Grzegorz Chrupała Saeedeh Momtazi Michael Wiegand Stefan Kazalski
Fang Xu Benjamin Roth Alexandra Balahur Dietrich Klakow
Spoken Language Systems
Saarland University
D-66123 Saarbrücken, Germany
lsv_trec_qa@lsv.uni-saarland.de

Abstract

For the slot filling task of TAC KBP 2010 we developed as a system a simple pipeline architecture whose main components are a two-stage retrieval module and a relation extraction module. We use word-cluster features in the system as a method of achieving generalization by exploiting raw text.

In the relation extraction module we use distant supervision in order to extract training examples from a partially completed knowledge base. The best-ranked run of the full system achieves an F-score of 13.6% on the official test queries.

1 Introduction

In the slot filling task of TAC KBP 2010 the objective is to develop a system which given an entity (person or organization) fills in missing information about it in a knowledge base. In this paper, we present the overview of our submission. The main points of interest of our system are that we (i) make limited use of task-specific external resources (ii) experiment with the use of *word clusters* as useful features at several stages of processing (iii) rely on *distant supervision* in the relation extraction module.

Our system follows a simple pipeline architecture. First, we retrieve documents related to the entity in the input query using an off-the-shelf document retrieval component (Section 2.1). Second, we process the documents, segmenting them into sentences, tokenizing and annotating them with named-entity labels (Section 3). Third, we translate each missing slot type (attribute) for this entity into a query and perform sentence retrieval on the union of sentences from the relevant documents (Section 2.2). Fourth, we decide which named entities and other sentence fragments in the retrieved sentences are plausible fillers of the required slot type using a relation extraction module. The relation extraction module is trained on examples created by extracting entity-slot pairs from the TAC knowledge base and retrieving sentences in which they occur from the corpus (Section 4).

The best-ranked run of the full system achieves an F-score of 13.6%. We present and discuss the results of the different configurations of our system in Section 5.

2 Document and Sentence Retrieval

2.1 Document Retrieval

The task of document retrieval is an important part of a slot filling system as it provides the input to sentence retrieval and the relation extraction component. Its purpose is to select relevant documents with regard to a specific query from a large corpus. Thus, the search space for subsequent modules is reduced which is an important pre-requisite for more computationally intensive algorithms – as they are common for relation extraction – to be used.

The document retrieval component of our slot filling system uses the *INDRI* toolkit[1] as a retrieval engine which is part of the *Lemur* project. It improves language model-based retrieval by including inference networks. Our inference network includes the target entity and chunks extracted from passages with a mentioning of the entity in the reference document. We estimated the weights in the inference network with the dice-coefficient. This makes the retrieval process more robust against noisy expansion terms. Based on our experiments, we achieve the best performance by retrieving the top 50 documents.

2.2 Sentence Retrieval

Although document retrieval is helpful to reduce the search space, entire documents are still too large to be processed by our relation extraction component. Therefore, further irrelevant information within a document is removed by applying sentence retrieval. It re-ranks all sentences contained in the relevant documents. The most highly ranked sentences are then used as input for relation extraction. Based on our experiments, we achieve the best performance by retrieving the top 40 sentences.

[1] `http://www.lemurproject.org/indri`

We use a language model-based sentence retrieval module for our slot filling system where the probability $P(Q|S)$ of generating the query Q conditioned on the observation of the sentence S is first calculated, and thereafter sentences are ranked in descending order of this probability. $P(Q|S)$ is estimated by a unigram model based on the query terms [Ponte and Croft, 1998]:

$$P(Q|S) = \prod_{i=1}^{M} P(q_i|S) \quad (1)$$

where M is the number of query terms and q_i denotes the i^{th} query term in $Q = \{q_1, q_2, \ldots, q_M\}$. The query in our task consists of three elements:

- **The target named entity:** The target named entity is the string denoting the entity of the target, i.e. either a person or an organization, as provided by the input file of the slot filling task. For example, when the system is to find the date of birth of Albert Einstein, the target named entity is "Albert Einstein". By including that entity in our query, we ensure to retrieve sentences dealing with the target.

- **The expected named entity:** The named entity tag of the slot value that is to be found is called the expected named entity. In the above example, "DATE" is the expected named entity. This part of the query helps sentence retrieval to retrieve sentences which include the named entity tag of the slot value. The named entity annotation for each sentence is provided by a named entity recognizer run on the output of document retrieval prior to sentence retrieval.

- **The expansion terms:** The expansion terms are predefined words that are predictive for specific slot types. In the above example, "born"

is one of the words that are very likely to appear in relevant sentences. As a result, sentence retrieval assigns a higher rank to the sentences that contain this predictive term.

We also found that the consideration of the document from which each sentence originates can play an important role for ranking the sentences. This is, in particular, true for ties, i.e. different sentences which receive the same score from sentence retrieval. By also including the score from document retrieval in sentence retrieval, we should (at least) be able to discriminate between these sentences if they originate from different documents (as the ranking scores from document retrieval should be different). Thus, we compute the final score of sentence retrieval as the product of the following two scores:

- $P(Q|S)$ as described before
- $\frac{1}{R}$, where R is the rank of the document that contains sentence S.

3 Named Entity Recognition

Both the sentence retrieval and the relation extraction components of our system need access to named entity (NE) labels specific to the slot filling task. We thus develop a custom NE labeler which detects the relevant types. As a starting point, we take the BBN corpus as training data [Weischedel and Brunstein, 2005]. This resource consists of the Wall Street Journal text from Penn Treebank labeled with a fine-grained set of named entity and pronoun reference labels. We map the BBN label set to the coarse-grained set shown in Table 1. We then train a perceptron sequence labeler [Collins, 2002] on the BBN training data. We do not use features which rely on a part-of-speech tagger or chunker. We do, however, use word-cluster features, which have been shown to be very useful for Named Entity Recognition (NER) [Miller et al., 2004, Ratinov and Roth, 2009, Chrupała and Klakow, 2010, Turian et al., 2010]. We use the Brown et al. [1992] algorithm to create a flat partition of word types from a 17-million-word portion of the Reuters corpus[2] into 500 clusters. We then use cluster ids in combination with standard contextual and spelling features in the training of the NE labeling model. The overall performance of the NE labeler on section 22 of the BBN corpus is shown in Table 2.

BBN-mapped labels	
CARDINAL	DATE
CITY	COUNTRY
MONEY	ORDINAL
PERCENT	STATE-OR-PROVINCE
POLITICAL	ORGANIZATION
PERSON	QUANTITY
Extra labels	
RELIGION	JOB-TITLE
URL	

Table 1: NE labels.

Precision	Recall	F-measure
91.18	92.15	91.66

Table 2: NER results on BBN section 22.

Additionally, we use simple heuristics to label sentences with three additional labels for which we do not have sufficient training data in order to include them in the statistical model. For RELIGION and JOB-TITLE, we created gazetteers from online resources, such as Wikipedia, and we use gazetteer lookup in order to assign labels. We also label URLs by matching a character pattern.

[2] http://trec.nist.gov/data/reuters/reuters.html

4 Relation Extraction

4.1 Distant Supervision

In the basic configuration of the relation extraction module we used the distant supervision approach presented in [Mintz et al., 2009]. In this scenario, the positive training examples are pairs (entity, slot value) present in the knowledge base (KB) while the negative examples are such pairs which do not occur in the KB.

Each example is converted to a feature vector using the following procedure. For a given (entity, slot value) pair we retrieve the sentences which contain mentions of both elements of the pair from the text corpus (for speed we limit the sentences to those in the first most relevant document as ranked by the document retrieval component). From each occurrence of the pair in a sentence we extract a number of local features (cf. Section 4.2); the global feature vector is the sum of the local vectors extracted from each sentences.

More formally, let the i^{th} training example x_i be the triple (e_i, s_i, W_i) where e_i is the entity, s_i is slot value, and W_i is the set of sentences in which they both occur. The local feature function extracts features from a single occurrence of e_i and s_i in the sentence $\mathbf{w}_j$: $\phi(e_i, s_i, \mathbf{w}_j)$. Thus the global feature function is defined as:

$$\Phi(e_i, s_i, W_i) = \sum_{\mathbf{w}_j \in W_i} \phi(e_i, s_i, \mathbf{w}_j) \qquad (2)$$

We use the KB and the text corpus to create such training examples for all the slot types. Then for slot type τ_i we train a binary linear classifier using all the triples which instantiate relation τ_i as positive examples, and the triples which instantiate other extraction $\tau_j \neq \tau_i$ as negative examples.[3]

For training the classifiers we use the averaged perceptron algorithm [Freund and Schapire, 1999, Collins, 2002] with a constant learning rate of 0.01 which we run until training classification F-score reaches 97.5%, for up to a maximum of 100 iterations. We remove any features appearing in only one example of the training data.

At test time, we retrieve sentences relevant to the query entity and slot type, and use a manually created mapping from slot types to named entity labels in order to extract entity - candidate slot value pairs. We aggregate all the sentences for a particular slot value and create test feature vectors as described above. For single-value slots, we return them as correct if they are classified as positive, otherwise we return NIL. For list-valued slot types we rank the candidate values in descending order of the score assigned to them by the classifier, and return the n best items. For each list-valued slot type we set n_{τ_i} as the maximum number of answers judged as correct for any entity for slot type τ_i in judgment files from TAC KBP 2009 and development data for TAC KBP 2010.

We still need to choose the document id to return together with the slot value. We do this by splitting the training example (e_i, s_i, W_i) into as many sub-examples as there are documents containing sentences in W_i; that is for each document D_k such that some sentence in W_i is in D_k, we create $(e_i, s_i, \{\mathbf{w}_j | \mathbf{w}_j \in D_k \wedge \mathbf{w}_j \in W_i\})$, and apply the classifier to each of the sub-examples. We choose the document id k which gives the highest score.

4.2 Features

We used a number of simple surface-based features. In order to improve generalization, we used word-cluster features, similar as for our NER component.

[3] For efficiency we limit the number of positive and negative examples per classifier to a maximum of 3200 and 6400 respectively.

In the case of relation extraction we used a hierarchical clustering of word types with 3200 classes at the leaves of the hierarchy, trained on the same Reuters news data.[4] Following previous practice [Ratinov and Roth, 2009, Turian et al., 2010], we used cluster id prefixes of lengths 2, 6, 10 and 20.

The local features $\phi(e_i, s_i, \mathbf{w}_j)$ we used are:

- The conjunction of the direction of relation (left if e_i precedes s_i in $\mathbf{w}_j$, right otherwise)
- Log distance between target e_i and slot value s_i in $\mathbf{w}_j$
- Words in target entity e_i, slot value s_i, between e_i and s_i, up to 3, words before and after them
- Brown cluster id prefix of length 2, 6, 10, 20 of the above words
- Bigrams of Brown cluster ids as above
- Trigrams of Brown cluster ids of words between target e_i and slot value s_i.

We used the basic configuration described above in the first submitted run.

4.3 Using Annotated Data

In addition to using the KB as the source of training data we also attempted to make use of the annotated data provided by the organizers.

We used:

- Test data from the 2009 slot filling task
- Development data from the 2010 task
- Participant annotated data from the 2010 task.

[4]We used the hierarchical clustering made available by J. Turian at `http://metaoptimize.com/projects/wordreprs/`

The annotated data from 2009 consists of queries associated with human judgments specifying whether slot values for a slot type for a particular entity extracted from a particular document is correct, inexact or incorrect. Judgment files from 2010 only list correct slot values together with the document ids: in this case we assume that any slot value not listed is incorrect. This data still does not provide sentence-level annotations but unlike the distant supervision scenario, it does provide document-level judgments and we thus thought it would usefully complement the basic system configuration.

We create the training examples by first running the system components up to sentence retrieval and candidate slot value extraction using the queries from the pertaining annotated dataset we want to use. At this point we generate the training feature vectors from the extracted candidate slot values in the retrieved sentences. We assign examples to the positive or negative set based on the judgments associated with the queries.

We then use the training examples generated in this fashion either as the only source of supervision, or jointly with the training data obtained from the KB as described in Section 4.1. The joint training set up was submitted as run 2.

5 Results

For the slot filling task of TAC KBP 2010 we considered the following configurations:

- KB - basic system configuration, with training examples obtained from the KB
- An - system version using only training examples obtained from annotated data (Section 4.3)
- KB+An - system version using the union of the training examples KB and annotated data

Model	Run	Prec.	Recall	F-score
KB	1	20.07	**10.34**	**13.65**
An	-	31.48	04.93	08.53
KB+An	2	29.85	07.73	12.28
ET+KB+An	3	**32.92**	07.73	12.52

Table 3: Scores of different system configurations on the 2010 test data.

- ET+KB+An - same as KB+An, but additionally sentence retrieval uses expansion terms (Section 2.2).

Table 3 shows the scores obtained by four configurations of our system on the official 2010 test data. Three of those configurations were submitted as runs.

The basic configuration KB achieved the best overall performance. The other configurations trade higher precision for much lower recall, resulting in a lower F-score. Training only on examples from annotated data (An) resulted in an especially large drop in recall. Somewhat surprisingly, combining KB with An (KB+An) does not improve on KB alone. It is not completely clear why this is the case, but we suspect that part of the problem may be that we are not optimizing F-score directly, but rather the classification accuracy, which makes it hard for the system to maintain a proper balance between recall and precision without manual tuning. Finally, adding expansion terms to KB+An improves precision without harming recall and thus improves the F-score slightly.

6 Conclusion

We have developed a slot filling system for TAC KBP 2010 which uses a simple pipeline architecture whose main components are a two-stage retrieval module and a relation extraction module. We have used word-cluster features in the system as a method of achieving generalization by exploiting unlabeled data. In the relation extraction module we used distant supervision in order to extract training examples from a partially completed knowledge base.

Slot filling is a complex and difficult task, and it is by no means trivial to achieve good performance with shallow methods and while avoiding dependence on external resources. In future, we would like to investigate whether we can improve our system by using linguistic analysis, such as dependency parsing, and by implementing more sophisticated statistical models which take into account the dependencies between different relation types.

References

P. F. Brown, R. L. Mercer, V. J. Della Pietra, and J. C. Lai. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, 1992.

G. Chrupała and D. Klakow. A Named Entity Labeler for German: exploiting Wikipedia and distributional clusters. In *Proceedings of the Conference on International Language Resources and Evaluation (LREC)*, pages 552–556, 2010.

M. Collins. Discriminative training methods for Hidden Markov Models: Theory and experiments with perceptron algorithms. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1–8, 2002.

Y. Freund and R. E. Schapire. Large margin classification using the perceptron algorithm. *Machine learning*, 37(3):277–296, 1999.

S. Miller, J. Guinness, and A. Zamanian. Name tagging with word clusters and discriminative training. In *Proceedings of the Human Language Tech-*

nology Conference of the North American Chapter of the ACL (HLT/NAACL), pages 337–342, 2004.

M. Mintz, S. Bills, R. Snow, and D. Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL/IJCNLP)*, pages 1003–1011, 2009.

J. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 275–281, 1998.

L. Ratinov and D. Roth. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Conference on Natural Language Learning (CoNLL)*, pages 147–155, 2009.

J. Turian, L. Ratinov, and Y. Bengio. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 384–394, 2010.

R. Weischedel and A. Brunstein. BBN pronoun coreference and entity type corpus. Linguistic Data Consortium, 2005.