# Generalizing from Freebase and Patterns using Distant Supervision for Slot Filling

Roth, Benjamin; Chrupala, Grzegorz; Wiegand, Michael; Singh, Mittul

*Published in:*
Text Analysis Conference

*Publication date:*
2012

*Document Version*
Peer reviewed version

# Generalizing from Freebase and Patterns using Cluster-Based Distant Supervision for KBP Slot-Filling

**Benjamin Roth Grzegorz Chrupała Michael Wiegand Mittul Singh 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 2012 we extended last year's system in several respects. The core of the system is a set of semi-supervised per-relation classifiers, trained by a scheme known as *distant supervision*. Training data are generated by using Freebase and applying patterns. Relation models rely on (1) word clusters generalizing from context surface forms and (2) additional argument-level features. For the retrieval of answer candidates, we use document retrieval in combination with an entity expansion model based on Wikipedia link texts. We do not use a separate sentence retrieval step and rely entirely on the classifier for filtering out bad candidates. Our system does not rely on any syntactic analysis or co-reference resolution. The best-ranked run of the full system achieves an F-score of 23.4% on the official test queries.

## 1 Introduction

In the slot filling task of TAC KBP 2012 the objective is to develop a system which given an entity (person or organization) fills in missing information about it in a knowledge base. The evaluation is done on 42 relations where one argument is the query entity and the other argument has to be extracted from a document collection.

In general, several challenges are connected to this task:

1. Retrieving all documents and sentences from the text collection where relevant information are stored.
2. Mapping the human readable task definition to a machine readable representation.
3. Modeling both the contexts that express a relation as well as possible relation arguments.
4. Generating training data for machine learning algorithms.
5. Dealing with redundancy and ambiguity.

Our system tackles these challenges focusing on shallow machine learning techniques rather than deep linguistic analysis. Generalization is achieved by abstracting from words using automatically induced word clusters [Brown et al., 1992]. The seed argument pairs for the distant supervision [Mintz et al., 2009] training data are acquired from Freebase and patterns. Query redundancy and ambiguity is dealt with by a translation model based on Wikipedia link anchor texts [Roth and Klakow, 2010].

The structure of the paper is as follows: We give an overview of our architecture in Section 2.1. Sections 2.2 to 2.7 discuss the single components of the relation extraction pipeline. In Section 3 we provide details about training the relation classifier. The results achieved in the TAC KBP 2012 slot filling benchmark are discussed in Section 4.

## 2 Relation Extraction Pipeline

### 2.1 Overview

Figure 1 gives an overview of the relation extraction pipeline. The system starts with the query as provided by TAC and expands the entity name to

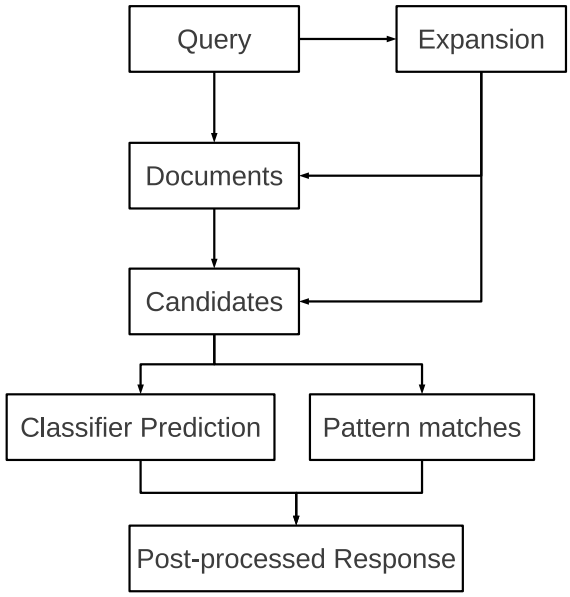


Figure 1: Pipeline of the relation extraction system.

possible other name variations of the query entity (see Section 2.2). The original query and possibly some variants are then used to retrieve documents that may contain information about the entity (Section 2.3). From the retrieved documents those sentences are filtered out that contain possible slot filler candidates for any of the sought relations (Section 2.4). Candidate sentences contain a reference (name variant) to the query, and a token sequence of the appropriate slot type. Features are extracted from the candidate sentences and the instances are judged by binary per-relation classifiers (Section 2.5). Another (high precision, low recall) module operates directly on the sentence surface strings and tries to match relation specific patterns (Section 2.6). The answers returned by the classifier and pattern matching are then merged and post-processed to match the task-specific guidelines (Section 2.7): Redundant answers are removed by a mechanism similar as in entity expansion, some cut-offs are applied to the number of answers (e.g. for single-slot) and dates are normalized.

### 2.2 Query Expansion

For the query entity name a list of aliases is computed from Wikipedia link anchor text. The expansion scheme is similar to translation models successfully applied in cross-language information retrieval [Roth and Klakow, 2010]. The set of aliases $A$ is computed as follows:

$$A = \{alias : P(alias|wp) \in topn \wedge wp = \arg\max_{wp} P(wp|q)\} \quad (1)$$

For a query $q$, the Wikipedia article page ($wp$) is selected that $q$ is most likely linked to. For this article $wp$, the top-$n$ link anchor texts are returned, $n$ is set to 10. For the strings $alias$ and $q$ the probabilities are estimated from their link counts $n(\cdot, wp)$ with the articles in Wikipedia (redirects are followed), e.g.:

$$P(alias|wp) = \frac{n(alias, wp)}{\sum_{alias} n(alias, wp)} \quad (2)$$

For the probability estimates there needs to be a minimum frequency of 2. For persons also the last name (last token) is optionally added to the aliases. The query expansions are also used as candidates for the *org:alternate_names* and *per:alternate_names* relations.

### 2.3 Document Retrieval

Document retrieval is a vital step of the pipeline. An Apache Lucene[1] index is used for it. The aim is to obtain all, or at least sufficiently many, documents containing information about the query entity. The query entity may be expressed in one of their alias forms in the documents. However, just using all aliases leads to ambiguity and precision problems as too unspecific alias forms may be contained in the expansion. Therefore, the Lucene query is built up in the following way:

1. Add the original name to the query.
2. For each alias, compute the point-wise mutual information (PMI) with the original name on the document collection. Add (with OR) the alias with the highest PMI, if the PMI is positive.
3. If there are no documents returned by the query obtained so far, use the following back-off

[1] http://lucene.apache.org/

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

Table 1: NE labels.

| Precision | Recall | F-measure |
|---|---|---|
| 91.18 | 92.15 | 91.66 |

Table 2: NER results on BBN section 22.

mechanism: Retrieve the highest ranked document with a new query containing all aliases.

The document threshold is set to 500 documents.

### 2.4 Candidate Generation

From the retrieved documents, those sentences are retained that contain a mention of the query name or an alias, and a token sequence tagged with the expected slot type. We use a perceptron-trained sequence labeler [Collins, 2002] on the BBN training data [Weischedel and Brunstein, 2005] after mapping the BBN label set to the coarse-grained set shown in Table 1. We use the same word cluster features as described in [Chrupała and Klakow, 2010]. The overall performance of the NE labeler on section 22 of the BBN corpus is shown in Table 2.

Additionally we provide lists for types that cannot be mapped to the BBN labels or where there is insufficient training data, and mark all token sequences that match a list entry. We obtain these lists by enumerating all entries of the corresponding types in Freebase[2]. URLs are matched by a regular expression.

### 2.5 Relation Classification

From the candidate sentences, features are extracted to build instances for the classifier. Context features are heavily based on automatically induced word clusters [Brown et al., 1992]. We use a hierarchical clustering of word types with 3200 clusters at the leaves of the hierarchy, trained on the same Reuters news data.[3] Following previous practice [Ratinov and Roth, 2009, Turian et al., 2010], we used cluster id prefixes (i.e. levels of depth in the binary cluster tree) of lengths 2, 6, 10 and 20. Additionally we extract features about the slot value and its connection to the query entity match. The extracted features are (we refer to slot candidate and query entity match as *arguments*):

- Counts of uni-, bi-, tri-, and four-grams:
  - Word n-grams in query entity and slot value, and word n-grams between the arguments, as well as three before and three after the arguments.
  - Brown cluster n-grams with cluster id prefix of length 2, 6, 10 and 20.
  - Word stems of whole sentence.
- Log distance between query entity and slot value.
- Argument modeling:
  - Positive point-wise mutual information of arguments in corpus (doc-level).[4]
  - Argument Brown clusters (2, 6, 10, 20) n-grams (1, 2, 3).
  - Jaccard coefficient of tokens in slot / query match.
  - Jaccard coefficient of character bi-grams in slot / query match.
  - Is slot acronym of query / vice versa?
  - Prefix / suffix letter overlap ratio.
  - Slot / query match, first and last character n-grams (1, 2, 3, 4).
  - Is slot / query match all CAPS?
  - Number of slot tokens.
  - Log number of slot characters.

[2] `http://www.freebase.com/`

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

[4] This feature requires access to the index, which is slow compared to the other features that are only memory-based look-ups. Therefore this feature is not used in all runs.

Classification is performed by a support vector machine trained on distant supervision data (see Section 3); the SvmLight toolkit[5] is used. The sentence instances are scored by the classifier one by one. If several sentence instances contain the same slot fillers, the instance with the highest classifier score is considered in the further steps. This instance also determines the document id as required by the task guidelines.

### 2.6 Pattern Matching

The task guidelines and slot definitions contain roughly half a page of description per relation. These descriptions consists of definitions and examples that are supposed to give a human readable guidance for judging whether a relation is considered to hold in a particular context. Whatever learning algorithm is used, there always has to be a mapping or transformation of the guidelines performed by a human to some machine readable resource or algorithm.[6] The manual human effort can be e.g. a mapping to Wikipedia info-boxes or to Freebase relations, the creation of gazetteers, the annotation of training data, specific algorithmic routines, or formulation of question templates or patterns.

In this step, we employ one of the simplest of the above methods, namely to use patterns directly following from the definitions and examples given in the guidelines. For example, if the guidelines contain an example sentence for *per:stateorprovince_of_birth*

*Harper, born in April of 1959 in Toronto, Ontario*

then a pattern to consider would be

*ARG1 , born * in *, ARG2*

where *ARG1* stands for the query entity match and *ARG2* for the slot filler, the star (*) is used to indicate 1 to 4 tokens. If such a pattern matches, the slot candidate is scored positive by the system. Together with the type filter on slot candidates from the previous step these pattern are high precision, but it is obvious that they are of very limited coverage. The main use of these patterns therefore is to extract distant supervision training data (see Section 3), which can be seen as a form of pattern expansion.

### 2.7 Post-processing

The responses that are judged positive by the classifier (Section 2.5) and pattern matcher (Section 2.6) are ranked by their score. The pattern matcher assigns a score of 1.0 to its matches, while the classifier assigns the regression values.[7] For single-slot relations only the highest ranked slot filler is kept, ties are broken according to precedence in the retrieval step. For list-valued relations, all positive responses are mapped to a normal form, based on Wikipedia link anchor text.[8] For every slot filler the top-1 expansion is calculated (as described in Section 2.2), which is in turn lower-cased and stripped off all non-letters and non-decimals. If two slot fillers are mapped to the same normal form, only the higher ranked slot filler is kept. Dates are normalized by a special routine.

An optional step of post-processing is a relation-specific cut-off value for the number of highest ranked answers returned per slot. While setting such thresholds on the development data (TAC KBP 2011 queries) greatly improved performance, this year's runs indicate that it did not have the expected positive effect (see Section 4).

## 3 Training

Training is done in a distant supervision setting [Mintz et al., 2009]. Pairs of arguments that are known to stand in a particular relation are matched against a text corpus. Those sentences in which both arguments appear together are taken as positive examples, while the positive examples from the other relations are taken as negatives. In our system we use two ways of obtaining pairs associated with a particular relation:

1. Entities that are connected with Freebase relations that correspond to TAC KBP relations.

[5] `http://svmlight.joachims.org/`, [Joachims, 1999]

[6] A system that would read task guidelines and program itself accordingly probably would be AI complete.

[7] The regression scores for one slot are normalized to lie between 0 and 1.

[8] An exception is made for *org:alternate_names* and *per:alternate_names*, as here one is not interested in unique slot-filler entities but surface forms.

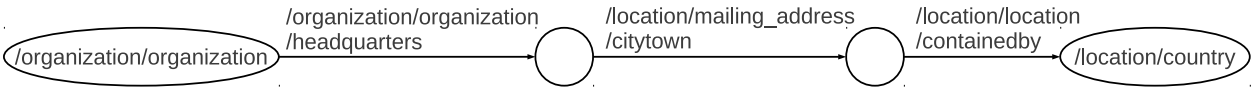


Figure 2: Path of Freebase relation that corresponds to the TAC KBP relation org:country_of_headquarters. Nodes correspond to Freebase entities that can be matched, they can contain restrictions on their type. Edges correspond to relation restrictions. The start node corresponds to the query argument in the TAC KBP relation, the end node corresponds to the slot argument.

2. Entities that co-occur at least once with a pattern match (see Section 2.6).

In the first case most TAC KBP relations correspond to joins on the Freebase database.[9] We formulate database queries on Freebase that can contain both restrictions on (binary) Freebase relations as well as on entity types. The database queries can be seen as graph configurations. See Figure 2 for an example of a graph configuration in Freebase that is mapped to a TAC KBP relation.

Also note that while in the second case (pattern matches) the pattern has to match at least one context of an argument pair, also the other occurrences of a pair are considered as training input for the distant supervision classifier. With these two strategies large amounts of training data can be obtained. In both cases we limit the number of pairs per relation to $10,000$, respectively.

We extract the same features as described in section 2.5. In order to limit the number of training instances, training sentences are aggregated per argument pair and feature weights are averaged.[10] For each relation we train 3 svm classifiers, differing in their cost-parameter (we set it to be $0.05$, $1.0$ and $20$). We found that different relations require widely different cost-parameters and so the final per-relation support vector machines are chosen according to their performance on the development data (TAC KBP 2011 queries).

## 4 Results

For the slot filling task of TAC KBP 2012 we considered the following configurations:

1. *Cooc* - Uses co-occurrence features on the document collection and fewer training data. Uses tuned cut-off for number of returned answers.
2. *NoCooc* - No co-occurrence features on the document collection, more training data than in *Cooc*. Uses cut-off for number of returned answers relative to last year's average/max.
3. *NoCutoff* - High recall run. Merge of *Cooc* and *NoCooc*. No cut-off for number of returned answers. Lenient matching of person names (last names suffice).
4. *LastYear* - Merge of *Cooc* and last years' system [Xu et al., 2011].
5. *LinReg* - filler ranking based on linear regression instead of SVM, no slot-type-specific tuning of thresholds, no patterns.

| Model | Run | Recall | Prec. | F-score |
|---|---|---|---|---|
| Cooc | lsv1 | 0.159 | **0.317** | 0.212 |
| NoCooc | lsv2 | 0.167 | 0.294 | 0.213 |
| NoCutoff | lsv3 | **0.250** | 0.220 | **0.234** |
| LastYear | lsv4 | 0.194 | 0.212 | 0.202 |
| LinReg | lsv5 | 0.123 | 0.199 | 0.152 |

Table 3: Scores of different system configurations on the 2012 test data.

Table 3 shows the scores obtained by the five submitted runs on the official 2012 test data. For the best performing configurations, recall is the most crucial factor. Attempts to improve results by increasing precision and setting thresholds on the number of returned answers did not lead to a higher overall F-score, as the comparison of *Cooc*, *NoCooc* and *NoCutoff* shows. Overall, the changes to last year's system, mainly entity expansion and normalization, pattern matching, new distant supervision training data and an extended feature set greatly improved performance. These improvements allowed us to relax and remove previously used filters like

[9] *per:age* is not encoded in Freebase, as it is relative to document creation time.

[10] The run using the index-based co-occurrences feature only uses 3200 pairs per relation

sentence retrieval and answer thresholds that had been necessary to increase precision. Adding the answers of last year's system even hurt the performance, as one can see from a comparison of *Cooc* and *LastYear*. We also experimented with an alternative linear regression classifier, however as one can see from the *LinReg* results, support vector machines seem to be a reasonable choice.

## 5 Conclusion

We introduced a distant supervision system for relation extraction where the distant supervision matches come both from mapped Freebase entries and patterns. The features used for classification are mainly based on automatically induced word clusters for context modeling while another feature subset is dedicated to argument modeling. Candidate retrieval and entity matching are other important parts of our system. We mainly use a translation model based on Wikipedia anchor text for entity expansion.

Our system focuses on sentence-level relation extraction. It prefers shallow context modeling and language independent resources over deep linguistic analysis. This approach showed strong performance and we believe it is advantageous also for portability to settings where linguistic tools are not available.

## 6 Acknowledgements

Benjamin Roth is a recipient of the Google Europe Fellowship in Natural Language Processing, and this research is supported in part by this Google Fellowship. Grzegorz Chrupała and Michael Wiegand were funded by the German Federal Ministry of Education and Research (BMBF) under grant number 01IC10S01O as part of the Software-Cluster project EMERGENT (www.software-cluster.org). Mittul Singh is supported by a Google Research Award.

## 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.

T. Joachims. Making large scale svm learning practical. 1999.

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.

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.

B. Roth and D. Klakow. Cross-language retrieval using link-based language models. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 773–774. ACM, 2010.

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.

F. Xu, S. Kazalski, G. Chrupala, B. Roth, X. Zhao, M. Wiegand, and D. Klakow. Saarland University Spoken Language Systems Group at TAC KBP 2011. In *Proceedings of TAC 2011*, 2011.