MSIIPL_THU's Slot-Filling Method for TAC-KBP 2015

Yangcheng Zhang[12], **Hengsheng Liu**[13], **Gang Zhao**[12], **Ji Wu**[14]
[1] Department of Electronic Engineering,
Tsinghua University, Beijing, China
[2]zhang-yc13,zhaog12@mails.tsinghua.edu.cn
[3]liuhs_ts@tsinghua.edu.cn
[4]wuji_ee@mail.tsinghua.edu.cn

Abstract

This paper presents the design and implementation of our first English slot filling system. The slot filling task aims at extracting attribute values of the given entities. The core of the system is a set of supervised per-relation classifiers, trained by a scheme known as distant supervision. We use Freebase and Wikipedia to generate our training query-filler pairs. Annoted Gigaword received from the organizer is used to train our models. For the retrieval of answer candidates, we use sentences retrieval in combination with query expansion in some certain ways. Relation models rely on a feature representation focusing on surface skip n-grams and the shortest dependency path which connects an query-value pair. A Rule-based method is also added to our system as a supplement to improve the performance. Evaluation results show the strength and weakness of our technique.

1 Introduction

This is the first year that MISSPL_THU participated in the Slot Filling task of TAC's Knowledge Base Population. English Slot Filling is a task to extract attribute values of a given entity(person or organization) from large collection of documents. The evaluation is done on 41 relations where there are two arguments. One argument is the query entity while the other has to be extracted from the specified document collection.

Generally, participants will face several main challenges in such a task [Benjamin et al., 2012]:

1. How to retrieve all documents and sentences from the specified text collection.

2. How to model both the contexts that express a relation and corresponding relation arguments.

3. How to generating training data to train a model.

In our work, we prefer using shallow machine learning algorithms rather than deep linguistic analysis. Since a query may have several coreferential name variants, we expand a given query with some suitable variants generated by rules or found in the context. Then we retrieve all these expanded query names using Lucene[1] in the training or test document set. The training data for the distant supervision [Mintz et al., 2009] are acquired from Freebase[2] and Wikipedia [3]. To improve the performance, we also use the rule-based method as a supplement.

The structure of the paper is as follows: We describe our system pipeline in Section 2.1. And from Sections 2.2 to 2.7, each component of our architecture will be discuss minutely. Section 3 presents the training details of the relation classifier. In Section 4, we analysis the results achieved in the TAC KBP 2015 slot filling track.

2 Slot Filling Pipeline

2.1 Overview

The pipeline of our slot filling system is shown in Figure 1. We first preprocess the given document collection (Section 2.2). Then each query entity will be sent into our system and expanded to other possible name variations (Section 2.3). The new queries include the variants and the original query. The new queries are used to retrieve sentences (or

[1]http://lucene.apache.org/
[2]http://www.freebase.com/
[3]https://en.wikipedia.org/

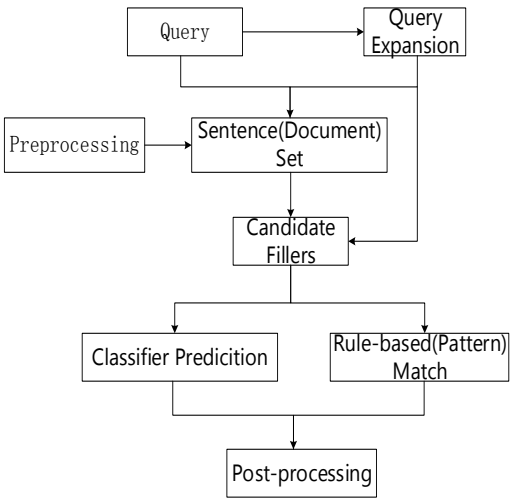

Figure 1: Pipeline of our slot filling system

documents) that contain information about the original entity(Section 2.4). Now we have some candidate sentences which contain both a query in the new queries and a token sequence of the appropriate slot type. To decide whether a candidate slot filler is correct, we should extract some features from the candidate sentences in a certain way, then use the trained binary classifier to judged the instances (Section 2.5). We also integrate a rule-based or pattern-based module with the classifier algorithm. Such a module is expected to match relation-specific patterns by working on the sentence surface strings (Section 2.6). The last stage of our pipeline is to post-processed the answers returned by the classifier and pattern-based matching. Answers for dates will be normalized, and redundant fillers are removed (Section 2.7).

2.2 Preprocessing

We first use the Stanford CoreNlp[4] package to annotate the linguistic features of all the given test corpus. The package includes the part-of-speech (POS) tagger, the named entity recognizer (NER), SUTime, the dependency parser and the coreference resolution. For some reasons, we make all the above annotation except the last one. Then we index the sentences with their annotations using Lucene. Another important work is to collect enrich query-filler training pairs. We use relations and relation instances from Freebase and Wikipedia infobox. Because the properties in Freebase, Wikipedia and TAC-KBP slot filling track are almost different, we manually map the properties in Freebase and Wikipedia to those in TAC-KBP. At this stage, we also categorize all the slot values into some domain types. The NER tool is used to recognized PERSON, ORGANIZATION, LOCATION, DATE, NUMBER etc., while URL, TITLE, CRIMINAL CHARGE, CAUSE OF DEATH and so on are mainly recognized using rules and pre-prepared lists.

[4]http://nlp.stanford.edu/

2.3 Query Expansion

A query expansion module is necessary to improve the Recall of a slot filling system. Such a module should identify as many as possible name variants of the given query. For ORG type, we expand the query by ignoring the corporate suffixes (e.g. ”Ltd”, ”Corp.”). We also searched the acronyms and full names of the query in the context. Using the coreference resolution tool in Stanford CoreNLP, we get the coreference names of the given query. To identify the alternate names of a query, we add a bit rules to our system. For example, if the query is followed by some tokens like ”known as”, ”called” and so on, then it is very likely to find the variant or filler behind such tokens. Benjamin et al. [2012] used a translation models to computed from Wikipedia link anchor text. We also try such a model for query expansion but get no performance improvement regrettably. Too many variants for a query may lead to ambiguity and precision problems. We remove some unspecific alias forms in the expansion by comparing the similarity of the variants with the original query.

The first run we submitted used the module, and the second run not. We will discuss the result in section 4.

2.4 Sentence Retrieval and Candidate Generation

We use the Apache Lucene tool to index the sentences in the training and test corpus. Indexing the whole document is an considerable alternative. The original query and its variants are used to index. The aim is to obtain as many sentences containing the

query or an alias and a token sequence tagged with the expected slot type as possible. We restrict the number of sentences retrieved by 500 per query. For the relations with unusual slot types, such as TITLE and NATIONALITY and so on, we provide lists to find a match. We also use Freebase and Wikipedia to collect many entries of the corresponding types. Slot types like CAUSE_OF_DEATH and URLs etc. are matched by rules.

2.5 Features Extract and Relation Classification

Features are extracted from the above candidate sentences and then used to build instances for the classifier. We use features including token n-gram-based and shortest dependency path between the query and the candidate filler.

The token n-gram-based features are similar to those of Benjamin et al. [2012]. A common token feature is the n-grams between the query (referred to as ARG1) and the filler (referred to as ARG2). We use such a feature up to length 3. Another token feature is called skip n-grams, where we remove the stop-words and wildcard some tokens in the middle of the n-grams between ARG1 and ARG2. Such a feature is more robust and can increase performance. The left and right context outside the arguments are also available token features. They have a length up to 3 (including the wildcarded argument). In our experiments, we additionally use a feature showing whether ARG1 or ARG2 comes first. An example in Figure 2 shows how to extract features for an instance.

The last feature is the shortest dependency path, which is achieved using the Stanford CoreNLP package. Also we wildcard some terms in the path to enhance the robustness of the feature. A query-filler pair which has a long dependency path more than 7 is discarded. Indeed we find too long path generally indicates an error pair.

To train a binary classifier per-relation, we use the handy tool called LIBLINEAR[5] with logistic regression (LR). We train the LIBLINEAR model on distant supervision data (see Section 3). The trained classifier scores each sentence instance. If the same fillers appear in several sentence instances, we reserve and consider the instance with the highest score in the next steps. The slot filling task of this year add a requirement to use the retrieved answers and inverse the slots to generate new answers. We just use the relevant old classifiers and some new patterns manually built to complete it.

[5]https://www.csie.ntu.edu.tw/ cjlin/liblinear/

2.6 Pattern-based Matching

There are patterns which indicate a specified relation between the given query and fillers. In the task guidelines, each relation has a description of definitions and examples. Such examples direct us to judge if a particular context holds a certain relation. At this stage, we manually construct a bit patterns according to the given examples in the task guidelines. The following is an example sentence for *per:stateorprovince_of_birth* in the task guidelines of 2015

Harper, born in April of 1959 in Toronto, Ontario

So we can build a pattern like this

*ARG1, born * in *, ARG2*

We use *ARG1* and *ARG2* to represents the query and the slot filler respectively. Redundant tokens are wildcarded by a character star (*). Such a pattern can match a high precision result, which almost always contains the correct slot filler we need. We construct patterns more or less for each relation, and mainly use them to validate the sentences discarded by the trained classifier. For some relations such as CAUSE_OF_DEATH and CHARGE, matching the sentences by patterns has a better performance than the classifier in our experiments. Table 1 shows the effect of patterns matching. We will discuss the result in Section 4.

use patterns matching	P	R	F
no	4.0%	9.1%	5.6%
yes	27.5%	7.0%	11.1%

Table 1: show the effect of pattern matching

The shortage of this method is also apparent and severe. It consumes human and is of limited coverage. Yan Li et al. [2013] designed an enhanced ad-

Relation: per:origin(Alan Bishop, Britain)

Candidate sentence: *The man said he is Alan Bishop, a London native and the first Britain citizen to be awarded medals.*

Feature examples:
BETWEEN_N_GRAM: ARG1>#,>
OUTSIDE_N_GRAM: ARG2>#citizen>#to>
SKIP_N_GRAM: native>###first>

Figure 2: An example of extracted token features. Only strings behind the colon are the features. Character # is used as a separator. To indicate whether the filler comes left or right of the query, we use characters $>$ (left) or $<$ (right) to mark each token.

aboost pattern-matching system, which inspired by the work of Grishman et al. [2010] and Snowball (Agichtein et al. [2000]). It can automatically generate a lot patterns, and improve the system performance to some extent.

2.7 Post-processing

The major work in this section is to fix the answers achieved above to meet the requirements. For the slots of DATE type, we used SUTime tool in Stanford CoreNLP to normalize the temporal expression to the required TIMEX2 format.

The LOCATION slot has fillers which should be countries or states/provinces or cities. We preprepare a list of all countries and states/provinces from Wikipedia. Fillers not in the list are regarded as cities.

For single-valued relations, the highest ranked slot filler is reserved, while the first 4 answers are kept for the list-valued slots.

3 Training Models

The process of training is similar with that in the work of Mintz et al. [2009]. We train a binary classifier using LIBLINEAR with logistic regression for most of the relations. LIBLINEAR is a linear classifer for data with millions of instances and features. It can quickly train a much larger set than LIBSVM[6] via a linear classifier.

The annotated Gigaword corpus is used as training set. We use Freebase and Wikipedia to get a lot of query-filler training pairs for most of the slots. The relations map from Freebse and Wikipedia to TAC-KBP slot filling task is essential. Then the positive training pairs are used to retrieve up to 500 sentences for each pair from the Gigaword corpus. Sentences containing both the query and filler in a pair are taken as positive examples. A relevant negative pair is constructed using the same query as the positive pair, but the second argument is different from the correct filler in the positive pair. The other requirement is that it should have the same slot type with the correct filler. An example is *(Barack Obama, U.S.)* for *per:origin*, then *(Barack Obama, England)* is a relevant negative pair, and the sentence contains *(Barack Obama, England)* is a negative example. We change the number of negative examples to decide a suitable proportion between negative and positive examples. Table 2 shows the result of different proportion. We will discuss the result in Section 4.

Features are extracted from each sentence. The features of those sentences belong to a specified pair are combined into a rich feature vector. Such a vector is a numerical instance of the relevant relation. Then we use these vectors to train a binary classifier for most of the relations. For unbalanced data, we set some weights before training. We tune the hyper-

[6]https://www.csie.ntu.edu.tw/ cjlin/libsvm

N/P	P	R	F
0.001	18.8%	10.5%	13.5%
0.01	23.0%	10.0%	13.9%
0.1	27.5%	7.0%	11.1%

Table 2: Result of different negative/positive examples proportion. N/P is the proportion of negative examples and positive examples; P, R and F are the Precision, Recall and F-value of our system respectively

parameters in the model using cross validation.

The distant supervision will introduce much noise. So noise reduction module is necessary. We try to combine a multi-instances multi-labels (MIML)[Mihai et al. 2012] module with our system, but can't complete eventually with limited time.

4 Results And Analysis

We submit four runs for the English Slot Filling task this year. The last two runs have some problems in the models, so have scores closed to zero. Table 3 shows the first two evaluation results.

	All_P	All_R	All_F
run1	0.1901	0.0595	0.0907
run2	0.2143	0.0524	0.0842

Table 3: the evaluation results

The first run we submitted use query expansion, and the second run not. Both runs use the patterns matching. With limited time, we do not test another modules in the runs. But we can get some inspiration from the earlier experiments.

While our query expansion module has unconspicuous effect on enhancing the performance of our system, the pattern matching module brings in a huge improvement. Both modules are based on the classifier algorithm. From table 1, we find our classifier algorithm is too rough.

The evaluation results show the Recall of our system is too low. We think there are two main reasons for this fact. First, our LIBLINEAR-based supervision approach is not well. Apart from the classifier itself, we use a lot training data from Freebase, then retrieve the candidate sentences in Gigaword corpus, and we train the classifiers following the assumption of distant supervision method. This process introduce a lot noise with which we have no special module to deal. Second, Our retrieval module including query expansion does not work well. In our early experiment on the test corpus of last year, we find the correct fillers in our candidate sentences are less than half. After using NER to match the pre-defined slot types and extracting features to form numerical instances, the remain examples contain no more than one-fourth correct fillers. Then the residual examples should still be classified by the trained classifiers. Eventually, most of the correct fillers are filtered out, resulting in a poor Recall. Also we do no co-reference resolution, and only build a bit patterns to matching the relations. Another weakness is that we get poor performance in the hop1.

Next we will first introduce a noise reduction module to improve the performance of classifiers. And We may choose another exquisite model to try and tune the parameters detailedly. Also we will spend time improving the patterns matching module to get more high-precision candidates. To solve the problem that the query and the correct fillers belong to different sentences, we should consider how to retrieve implicit candidates from document set, which includes the techniques of inference and co-reference resolution.

5 Conclusion

The paper gives an overview of our submission to the TAC-KBP 2015 slot filling task. We develop our slot filling system combining pattern-based method and classifier algorithm. The evaluation result shows the strength and weakness of our approach. In the future, we will try to improve the performance of existed modules and introduce some new modules to obtain a whole advance.

References

Benjamin Roth, Grzegorz Chrupala, Michael Wiegand, Mittul Singh, Dietrich Klakow. 2012. *Generalizing from Freebse and Patterns using Cluster-Based Distant Supervision for KBP Slot-Filling*. In Proceedings of the Text Analysis Conference(TAC), 2012.

Mike Mintz, Steven Bills, Rion Snow, Dan Jurafsky.

2009. *Distant supervision for relation extraction without labeled data*.

Yan Li, Yichang Zhang, Doyu Li, Xin Tong, JianLong Wang. 2013. *PRIS at Knowledge Base Population 2013*,.

Bonan Min, Xiang Li, Ralph Grishman. 2013. *New York University KBP 2010 Slot Filling System.*, In proceedings of TAC-2010.

Eugene Agichtein, Luis Gravano. 1999. *Snowball: Extracting relations from large plain-text collections*, *Columbia University Computer Science Department Technical Report CUCS-033-99*, December 1999.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, Christopher D.Manning. 2012. *Multi-instance Multi-label Learning for Relation Extraction*, *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 455-465, Jeju Island, Korea, 12-14 July 2012.