Ghent University-IBCN Participation in TAC-KBP 2015 Cold Start Slot Filling task

Lucas Sterckx, Thomas Demeester, Johannes Deleu, Chris Develder
Ghent University - iMinds
Gaston Crommenlaan 8
Ghent, Belgium
{firstname.lastname@intec.ugent.be}

Abstract

This paper presents the system of the UGENT_IBCN team for the TAC KBP 2015 cold start (slot filling variant) task. This was the team's second participation. The slot filling system uses distant supervision to generate training data combined with feature labeling and semi-supervision, and two different types of classifiers. We show that the noise reduction step significantly improves precision, and propose an application of word embeddings for slot filling.

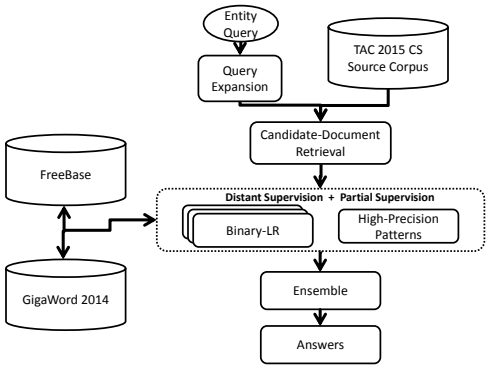

Figure 1: System Overview

1 Introduction

Relation extraction is a vital step for information extraction. This task has received attention in TAC KBP for a number of years as the Slot Filling track, and as a vital sub-task of the recently-introduced Cold-Start track. As this is our first participation in the Knowledge Base Population - Slot Filling and Cold Start tracks, our system starts from previous work by other teams, in particular systems described in (**?**) and (**?**) which use facts from external databases to generate training data, also known as distant supervision.

Distant supervision has become an effective way for generating training data in the slot filling task, as proven in last year's top submission (**?**). The main contribution of our system is twofold: (1) we add a noise reduction step that boosts precision of the distant supervision step , and (2) we explore the use of word embeddings (**?**) for a relation detection classifier.

In the following Section, we give an overview of the system and describe different components. A more elaborate discussion of the training with distant supervision is given in Section 3. Section 4 discusses some adaptations of the system for the cold start task. Finally, results and a brief conclusion are given in Sections 5 and 6.

2 System Overview

Figure 1 shows an overview of the slot filling system. Interactions between different components of the system and the different sources of data are visualized by arrows. In this Section we discuss those parts of the system which act at run time for the generation of slot fillers.

2.1 Query Expansion and Document Retrieval

The first step is the retrieval of all documents containing the entity query (person or organization) from the TAC 2014 source document collection. We

expand the query by including all alternate names obtained from Freebase and Wiki-redirects for the given query. When we do not retrieve any alternate names, we clean the query, e.g., remove middle initials for persons and remove any company suffixes (LLC, Inc., Corp.) and repeat the search for alternate names using this filtered query. For indexing and search of the source collection we use Whoosh[1].

2.2 Relation Classifiers

In each retrieved document we identify relevant sentences by checking if any of the entities from the expanded set of query entity names are present. Note that we did not use any co-reference resolution to increase recall, as suggested in earlier work (**?**; **?**). Next, we assign all slot candidates from the relevant sentences with a type (e.g., title, state-or-province). Slot candidates are extracted using the Stanford 7-class Named Entity Recognizer (**?**) and assigned a type using lists of known candidates for each type. For each combination of the extracted candidates with the query entity, we perform a classification of a type-matching relation from the TAC-schema. We trained four different sets of classifiers, i.e., two sets of binary Support Vector Machines (SVMs) and two multiclass classifiers, that resulted in four different runs submitted for the slot filling task. Only the first classifier was used for a run in the cold start task.

2.2.1 Binary SVMs

41 different binary SVMs are used to detect the presence or absence of each relation in the sentence for the query entity and a possible slot filler.

All binary SVMs trained for different relations use the same set of features, which is a combination of dependency tree features, token sequence features, entity features, semantic features and an order feature. We extract these features using Stanford CoreNLP tools (**?**). This corresponds to the features used in (**?**). A complete overview of the used features is given in Table 1 using an illustration of the features for example relation-tuple <Ray Young, General Motors> and the sentence "Ray Young, the chief financial officer of General Motors, said GM could not bail out Delphi"[2].

[1] `http://pythonhosted.org/Whoosh/`

[2] The same example sentence as used in (**?**)

name	description
wbo	words in between the two entities
bm	two words in front of the first entity
am	two words after the last entity
et1,et2	the entity types of the two entities
ntw	# words in WBO

Table 2: The features used to train the multiclass classifiers.

Figure 2: Schematic representation of the CNN classifier.

The two sets of binary SVM classifiers differ in their training data. Classifiers for the second run use the original output from the distant supervision step, while the first set is trained on training examples after a noise reduction step. Section 3 describes this training data in more detail.

2.2.2 Multiclass Convolutional Neural Network

We experiment with word embeddings and see if relation classification can benefit from their use in the classification task. Therefore, we implement a single Convolutional Neural Network (CNN) that functions as a multi-class classifier and we compare the performance of this network with the classification obtained by a logistic regression classifier with the same set of features, but without the use of word embeddings (discussed in the next Section).
The CNN only uses a subset of the features used by the SVMs, as shown in Table 2. The network is trained on the cleaned training data, which we introduce later on.

We use the SENNA-embeddings (**?**) as word embeddings for the lookup tables. The length-varying WBO-features are modeled by a convolutional layer combined with a max layer. A schematic representation of our network is shown in Figure 2.

2.2.3 Multiclass Logistic Regression

The final classifier is a multi-class logistic regression classifier. It uses the same features as the CNN classifier (see Table 2), and the same training data. The classifier is constructed to test the impact of the word embeddings on the classification results. This classifier consists of a single logistic regression layer and the features are a concatenation of the features given in Table 2. The WBO-features are represented

Feature		Description	Feature Value
Dependency Tree	name	Shortest path connecting the two names in the dependency parsing tree coupled with entity types of the two names	PERSON appos offficer prep_of ORGANIZATION
	e1dh	The head word for name one	said
	e2dh	The head word for name two	officer
	same_e12dh	Whether e1dh is the same as e2dh	false
	e1dw	The dependent word for name one	officer
	e2dw	The dependent word for name two	nil
Token Sequence Features	tpattern	The middle token sequence pattern	,the chief financial officer of
	ntw	Number of words between the two names	6
	wbf	First word in between	,
	wbl	Last word in between	of
	wbo	Other words in between	{the, chief, financial, officer}
	bm1f	First word before the first name	nil
	bm1l	Second word before the first name	nil
	am2f	First word after the second name	,
	am2l	Second word after the second name	said
Entity Features	e1	String of name one	Ray_Young
	e2	String of name two	General_Motors
	e12	Conjunction of e1 and e2	Ray_Young–General_Motors
	et1	Entity type of name one	PERSON
	et2	Entity type of name two	ORGANIZATION
	et12	Conjunction of et1 and et2	PERSON–ORGANIZATION
Semantic Feature	mTitle	Title in between	True
Order Feature	order	1 if name one comes before name two; 2 otherwise.	1

Table 1: The features used to train the SVMs for each relation type.

as a Bag of Words, thereby ignoring the order of the words. To reduce the number of dimensions, we do not use all word ids, but use only the 6.000 most representative words. The representative power was measured by the information gain of the words for the classification task, obtained with Pearsons χ^2-test.

2.3 Entity Linking

Finally, the slot fillers extracted from the different documents are combined in an Entity Linking step. We link the entities from different documents and combine the extracted relation-tuples to obtain our final set of extracted relations. The output of this step consists of a list of all possible relation-tuples if the relation can have multiple tuples, e.g., for person_cities_of_residence. If only one relation instance is allowed, e.g., for city_of_birth, we choose the relation-tuple with the highest evidence. The evidence score for each relation-tuple is obtained by summing the evidence of all relation instances of this relation-tuple, i.e., the sum of the evidence-score given by the classifier for each sentence that expresses the relation-tuple.

3 Distant Supervision with Noise Reduction

Training data for the classifiers is generated using distant supervision. The left side of Figure 3 shows the different steps for the generation of the training data. We start by mapping FreeBase relations to KBP slots and subsequently search the full GigaWord corpus for possible mentions of these rela-

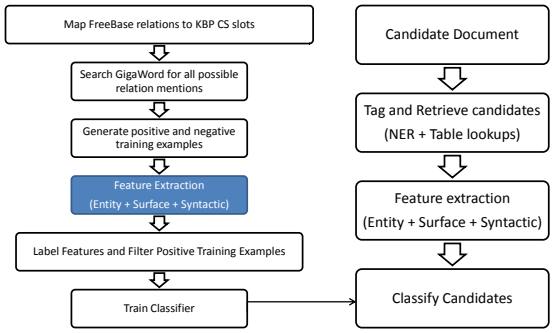

Figure 3: Classifier Overview

tions, i.e., two entities from a fact tuple co-occurring in sentences. Negative examples are all co-occuring entities which are not present in FreeBase.

3.1 Noise Reduction

The training data obtained by distant supervision is noisy, i.e., not all extracted sentences that may indicate a given relationship actually express this relationship. e.g., , "President Obama visited Honolulu" does not express the relationship "per:city_of_birth", although president Obama was born in Honolulu. To address this problem, we add a noise reduction for 14 (due to time constraints) frequently occurring relationtypes. We let a team of students label approximately 2.000 distantly supervised instances per relation type with a true or false label. The fraction of instances labeled 'True' for the different relation types is shown in Table 3.

The fraction of instances labeled 'True' strongly depends on the type of relation. To use these manually refined instances, we train a logistic regression (LR) classifier for each annotated relation and apply this classifier to the rest of the examples of the distant supervision output. For the relations with only a very small fraction of true examples (e.g., city_of_birth etc.) we did not train a LR classifier, but use a list of trigger words that need to be included, to filter the data. The results in Table 4 show that this noise reduction step indeed results in a significant increase in precision.

3.2 Subsampling

The dataset contains a lot more negative instances than positive instances. Therefore, we subsample the negative instances for each relation, to obtain approximately 50% positive instances and 50% negative instances for each relation.

relation	True fraction
per:title	91.78%
per:employee_or_member_of	90.90%
per:origin	86.48%
org:stateorprovince_of_ headquarters	86.64%
org:top_members_employees	73.61%
per:age	65.60%
per:charges	58.97%
per:spouse	56.10%
per:countries_of_residence	58.97%
per:city_of_death	15.26%
org:founded_by	12.89%
per:cities_of_residence	2.15%
per:city_of_birth	1.29%
per:country_of_birth	0.30%

Table 3: Fraction of instances labeled 'True' for 14 relation-types

4 Adaptations for the Cold Start Task

In our participation for the Cold Start Task (slot filling variant), our setup only required minimal modifications. We only used our SVM classifier with noise reduction for this task. The slot filling system and tagger were extended to handle relations involving Geopolitical Entities for the cold start variant. A first run is performed on the provided single-slot queries, and a second on slots for the resulting answers from the first run.

5 Results

5.1 Slot Filling task

The results for the slot filling task are shown in Table 4. Our best results are the median score for this year's competition. We ranked at the 10th place of the 18 participating teams and ended as the 3th new team of the 6 new teams. Furthermore, by only evaluating based on the retrieved fillers (ignoring document id's) our F1 increased by 3.15%, resulting in a 9th place.

Binary SVMs clearly perform better than the multiclass CNN and the multiclass LR. This large differ-

	precision	recall	F1
b* VSM + NR	24.1	15.9	19.2
b* VSM	16.4	16.3	16.3
m* CNN + NR	5.3	10.7	7.1
m* LR + NR	4.6	8.9	6.0
Median Score	25.8	16.1	19.8

Table 4: Results of the different runs on the slot filling task. b* stands for binary and m* for multiclass. NR represents the noise-reduction step.

	precision	recall	F1
Hop 0	24.7	16.6	19.9
Hop 1	7.5	4.9	5.9
All Hops	16.7	11.1	13.3

Table 5: Results of the different hops and the aggregate in the slot filling variant of the Cold Start task.

ence between approaches is mostly due to the lack of extra features in the multiclass CNN. Unfortunately, this makes it difficult to assess the impact of using a multiclass classifier vs. the multiple binary classifiers. Note that the multiclass classifiers use the same filtered training instances of the binary SVM with noise reduction. The noise reduction is clearly beneficial for precision. By using the filtered training data, the precision of the binary SVMs improves from 16.4% to 24.1%, while recall only drops from 16.3% to 15.9%. The comparison of the CNN with the LR classifier shows a slight increase in F1 obtained by incorporating the word embeddings. However, both values are far below the F1 obtained with the binary SVMs. So far, we were unable to improve any results by using word embeddings. In future work we will test whether we can improve these results by also including additional features for the CNN network.

5.2 Cold Start task

The results for the different hops of the slot filling variant of the Cold Start task are shown in Table 5. Precision and recall on the initial queries (Hop 0) are close to the performance on the regular slot filling task. In the second iteration (Hop 1) a lot of recall is lost, since a considerable fraction of these queries were not generated after the first run. Our system achieved second place out of three teams performing in the slot filling variant.

6 Conclusion

This paper described our first setup for the slot filling and cold start task which achieved results close to the median performance. We can conclude that the multiclass classifiers, only using lexical data, seriously underperformed the more elaborate binary SVMs and that we can significantly increase the performance of the classifiers by incorporating noise reduction of the training data obtained with distant supervision.