UBC at Slot Filling TAC-KBP 2010

Ander Intxaurrondo, Oier Lopez de Lacalle, Eneko Agirre
IXA NLP Group, University of the Basque Country, Donostia, Basque Country
aintxaurrond001@ikasle.ehu.es, oier.lopezdelacalle@ehu.es, e.agirre@ehu.es

Abstract

This paper describes our submissions for the slot filling and *surprise* slot filling tasks of TAC-KBP. The system is based on the distant supervision strategy presented by (Mintz et al., 2009). We did a straightforward implementation, trained using snippets of the document collection containing both entity and filler from the KB provided by the organizers (a subset of Wikipedia infoboxes). Our system does not use any other external knowledge source, with the exception of closed lists of words for religion, causes of death, charges and religious/political affiliation, plus the use of Geonames to distinguish between cities, countries, states and provinces. We submitted three runs based on different post-processing options of the output of our classifiers, with results below the median. We did expect low results, as our system is still under development, and we still have plenty of room for improvement.

1 Introduction

This paper describes our participation in the TAC-KBP 2010 slot-filling and surprise slot-filling tasks. Our system is a straightforward implementation of a distant supervision system (Mintz et al., 2009). The system was trained using snippets of the document collection containing both entity and filler from the KB provided by the organizers (a subset of Wikipedia infoboxes). Our system does not use any other external knowledge source, with the exception of closed lists of words for religion, causes of death, charges and religious/political affiliation, plus the use of Geonames to distinguish between cities, countries, states and provinces.

The paper is structured as follows. In Section 2 the tasks of *slot filling* and *surprise slot filling* will be described. In Section 3 the main components for the distant supervision system will be explained, including slot preparation, extraction of training examples, classifiers and the post-processing to produce the output. Next, we will focus on the results obtained by the three runs for slot filling, and for the unique run for surprise slot filling. Section 5 is devoted to error analysis, and finally, in Section 6, we draw some conclusions.

2 Slot Filling

The *slot filling task* in TAC-KBP consists on learning a set of predefined relationships and attributes for named entities (people or organizations) based on a pre-existing knowledge base extracted from Wikipedia Infoboxes. The learned information is then used to extract new facts from a large document base (1,7 million documents) for a set of target entities. The main objective is thus to feed Wikipedia Infoboxes with new additional values extracted from the document collection.

When we developed this system, in 2010, the TAC-KBP track was on its second edition.

The information in the KB is organized around *entity-slot-filler* triples. An entity is the name of the article of Wikipedia, and can include people or organizations. The slot is the type of information of the entity, for example the birthplace of a person. The filler is the value of the slot. An example of an *entity-slot-filler* triple could be *Paul Newman - date*

of birth - January 26, 1925. The target slots were defined by the organizers, including which are possible values, as made explicit in the task guidelines.

2.1 Surprise Slot Filling

The surprise task was optional in TAC-KBP 2010. In this task, participants had to find information for new entities and new slots, specified by TAC-KBP organizers. The main idea of the task consisted on giving portability to the information extraction systems developed by participants.

3 Distant supervision system

We tried a straightforward strategy for slot filling, designed around distant supervision (Mintz et al., 2009) and joint work by Stanford and UBC in TAC-KBP 2009 (Agirre et al., 2009). Our system is very similar to the later, with the difference that we used freely available tools and that our system is still under development.

Our systems has a training phase and an application (or test) phase. For training we perform the following steps:

- Slot preparation, including the extraction of entity-slot-filler triples from infoboxes, mapping them to *official* KB slots, and assigning a named-entity type or a closed list depending on the expected fillers.
- Example extraction, where we retrieve text fragments which include both the entity and filler in the triples
- Training of classifiers using the extracted examples

When applying the system we perform the following steps:

- Search of examples of mentions to the target entities
- Identification of potential fillers for possible slots
- Applying the classifiers to each filler in each mention
- Collation of results, where for each entity and slot the system returns the filler with maximum weight from classifiers[1]. When no filler is classified positively, the system returns NIL.

For the *Surprise Slot Filling* task, the organizers did not provide any training triples from Wikipedia infoboxes, and training examples were directly provided. Given the very small number of examples provided, we looked for additional examples containing those entity-filler pairs in the document base.

The development of the system did not involve manual curation of data, except assigning named entity classes (e.g., date, person) or closed lists of fillers (e.g., religions, countries, products, diseases) to each slot.

Below, we first present the details of how we prepared the slot information, then how we extracted the textual fragments (examples) of entity occurrences, followed by the method to train the classifiers. The application of the classifier to produce the slot filling results is explained next.

3.1 Slot Preparation

In order to prepare the training data for the slot classifiers, we first extracted entity-slot-filler triples from Wikipedia infoboxes using the mapping provided by the organizers.

As part of slot preparation, different slots based on the expected NE type were categorized (see Table 1: `ORG`, `PER`, `LOC`, `DATE`, and `NUMBER`. The NE type is used to help assign ambiguous infobox values to the appropriate slot, as well as to identify potential fillers for a text fragment for a slot. For `org:website`, regular expressions were used; nothing was done for `per:title`.

In the *Surprise Slot Filling* task, closed lists of fillers were used for all new slots.

Due to the ambiguity and noisiness of the infobox to slot mapping, we processed the infobox values for the entity-slot-filler triple as follows:

- We run a named-entity recognition and classification system (Finkel et al., 2005) on the entity itself to determine if the entity is `ORG`/`PER`. Because the slots are specific for the two entity types, we can safely ignore any entity that

[1] We tried slightly different post-processing strategies in the three submissions (cf. Section 4) following this idea.

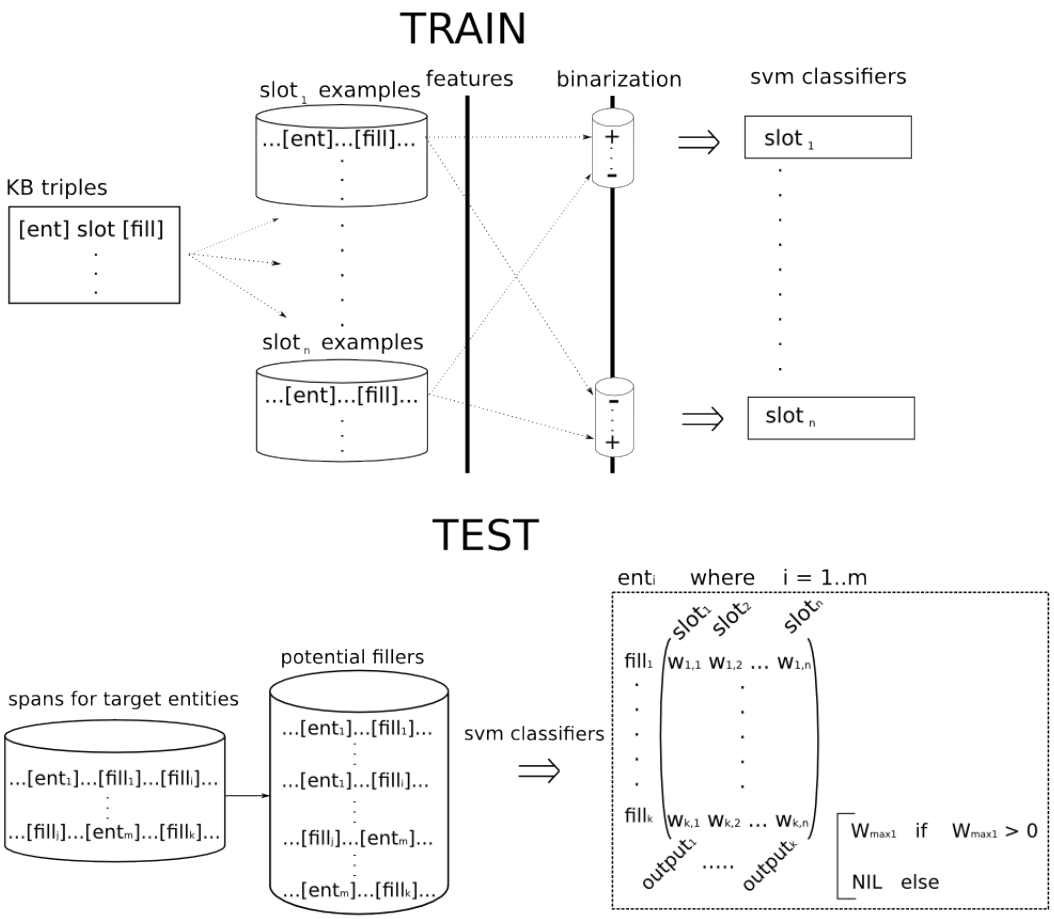

Figure 1: The architecture of the slot filling system. TRAIN: Extraction of KB triples, which are used to acquire training examples for each slot ($1..n$ slots) from the document base, followed by featurization and binarization. We finally train n classifiers, one per slot. TEST: examples containing mentions to the target entities (m entities) are retrieved from the document base (m target entities). Potential fillers are identified, and then each example containing one entity-filler is classified, obtaining a weighted prediction for each slot. Predictions are collated and the result returned.

NE (ORG)	org:alternate_names, org:founded_by, org:member_of, org:members, org:parents, org:shareholders, org:subsidiaries, per:employee_of, per:member_of, per:schools_attended
NE (PER)	org:founded_by, org:shareholders, org:top_members/employees, per:alternate_names, per:children, per:other_family, per:parents, per:siblings, per:spouse
NE (LOC)	org:headquarters, per:place_of_birth, per:place_of_death, per:residences
NE (DATE)	org:dissolved, org:founded, per:date_of_birth, per:date_of_death
NE (NUMBER)	org:number_of_employees/members, per:age
Closed List	org:political/religious_affliation, per:cause_of_death, per:charges, per:origin, per:religion
RegExp	org:website
NIL	per:title

Table 1: Mapping of slot to NE type or closed list. *LOC* slots belong to the 2009 TAC-KBP Slot Filling task, these slots will be later adaptted to fit with the 2010 task (see section 3.4)

is not ORG/PER. Besides, note that some entities in the knowledge base have been tagged as UNK by the organizers (instead of ORG/PER). We also run NER on this to determine the entity type.

- Run NER on infobox fillers to extract fillers for ambiguous slots. The mapping from the Wikipedia infobox to the TAC-KBP slots can be ambiguous. For instance, the Wikipedia infobox "born" can map to date_of_birth, country_of_birth, stateorprovince_of_birth and city_of_birth:

  Carrie Underwood
  Born March 10, 1983 (1983-03-10) (age 6)
  Muskogee, Oklahoma, USA

After obtaining the entity-slot-filler triples, we extract examples from the document base for training and development.

3.2 Example Extraction

The training examples were drawn from the 2009's TAC KBP Entity Linking Sample Corpus. Due to time limitation we were not able to build a training set based on the 2010 document base. We indexed the document base using the *KBP_Toolkit* search tool provided by NIST, which had Lucene on its base.

In order to extract the training examples, we used the known entity and filler pairs, and looked for occurrences of these in the document base. Exact string match is used for both the entities and fillers. We looked for examples with up to 10 tokens between the entity and filler, and five words to surrounding the entity and filler. The examples are of the form:

```
5w entity 0-10w filler 5w
5w filler 0-10w entity 5w
```

where Nw corresponds to N words/tokens; for the middle span, this ranged from zero to ten.

Note that because we look for exact matches for the entity and filler, we miss examples that contain variations of the entity or filler strings.

For target entities for slot filling, we extracted examples that matched the string of the entity exactly. These examples are of the form:

```
30w entity 30w
```

Similarly, we miss examples that use different names for the target entity.

3.3 Training the Classifiers

For each slot, we trained a binary classifier that takes a text fragment with the entity and potential filler and decides whether or not the potential filler is an actual filler for the slot. We used Support Vector Machines (SVM) trained on the entity-slot-filler examples extracted from the document base (cf. Section 3.2). As explained in the previous section, the development of the system was done using the TAC-KBP 2009 dataset. Basically, our development consisted of feature set selection and setting of the SVM cost parameter (C).

For positive examples, we used examples containing the known entity and filler pairs based on slots derived from Wikipedia infoboxes. To avoid misleading infoboxes, we only used examples that had an entity type matching the entity type of the slot.

We did not use the Participant Annotation samples as positive examples due to time problems.

For negative examples, we distinguish between persons and organizations. For instance, given a specific classifier of slot i for person entity, the rest of the person slots were considered as negative examples. We followed the same strategy for slots of organization entities.

Regarding learning features, we considered three sets of features in order to develop the final system. The first set was based on the features introduced by Mintz et al. (2009). The second set was based on the features proposed by Zhou et al. in (Zhou et al., 2005). Finally, the third feature set was build by joining the previous two sets in one.

This way, for the first set we extracted the following feature types:

- The sequence of words between the entity and filler (10 words maximum).
- The part-of-speech tags of these words.
- The name-entity types of the entity and filler.
- A window of *k* words to the left of the first entity/filler and their part-of-speech tags
- A window of *k* words to the right of the second entity/filler and their part-of-speech tags.

Each lexical feature consists of a conjunction of all this components. We generate a conjunctive feature for each $k \in \{0, 1, 2\}$. Thus, Table 2 shows the resulting lexical feature (note that the each row in the table represents a single lexical feature).

For the second-type features (those based on (Zhou et al., 2005)), we extracted the following feature types:

- A flag indicating there is no word between the entity and filler.
- A flag indicating there is only one word between the entity and filler.
- The first word after the first-coming entity/filler.
- The last word before the second-coming entity/filler.
- All words between the entity and filler, except the first and last.
- The first word before the first-coming entity/filler.
- The second word before the first-coming entity/filler.
- The first word after the second-coming entity/filler.
- The second word after the second-coming entity/filler.
- The name-entities of the entity and filler.

Finally, the third feature set contained the features from both (Mintz et al., 2009) and (Zhou et al., 2005). Among all three features set, the best results were obtained deploying the first set (those introduced in (Mintz et al., 2009)). We think that this is because the conjunction of features yields high-precision features (but low-recall). With a small amount of data, this approach would be problematic, since most features would only be seen once, rendering them useless to the classifier. Since we use large amounts of data, even complex features appear multiple times, allowing our high precision features to work as intended. For Surprise Slot-Filling task we use the second set of features (Zhou et al., 2005). Due to the lack of training data, it is unlikely that complex features occur enough times for learning. So that we would expect higher recall by the use of independent features.

We used *svmperf*[2], which is an extension of *svm-light* to manage large sets of data, as implementation of a linear SVM classifier. We tried different values of C to tune the classifiers. The used values of C were 0.01, 1, 10, 20, 50, 100 and 200. We obtained the best results with $C = 10$ in the development dataset (cf. Section 3.2).

3.4 Applying the classifiers

Once the classifiers was trained, we used them to determine the most likely fillers for the target entities. Using the examples extracted from the document base for each entity, we identified potential fillers using a NER module or closed lists of strings (see Table 1). After identifying potential fillers within the span, we expanded the examples for target entities in entity-filler pairs (see Figure 1, test part). For each entity-filler pair extraction of features was carried out, and the prediction of the classifier in the slot was obtained deciding whether the filler was positive or negative.

For each entity-slot, we selected the positive the top-scoring filler for single-valued slots. Depending on the run (cf. Section 4) for multi-valued slot we returned the list of all positive fillers. If a slot had all the fillers with negative predictions, the system would return a NIL value for that slot (see Figure 1, "Output" part).

In the 2009 edition of TAC-KBP, slots like `place_of_birth`, `place_of_death`, `residences` and some others relation with locations were used. In 2010, this slots were separated into 3 parts; for example, instead of `place_of_birth` we got the following slots: `country_of_birth`, `stateorprovince_of_birth` and `city_of_birth`. We distinguish between countries, states and cities, after applying the classifiers . We used the GeoNames geographical database[3] to determine if the filler value was a city, country, state or province; also taking control if the filler value contains, for example, a country and a city.

In cases like the *Carrie Underwood* example mentioned before, our system will determine that "March 10, 1983" is a `DATE` while "Muskogee", "Oklahoma" and "USA" are `LOC`. We then map

[2] `http://www.cs.cornell.edu/People/tj/svm_light/svm_perf.html`

[3] `http://www.geonames.org`

ENTITY - SLOT - FILLER : Kim Il-Sung - date_of_birth - April 15, 1912				
SPAN: *Kim Jong-Il's late father <entity> Kim Il-Sung </entity> , who was born <filler> April 15, 1912 </filler>.*				
LEFT WINDOW	NE1	MIDDLE	NE2	RIGHT WINDOW
[]	PERSON	,/, who/WP was/VB born/VB	DATE	[]
[father/NN]	PERSON	,/, who/WP was/VB born/VB	DATE	[]
[late/JJ father/NN]	PERSON	,/, who/WP was/VB born/VB	DATE	[]
		...		

Table 2: Result of the conjuntive lexical features.

"March 10, 1983" to the date_of_birth slot "Muskogee" to the city_of_birth slot, "Oklahoma" to the stateorprovince_of_birth slot, and "USA" to the country_of_birth slot.

4 Results

Due to the limited time we had to build the entire slot-filling system, we were not able to tune our system. We submitted three runs based on different post-processing of the output of the classifiers. For the first run (UBC1), we submitted a basic system which, for each entity and slot, takes the filler that maximizes the prediction of the slot classifier. The system returns NIL if the SVM prediction is negative for all potential fillers within a slot.

In the second run (UBC2), for each entity, if its slot is single-valued we return the potential filler that maximizes the prediction of the classifier, as we did in the first run. But if the slot type is multi-valued, the system returns all positive potential fillers. For this submission we removed the slots that had to be ignored, as specified by the organization.

Finally, in the last run (UBC3), for each entity, we run all the slots looking for the one which maximizes the entity-slot-filler triple (as in UBC1 and UBC2). Depending on the type of the slot, it is treated differently. Given an entity, for each we first check if the slot is single-valued. In that case, we select the filler which maximize the slot. In the case that the slot is a location slot (org:headquarters, per:place_of_birth, per:place_of_death, per:residences), we select up to three countries, cities or states above a threshold of prediction confidence given by the classifiers. For the rest of the multiple valued slot we return the filler that maximizes the triple, in the same way we did for unique valued. Based on some preliminary results on the development dataset, we set the confidence threshold in -0.8 in order to increase the recall of the system. Again, we ignored some entity-slots, as specified by the organization.

Table 3 shows in the first three columns the official results in TAC 2010 KBP Slot Filling task, followed by the median. Although all the runs are very low, they show that the more sophisticate is the post-processing the more accurate are the results.

The last columns show the results for the Surprise slot-filling task. For this subtask we post-process the output of the classifiers as we did for UBC1: We returned for each entity and slot the filler with the maximum weight given by the SVM classifier.

5 Analysis

Although we were expecting low results, the obtained results are far from satisfactory. This lead us to analyze the outputs of our system. We next list some issues, and their possible solutions.

- **Lack of positive examples**. There were some slots without no positive examples in the training set.
- **Noisy positive examples**. Many of the gathered training examples were inaccurate for appropriate automatic learning. This means that we should apply some kind of filtering or instance weighting technique to get rid of useless examples.
- **Negative examples**. We generated too many negative example producing an unbalanced training set. Unbalanced training sets introduce undesirable biases in the learning process. Smart filtering of negative examples or weighted SVM classifiers might be a desirable solution to the problem. In Table 4 is possible to compare the number of tuples, positive spans and negative spans between slots.

- **Post-processing**. We treated the output of the classifiers equally. In other words, we did not take into account that each slot would need to tune its own threshold independently. This caused the system to select too many fillers for some slots like `per:title`. In addition, the post-processing phase could be improved by using semantic classes to constrain the final output of our system.

- **Suprise slot filling task**. The slots provided for the *surprise slot filling* task did not appear in Wikipedia infoboxes, so we could not apply our distant supervision strategy. On the other hand, the training provided by the organizers data was too small to train our classifiers.

6 Conclusions

We have participated with a preliminary implementation of a distant supervision system. The idea was to train the system using snippets of the document collection containing both entity and filler from the KB provided by the organizers (a subset of Wikipedia infoboxes). Our system does not use any other external knowledge source, with the exception of closed lists of words for religion, causes of death, charges and religious/political affiliation, plus the use of Geonames.

Our main goal was to setup a preliminary system, and we submitted three runs based on different post-processing options of the output of our classifiers, with results below the median.

The low results of our system in the main slot filling task, although expected, are far from satisfactory. The core Information Extraction module of our system is preliminary and buggy, with a few unsolved issues. One of the lessons that we have learned is that we first need to develop a traditional Information Extraction system and evaluate it on standard datasets (e.g. ACE relation detection task). We would then explore the challenges posed by the slot filling task proper, which include issues like getting false positive examples for training, or treating each slot as a separate problem.

Regarding the surprise slot filling exercise, the organizers released a few training examples for each target slot, where the examples where snippets of text where the entity, slot and filler were explicitly attested. Given the spirit of the main task (where a large number of entity-slot-filler triples from the KB had been made available), we were expecting such entity-slot-filler examples for the surprise task as well. This might partially explain our low results.

References

Eneko Agirre, Angel X. Chang, Daniel S. Jurafsky, Christopher D. Manning, Valentin I. Spitkovsky, and Eric Yeh. 2009. Stanford-ubc at tac-kbp. In *Proceedings of the Second Text Analysis Conference (TAC 2009)*, Gaithersburg, Maryland, USA, November.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 363–370, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of ACL-IJCNLP 2009*.

GuoDong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring various knowledge in relation extraction. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 427–434, Morristown, NJ, USA. Association for Computational Linguistics.

	MAIN TASK				SURPRISE TASK	
	UBC1	UBC2	UBC3	median	UBC	median
# filled slots in key	1034	1034	1034		505	
# filled slots in response	37	6398	109		3	
# correct non-NIL	1	3	5		1	
# incorrect/spurious	35	6380	103		2	
# inexact	0	9	0		0	
Recall	0.0009	0.0029	0.0048	0.1412	0.0019	0.1544
Precision	0.0270	0.0004	0.0458	0.2141	0.3334	0.5032
F1	0.0018	0.0008	0.0087	0.1054	0.0039	0.2363

Table 3: TAC 2010 KBP Slot Filling Results.

slot	**triples**	**pos. examples**	**neg. examples**	**slot**	**triples**	**pos. examples**	**neg. examples**
per:age	54	131	99790	org:alternate_names	1121	6690	182235
per:alternate_names	590	1755	98166	org:dissolved	227	1486	187439
per:cause_of_death	0	0	99921	org:founded_by	253	1225	187700
per:charges	10	25	99896	org:founded	1243	5379	183546
per:children	157	598	99323	org:headquarters*	13352	93277	95648
per:date_of_birth	146	249	99672	org:member_of	665	3924	185001
per:date_of_death	83	162	99759	org:members	137	855	188070
per:employee_of	1676	13871	86050	org:number_of_employees/members	422	2843	186082
per:member_of	2623	18440	81481	org:parents	4304	31233	157692
per:origin	207	1529	98392	org:political/religious_affiliation	1145	7268	181657
per:other_family	11	13	99908	org:shareholders	0	0	188925
per:parents	11	77	99844	org:subsidiaries	87	348	188577
per:place_of_birth*	4613	24426	75495	org:top_members/employees	6109	31502	157423
per:place_of_death*	1125	5554	94367	org:website	1265	2894	186031
per:religion	223	857	99064				
per:residences*	2835	17256	82665				
per:schools_attended	83	156	99765				
per:siblings	6	10	99911				
per:spouse	776	3391	96530				
per:title	2302	11416	88505				

Table 4: Statistics for all slots, including number of triples, positive and negative examples. The slots with an asterisk (*) belong to the 2009 TAC-KBP Slot Filling track, before separating them to cities, states, provinces or countries.