UBC at Slot Filling TAC-KBP 2011

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

Abstract

This paper describes our submissions for the Slot Filling task of TAC-KBP 2011. The system takes as baseline the one we developed for the 2010 edition (Intxaurrondo et al., 2010), which is based on distant supervision. We did a straightforward implementation, trained using snippets of the document collection containing both entity and filler from the KB provided by the organizers. Our system does not use any other external knowledge source, with the exception of closed lists of words for some of the slots. We submitted three runs based on different datasets and inference options on the output of each classifiers. Ours run are below the median, but we obtained significant improvements from our last system.

1 Introduction

This paper describes our participation in the TAC-KBP 2011 Slot Filling task. Our system is a straightforward implementation of a distant supervision system (Mintz et al., 2009). To develop this system, we took the one developed for last year's edition, following the same steps as in Intxaurrondo et al. (2010) and making some improvements. 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 some of the slots.

The paper is structured as follows. In Section 2 the Slot Filling task 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 inference heuristics to produce the output. Next, we will focus on the results obtained by our three runs. 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.

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 the possible fillers, and made explicit in the task guidelines.

3 Distant supervision system

In 2010, we tried a straightforward strategy for Slot Filling (Intxaurrondo et al., 2010), designed

around distant supervision (Mintz et al., 2009) and the joint work by Stanford and UBC in TAC-KBP 2009 (Agirre et al., 2009). This year we worked with the same system of 2010, and improved the results of the previous year.

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 above threshold, the system returns NIL.

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,...) to each slot, as described below.

We will now present the details of how we prepared the slot information, followed by how we extracted the textual fragments (examples) of entity occurrences, and by the method to train the classifiers. The application of the classifier to produce the Slot Filling results is explained next.

[1]We tried different inference strategies in the three submissions (cf. Section 3.5.2 and 4) following this idea.

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. Closed lists are used to improve the assignment of name-entities, they are taken from Surdeanu et al. (2010), as well as the regular expression for websites.

After obtaining the entity-slot-filler triples, we extract examples from the document base for training and development. The mapping of the infoboxes was made and provided to us by Mihai Surdeanu from Stanford University.

3.2 Train Example Extraction

The training examples were drawn from the 2010's TAC KBP Corpus. Due to time limitations we were not able to build a training set using all entity-slot-filler triples, so we used approximately 10%. 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 (see below).

We tried two variation. In the first we use all spans obtained for training and testing. Note that the spans

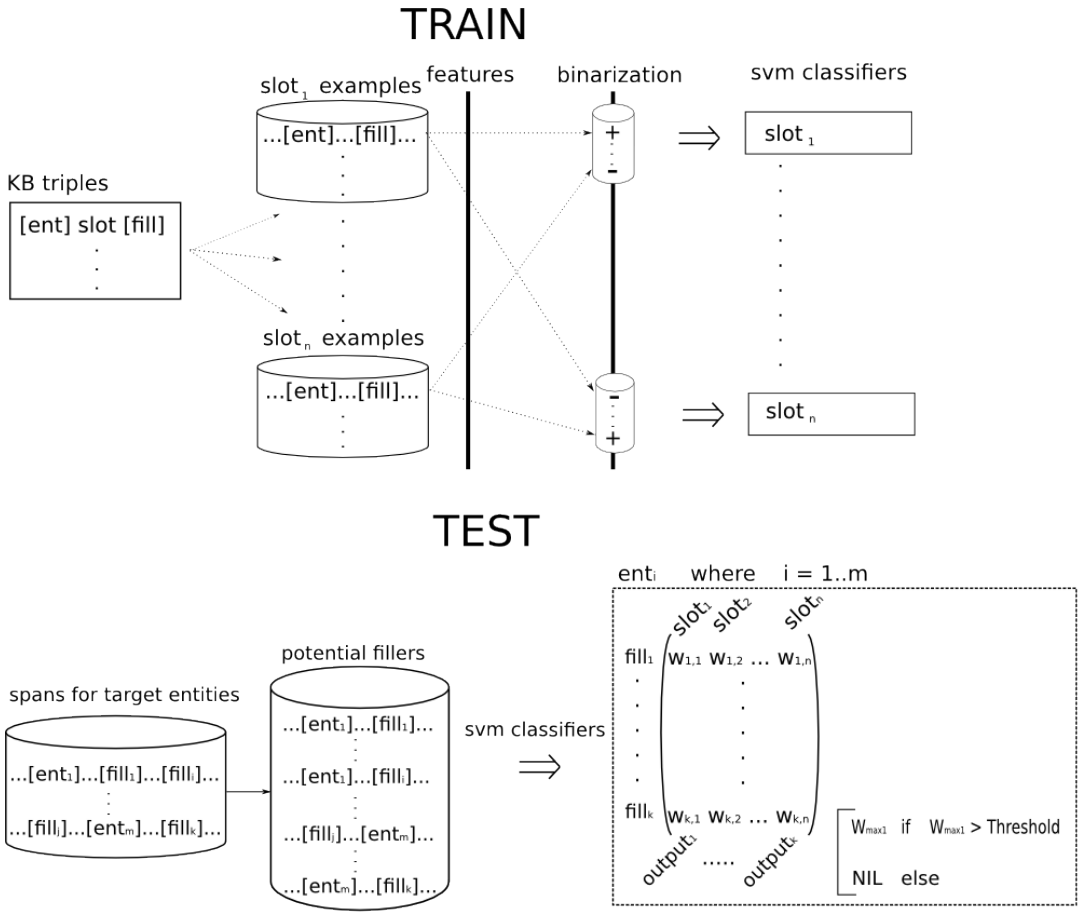

Figure 1: The architecture of the Slot Filling system. TRAIN: Extraction of KB triples, which are used to acquire training examples for each slot ($1..n$ slots) from the document base, followed by featurization and binarization. We then train n classifiers, one per slot. TEST: examples containing mentions 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, per:other_family, per:parents, per:siblings, per:spouse
NE (LOC)	org:city_of_headquarters, per:city_of_birth, per:city_of_death, per:cities_of_residence
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, org:country_of_headquarters, org:stateorprovince_of_headquarters, per:country_of_birth, per:country_of_death, per:countries_of_residence, per:stateorprovince_of_birth, per:stateorprovince_of_death, per:stateorprovinces_of_residence, per:cause_of_death, per:charges, per:origin, per:religion, per:title
RegExp	org:website

Table 1: Mapping of slot to NE type or closed list.

previously obtanined may contain parts of different sentences, mostly in the test set, between the entity and filler. In the second variation, we eliminate all spans containing periods (".") between the entity and filler. Each of this variations was tried in a different run, UBC1 and UBC2 respectively.

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). We deployed *svmperf*[2], which is an extension of *svmlight* to manage large sets of data, as implementation of a linear SVM classifier. 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.

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, in related experiments on the ACE 2005 dataset we carried out a selection of the learning features. Our system make use of the features introduced by Mintz et al. (2009), the ones proposed by Zhou et al. (2005), and some of the surface features proposed by Surdeanu et al. (2010).

This way, following Mintz et al. (2009) 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 $k \in \{0, 1, 2\}$.

Features based on Zhou et al. (2005) are the following 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.

And finally, features based on Surdeanu et al. (2010) are the following ones, with some extras:

- A flag indicating if the entity comes before the filler or the filler before the entity.
- Distance between entity and filler.
- A flag indicating the word form of the entity. If the entity is formed by more than one word, all these words are separated by "_"
- Some flags indicating all words in the entity separately. If the entity is formed by just one word, there will only be one flag.
- A flag indicating the word form of the filler. If the filler is formed by more than one word, all these words are separated by "_"
- Some flags indicating all words in the filler separately. If the filler is formed by just one word, there will only be one flag.
- Entity's part-of-speech.
- Filler's part-of-speech.
- Entity's name-entity type.
- Filler's name-entity type.

Table 2 shows the resulting lexical feature (note

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

that the each row in the table represents a single lexical feature).

3.3.1 Optimizing C

Due to the importance of the C parameter in SVM classifiers tried values of C ranging from 0.01 to 20 in the 2010 Slot Filling dataset. This way, we learnt the best C value for each of the submitted run, as shown in Table 4. for each run.

3.4 Getting test examples

In order to get examples where potential filler could be found for the target entities, we extracted examples in the document base that matched the string of the target entity exactly. These examples are of the form:

```
30w entity 30w
```

We also wanted to test whether examples of the variants of the target entity, as listed in Wikipedia, would increase the performance of the system. We used these additional test examples for the `UBC3` run.

3.5 Applying the classifiers

Once the classifiers were 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.

3.5.1 Optimizing the threshold

We learnt the optimum threshold to decide if a potential filler could be considered as a candidate filler. As we did with the C parameter with the classifier, for each best C we tried different threshold values of the classifiers predictions. We optimized the system according the threshold values between -1 and 1. The chosen parameter values where the ones that gave the best results with the target-entities used in the 2010 Slot Filling task.

If a slot had all the filler predictions below the threshold, the system would return a NIL value for that slot (see Figure 1, "Output" part).

3.5.2 Inference

Rather than returning the maximum filler directly, we first checked if the top-scoring filler was compatible with the slot; if the filler's name-entity type was compatible, we considered the potential filler as positive; if not, then we rejected that filler and checked if the next top-scoring filler was compatible or not; and so on until we found one. As an example, lets suppose that we are checking potential fillers for slot *per:date_of_birth*, the correct filler should be a date, but if we obtain as top-scoring filler a person's name, then we reject it and check the next one. This is the strategy used in our `UBC1` and `UBC2` runs.

As an additional piece of evidence, we also considered the frequency of a potential filler for each target entity. After checking the compatibility of each prediction, we take each potential filler and sum its prediction, even if that prediction is negative and below threshold in that slot. Once we obtain each ponential filler's sum, the take the top 3 sums of slots for that potential filler, check if every sum is above threshold for a slot, and if it is, consider that slot as a relation for the target entity and that potential filler. This was used in the `UBC3` run.

3.6 Improvements from TACKBP Slot Filling 2010 to 2011

The improvements of our Information Extraction system from the system developed in 2010 to the one developed in 2011 are the following:

- Better dataset: The *entity-slot-filler* triplets where less noisier in 2011. This constructed a cleaner dataset.
- Synonyms: Test sets were increased with more span examples. These extra examples contain synonyms of the target entity.
- Learning features: We learned last year that we needed to develop a supervised IE system before jumping to distant supervision. During 2011, we worked with the ACE 2005 corpus, using the same features as in 2010 from the beginning, to improve them, add more features,

ENTITY - SLOT - FILLER : Dominican University - org:city_of_headquarters - River Forest
SPAN: *...courses at <entity> Dominican University </entity> in the Chicago suburb of <filler> River Forest </filler> shortly before...*

LEFT WINDOW	NE1	MIDDLE	NE2	RIGHT WINDOW
[]	ORGANIZATION/ENTITY	in/IN the/DT Chicago/NN suburb/NN of/IN	LOCATION/FILLER	[]
[at/IN]	ORGANIZATION/ENTITY	in/IN the/DT Chicago/NN suburb/NN of/IN	LOCATION/FILLER	[]
[courses/NN at/IN]	ORGANIZATION/ENTITY	in/IN the/DT Chicago/NN suburb/NN of/IN	LOCATION/FILLER	[]
[courses/NN at/IN]	ORGANIZATION/ENTITY	in/IN the/DT Chicago/NN suburb/NN of/IN	LOCATION/FILLER	[shortly/RB]
[courses/NN at/IN]	ORGANIZATION/ENTITY	in/IN the/DT Chicago/NN suburb/NN of/IN	LOCATION/FILLER	[shortly/RB before/IN]
[at/IN]	ORGANIZATION/ENTITY	in/IN the/DT Chicago/NN suburb/NN of/IN	LOCATION/FILLER	[shortly/RB before/IN]
[at/IN]	ORGANIZATION/ENTITY	in/IN the/DT Chicago/NN suburb/NN of/IN	LOCATION/FILLER	[shortly/RB]
[]	ORGANIZATION/ENTITY	in/IN the/DT Chicago/NN suburb/NN of/IN	LOCATION/FILLER	[shortly/RB]
[]	ORGANIZATION/ENTITY	in/IN the/DT Chicago/NN suburb/NN of/IN	LOCATION/FILLER	[shortly/RB before/IN]

FEATURE TYPE	VALUE
WBF	in
WBL	of
WBO	the Chicago suburb
BM1F	courses
BM1L	at
AM2F	shortly
AM2L	before
DIR	ENTFILL
DIST	5
ENT	Dominican_University
ENT1	Dominican
ENT1	University
FILL	River_Forest
FILL1	River
FILL1	Forest
ENTPOS	NN
FILLPOS	NN
ENTTYPE	ORGANIZATION
FILLTYPE	LOCATION

Table 2: Example of the features used. The first 9 lines represent Mintz et al., while the next 7 lines correspond to Zhou et al., the last ones are from Surdeanu et al.

and analyze which features fitted better. The features that gave the best performance are used here.

- Optimization: Working with ACE, we also used SVM, and optimized the system using different C values with 5-fold cross-validation. Due that the final answers improved a lot with different values, we used the same technique in TACKBP 2011. SVM gives numerical predictions for each potential filler, using different thresholds to consider a filler as valid gives even better answers.

4 Results

The core of our system is the same used at the 2010 Slot Filling task (Intxaurrondo et al., 2010). We submitted three runs based on different datasets and post-processing of the output of the classifiers.

For the first run (UBC1), we used all the *entity-slot-filler* spans obtained as training dataset. Meanwhile, for the second (UBC2) and third (UBC3) run, we removed spans that contained periods between the entity and filler in the training set. The test set in the third run contains extra spans searched using synonyms of the target entity.

For the first and second run, we control if the top-scoring potential fillers for each slots and the slot type are compatible, checking their name-entity types (cf. Section 3.4.2). In the third run, apart of checking for compatibility, we sum their prediction values, and if their sum if above the threshold value, we check the top three slots where they have the maximun value.

Table 3 shows the values of the C parameter and prediction threshold used for each run.

Table 4 shows the official results in TAC 2011 KBP Slot Filling task, followed by the median. The second run shows an improvement when using spans with the entity and filler are in the same sentence.

	SVM - C value	Prediction Threshold
Run 1	5	-0.5
Run 2	1	-0.5
Run 3	5	-0.25

Table 3: Parameters used for each run.

	UBC1	UBC2	UBC3	median
Recall	2.96	2.85	3.28	10.31
Precision	4.45	5.36	4.74	16.50
F1	3.55	3.72	3.87	12.69

Table 4: TAC 2011 KBP Slot Filling Results.

We obtain the best results with the third run, adding spans with synonyms for the target entity and using frequency information.

4.1 What did not work

We will shortly review some of the techniques which did not work:

- Trying to remove noisy examples from the dataset, we took all *entity-slot-filler* examples and randomly took only one example per triplet. The f-measure decreased.
- Hoping to obtain better results, we combined all relations tagged in the ACE corpora with the Slot Filling task relations with the help of the annotation guidelines[3]. Due that ACE examples are tagged by hand and its lack of noise, we expected to increase the f-measure, but the final results were worse.

We also tried different the following heuristics for inference:

- For each slot, sum prediction values to each potential filler, and get the maximun filler as correct in case it was higher than the threshold value.
- For each potential filler, give as answer top three slots where they have highest prediction values only if the prediction values are higher than the threshold value.
- For each slot, we selected all potential fillers mentioned more than once, and then we calculated the average prediction value. We selected the potential filler with the maximun average value only if the average value was higher than the threshold value. Finally, if the selected filler was compatible with the slot, we considered it as valid.

None of them gave good results comparing to the combined heuristic used in run 3 (UBC3).

5 Analysis

Our system developed for the TACKBP 2011 is a significant improvement compared to the one developed for TACKBP 2010 ((Intxaurrondo et al., 2010)), but it's still weak, due to the following reasons:

- **Noisy positive examples**. Although this time the system had less noisy examples generated from the beginning, many of the gathered training examples were still 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.
- **Lack of positive examples**. There were some slots with no positive examples in the training set, such as *per:charges, per:children, per:other_family* and *org:shareholders*.
- **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 5 we show the number of tuples, positive spans and negative spans for the slots; note the excess of positive examples por slot *per:country_of_birth*, this slot makes person slots to be very umbalanced, fortunately this does not happen in organization slots. Slots like *per:cause_of_death* have no positive examples, having the maximun number of negatives.

6 Conclusions

We have participated with a preliminary implementation of a distant supervision system. The idea

[3] http://projects.ldc.upenn.edu/ace/docs/English-Relations-Guidelines_v5.8.3.pdf

slot	triples	pos. examples	neg. examples	slot	triples	pos. examples	neg. examples
per:age	81	152	43098	org:alternate_names	71	483	11344
per:alternate_names	41	522	42728	org:city_of_headquarters	214	2068	9759
per:cause_of_death	0	0	43250	org:country_of_headquarters	165	1235	10592
per:charges	0	0	43250	org:dissolved	93	335	11492
per:children	249	1943	41307	org:founded	32	103	11724
per:cities_of_residence	120	269	42981	org:founded_by	96	313	11514
per:city_of_birth	77	359	42891	org:member_of	47	176	11651
per:city_of_death	66	371	42879	org:members	155	1721	10106
per:countries_of_residence	252	986	42264	org:number_of_employees/members	37	127	11700
per:country_of_birth	2343	22593	20657	org:parents	70	370	11457
per:country_of_death	103	784	42466	org:political/religious_affiliation	189	2136	9691
per:date_of_birth	113	200	43050	org:shareholders	0	0	11827
per:date_of_death	7	8	43242	org:stateorprovince_of_headquarters	166	1190	10637
per:employee_of	171	2289	40961	org:subsidiaries	79	1434	10393
per:member_of	153	1018	42232	org:top_members/employees	49	127	11700
per:origin	235	1194	42056	org:website	6	8	11819
per:other_family	0	0	43250				
per:parents	90	1144	42106				
per:religion	94	564	42686				
per:schools_attended	64	145	43105				
per:siblings	5	10	43240				
per:spouse	89	237	43013				
per:stateorprovince_of_birth	75	268	42982				
per:stateorprovince_of_death	197	544	42706				
per:stateorprovinces_of_residence	474	7066	36184				
per:title	103	579	42617				

Table 5: Statistics for all slots, including number of triples, positive and negative examples used in UBC2 and UBC3.

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, and many more.

We submitted three runs, with different training and testing example settings, based on different post-processing options of the output of our classifiers. We have seen that using synonyms of the target entities improves the recall, and that inference can be used to filter wrong fillers.

Our main goal was to improve over last year's system, which we accomplished, but are still below the median. For the future we plan to focus on methods to deal with the noise in the examples.

Acknowledgements

We would like to thank Stanford researchers Mihai Surdeanu and Julie Tibshirani for providing us the Knowledge Base mapped to slots. Ander Intxaurrondo has a grant from the University of the Basque Country. Part of this research is funded by the European Commission (PATHS ICT-2011-270082) and the Ministry of Science and Innovation (KNOW2 TIN2009-14715-C04-01).

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.

Ander Intxaurrondo, Oier Lopez de Lacalle, and Eneko Agirre. 2010. UBC at Slot Filling TAC-KBP 2010. In *Proceedings of the Third Text Analysis Conference (TAC 2010)"*, Gaithersburg, Maryland, USA, November.

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

Mihai Surdeanu, David McClosky, Julie Tibshirani, John Bauer, Angel X. Chang, Valentin I. Spitkovsky, and Christopher D. Manning. 2010. A Simple Distant Supervision Approach for the TAC-KBP Slot Filling Task. In *Proceedings of the TAC-KBP 2010 Workshop*.

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.