# TAC KBP 2014 Slot Filling Shared Task: Baseline System for Investigating Coreference

**Heike Adel and Hinrich Schütze**
University of Munich
Center for Information and Language Processing (CIS)
Germany
heike.adel@cis.lmu.de

## Abstract

This paper describes our 2014 slot filling system. The system has been designed for the TAC KBP 2014 shared task. It consists of basic elements which have proved to be necessary in the previous years' evaluations. Since this year is our first participation in this task, we primarily intended to create a modular basic system which can be easily extended and improved in the following years. One of our main goals is the investigation of the impact of coreference resolution on the performance of slot filling systems.

## 1 Introduction

The TAC KBP slot filling task addresses the challenge of gathering information about entities (persons or organizations) from a large amount of unstructured text data. The previous evaluations showed that this task includes a variety of challenges like document retrieval, coreference resolution, cross-document inference and relation extraction / classification. This paper addresses most of these challenges (except for cross-document inference) and analyzes the suggested solutions in detail. We especially focus on coreference resolution and present how it effects the performance of our slot filling system.

The main focus of our system is modularity. Since this is our first participation in the shared task, the components provide mainly basic functionalities which, however, can be easily extended in order to improve the overall system's performance.

The remainder of the paper is organized as follows: First, an overview of the slot filling system is presented. Second, the different components of the system are described in detail and their performance and possible weaknesses are analyzed. Finally, the performance of the system in the shared task is presented.

## 2 System overview

Similar to previous slot filling systems, our system addresses the slot filling task in a modular way. This has several advantages, including extensibility, analyzability of the different components and modular development. Figure 1 shows the components of our system. In order to gather information about a person or organization and fill the pre-defined slots of the shared task, the following steps need to be performed:

- creation of a collection of possible aliases for the given name (alias component)
- retrieval of documents containing mentions of the entity (information retrieval component)
- retrieval of sentences with mentions of the entity and possible slot fillers (candidate extraction component)
- classification of the candidates (slot filler classification component)
- postprocessing of the candidates (postprocessing component)

In the following Section, the different components are described in more detail.

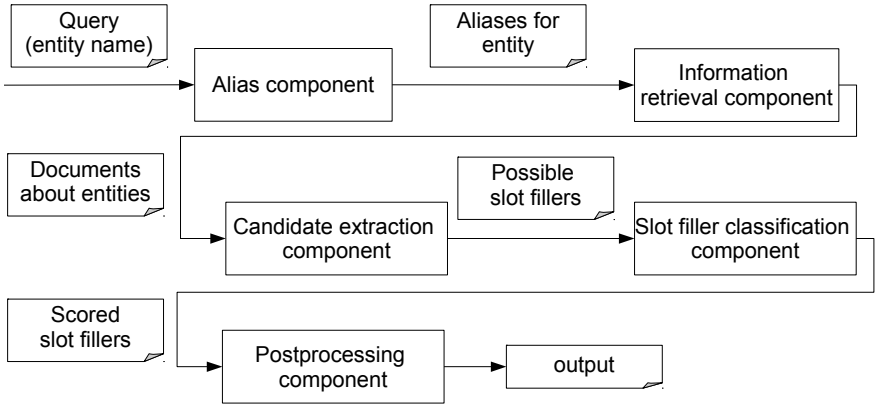


Figure 1: System overview: Basic components of our slot filling system

## 3 Component description

### 3.1 Alias component

The slot filling task is based on finding mentions of given entities in a large amount of text data. Since the corpus has been collected from various sources, the probability of finding name variations is rather high. In the 2013 annotations, for instance, more than 12% of all fillers occur in documents which do not contain the name of the person or organization as stated in the query file but a variation of it. The probability for name variations is especially high for non-English names which might have different spellings in English and for organizations which can be abbreviated in different ways. It is therefore not sufficient to only search for occurrences of the given name. In order to meet this challenge, we used a pre-compiled list of possible aliases. The aliases were obtained with JWPL (Ferschke et al., 2011) which provides a Java-based Wikipedia interface. For this study, we used a wikipedia dump from July 2014 and extracted all redirect information. The results were used as aliases (Qiu et al., 2012).

For the information retrieval component, we found that using all those aliases in addition to the given name leads to many falsly retrieved documents. Hence, we used only one "IR alias" for this component. This alias was chosen based on the lowest Levenshtein distance (Levenshtein, 1966) to the original name. This heuristic has two advantages: First, it is very simple and can be computed without any further information (e.g. there is no additional text data required). Second, it effectively covers spelling variations which still pose challenges to state-of-the-art information retrieval systems.

### 3.2 Information retrieval component

For document retrieval, we used the open source system Terrier (Ounis et al., 2006). We indexed the source documents of the slot filling task after some basic cleaning steps. To retrieve documents relevant to a given person or organization, the following queries were used:

- AND combination of the elements of the given name
- AND combination of the elements of the IR alias (see Section 3.1)
- OR combination of the elements of the given name

For each name, we extracted up to 100 documents.

Table 1 provides recall results for the information retrieval component on the previous evaluation data. Three different query combinations are compared: (1) search only for the name provided by the queries, (2) search for the name provided by the queries and for the IR alias of the name, (3) search for the name and all aliases. The results of the different query combinations show that using all aliases leads to lower recall values while adding or omitting the IR alias has almost no impact on the results.

For the 2014 evaluation, the combination (2) query name + IR alias was chosen to better cover name variations.

Table 1: IR recall results on previous evaluation data

| | (1) query name | (2) query name + IR alias | (3) query name + all aliases |
|---|---|---|---|
| 2011 recall | 74.3% | 73.9% | 65.6% |
| 2012 recall | 73.7% | 73.4% | 69.3% |
| 2013 recall | 82.9% | 82.5% | 82.5% |
| 2014 recall | 83.8% | 83.8% | 79.1% |

### 3.3 Candidate extraction component

To find the entity in the retrieved documents, we used the whole set of aliases as described in Section 3.1. Furthermore, we extended the list of aliases for organizations by replacing occurrences of "Organization", "Corporation", "Corp", "Corps" and "Co" with each other. For person names, we also added the name without middle name / initials if present. In order to find coreferred mentions of the given entity, the coreference component of CoreNLP (Manning et al., 2014) was applied.

After extracting sentences with mentions of the given entity, the system needed to detect possible fillers for each of the pre-defined slots. In order to reduce the amount of unnecessary candidates, the named entity recognition component of CoreNLP (Manning et al., 2014) was applied. Based on a manual mapping of slots to possible named entity types, only candidates with correct named entity types were extracted for the following classification step. In case of wrong named entity classification, this may discard correct fillers. However, this was taken into account in order to increase the precision of the system. Table 2 shows the recall results of the candidate extraction component. It shows the results for integrating the coreference resolution system as well as for leaving it out. Furthermore, the columns named "perfect IR component" show the potential of the candidate extraction component if only relevant documents are passed to it. The results show how crucial the performance of the IR component is for the candidate extraction and thus for the overall performance of the system. Moreover for future system improvements, it is important to note that the recall loss of the candidate extraction component is considerably higher than the recall loss of the information retrieval component.

The resulting slot filler candidates were then passed to a classifier which decided based on their contexts whether they were valid slot fillers or not.

### 3.4 Slot filler classification component

Teams in previous evaluations have used varieties of methods and models to extract and score candidates. Those models range from pattern based approaches (Gonzàlez et al., 2012; Liu and Zhao, 2012; Li et al., 2012; Qiu et al., 2012) over rule based systems (Varma et al., 2012) to classifiers like logistic regression classifiers (Malon et al., 2012) and support vector machines (Roth et al., 2013).

As classifiers for our basic slot filling system, we used pattern matchers and support vector machines (SVMs) with contextual bag-of-word features.

For pattern matching, we automatically extracted patterns from the official training data and cleaned them manually. Moreover, we integrated manual patterns based on the examples for the different slots from the official slot description document. In total, we used about 2-20 patterns per slot (4.5 in average).

For support vector machine classification, we trained one SVM per slot (with a linear kernel) and optimized its parameter $C$ on held-out validation data. To create training data for the SVM, we used distant supervision (Mintz et al., 2009). Our training data set consists of data from the following sources:

(1) official training data for each slot,

(2) example sentences of the official reference knowledge base,

(3) sentences of the source corpus, slot relations obtained using Freebase (Bollacker et al., 2008).

For (2), we used the officially provided mappings from the knowledge base relations to the slots and extracted example sentences as positive examples if they contained both relation entities (name and filler).

For (3), we manually mapped Freebase relations to the KBP slots. Then, we extracted all relations of Freebase which corresponded to one of the slots. Finally, we searched for these relation entities in the

Table 2: Candidate extraction recall results on previous evaluation data

| | system's IR component | | perfect IR component | |
|---|---|---|---|---|
| | without coreference | with coreference | without coreference | with coreference |
| 2011 recall | 14.3% | 16.7% | 16.8% | 20.5% |
| 2011 extracted fillers: FPs | 80,953 | 91,654 | 5,128 | 6,302 |
| 2012 recall | 18.29% | 25.9% | 21.0% | 28.6% |
| 2012 extracted fillers: FPs | 35,199 | 43,535 | 6,099 | 8,636 |
| 2013 recall | 22.1% | 30.7% | 25.6% | 34.3% |
| 2013 extracted fillers: FPs | 37,083 | 47,169 | 4,335 | 6,187 |
| 2014 recall | 22.9% | 28.2% | 22.6% | 27.5% |
| 2014 extracted fillers: FPs | 40,332 | 51,033 | 6,856 | 8,079 |

given source corpus and extracted all sentences as positive examples which contained both relation entities. This is similar to the approaches used by Min et al. (2012) and Roth et al. (2013).

In both cases ((2) and (3)), negative examples were created by applying the named entity recognition component of CoreNLP to the positive example sentences and extracting all words as negative slot fillers for a slot whose named entity type corresponded to the slot type but which were no valid slot fillers according to Freebase. We chose this approach of creating negative examples since it results in the same example format as the candidate sentences which the SVM classifier needs to cope with during the main evaluation.

Based on the resulting data, one SVM was trained per slot. If the ratio between negative and positive training examples was too extreme, the negative examples (which were typically much more than positive examples) were randomly subsampled until the ratio negative:positive was about 4:1. Afterwards, the data were split into a training and a validation set. The SVM was trained using the following features:

- bag-of-word vector of all words occurring left of the name and the filler,
- bag-of-word vector of all words between the name and the filler,
- bag-of-word vector of all words occurring right of the name and the filler,
- bag-of-word vector of all words in the sentence,
- ratio of capitalization of the filler,
- flag stating whether the name appears in front of the filler or vice versa.

The SVM results on the validation set for the different slots are provided in Table 3.

Despite all those different sources to create training data, there were still slots for which not enough training data could be extracted. Hence, no SVM classifier could be trained for them. Those slots were: per:cause_of_death, per:charges, per:other_family, per:title, org:date_dissolved, org:number_of_employees_members, org:website, org:political_religious_affiliation, org:shareholders.

Therefore, the final classification component worked as depicted in Figure 2. If no SVM could be trained for the current slot, only the pattern matching component was applied. In all the other cases, the SVM was used to classify the example. In order to get a meaningful confidence score for the slot filler, the result of the SVM decision function was transformed into a probability-like value (using the sigmoid function). Afterwards, the pattern matching component was applied. If a pattern matched, the SVM confidence score was increased by a fixed constant. Based on experiments with previous evaluation data, we set this constant to 0.3.

### 3.5 Postprocessing component

Finally, the classification component results were postprocessed: Based on a confidence threshold, the filler candidates were output as valid slot fillers or discarded.

In order to better train the SVM classifier, the "city-", "country-" and "stateorprovince-"slots had been merged to one "location-"slot since their fillers are expected to appear in the same contexts. Before outputting the results, this merging was reversed

Table 3: SVM results (F1-score) on the distant supervised validation set

| | | | |
|---|---|---|---|
| per:age | 0.864 | org:alternate_names | 0.710 |
| per:alternate_names | 0.796 | org:location_of_headquarters | 0.899 |
| per:children | 0.886 | org:date_founded | 0.840 |
| per:locations_of_residence | 0.791 | org:founded_by | 0.840 |
| per:location_of_birth | 0.855 | org:member_of | 0.849 |
| per:location_of_death | 0.759 | org:members | 0.917 |
| per:date_of_birth | 0.945 | org:parents | 0.939 |
| per:date_of_death | 0.784 | org:subsidiaries | 0.948 |
| per:employee_or_member_of | 0.835 | org:top_members_employees | 0.725 |
| per:origin | 0.689 | | |
| per:parents | 0.878 | | |
| per:religion | 0.779 | | |
| per:schools_attended | 0.769 | | |
| per:siblings | 0.921 | | |
| per:spouse | 0.921 | | |

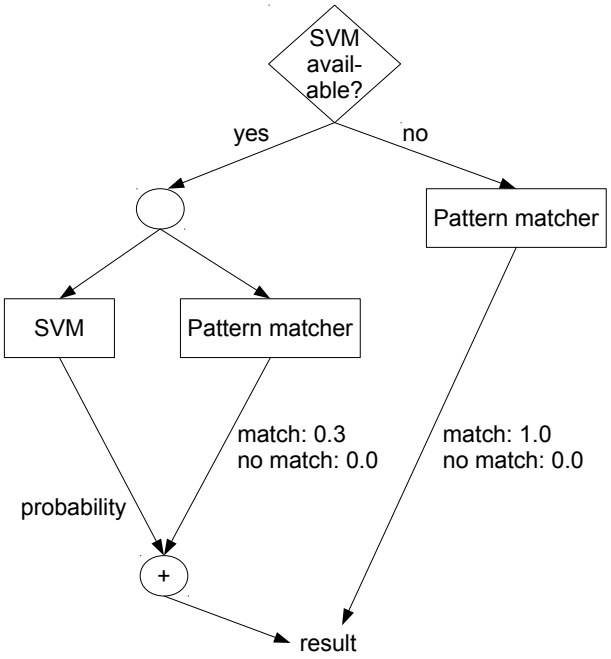


Figure 2: Slot filler classification

based on a country list[1] and a state list[2].

For single-valued slots, only the filler candidate with the highest confidence score was output. For list-valued slots, all filler candidates with confidence scores higher than the threshold were printed as results.

## 4 Submitted runs

For this year's evaluation, we submitted the following runs.

**Run 1: high precision run.** This run only reported answers with very high confidences (above 0.85). It can therefore be considered a high precision run.

**Run 2: slot specific and pattern specific thresholds.** We used slot specific and pattern matching specific confidence thresholds for reporting answers in this run. Those thresholds have been trained based on SVM results on distant supervised validation data. We sorted the answers of the SVM according to their confidence values (from high to low). Furthermore, we distinguished sentences for which a slot specific pattern matched from those for which no pattern matched. Hence, we established two sorted lists (one for pattern matches and one for no pattern matches). Then for both lists, we chose that confidence as the threshold which led to the highest F1 score when only considering the answers with higher confidences.

[1] http://www.listofcountriesoftheworld.com

[2] http://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations

Table 4: Detailed comparison of run 2 and run 4

| Run | # responses | # correct | # inexact |
|---|---|---|---|
| Run 2 (with coref.) | 824 | 43 | 13 |
| Run 4 (without coref.) | 648 | 34 | 8 |

**Run 3: slot specific thresholds.** This run can be regarded as a simplification of the second run since it used only slot specific confidence thresholds (without regarding pattern matches). Again, the thresholds have been extracted automatically as described in the previous paragraph.

**Run 4: no coreference resolution.** In this run, we used the same thresholds as in the second run. In contrast to the second run, our system did not apply the coreference resolution component. A comparison with the second run, therefore, directly shows the influence of the coreference resolution component on the overall results.

**Run 5: high recall run.** Finally, we submitted results of a high recall run. Here, our system reported all answers with a confidence above 0.55.

### 4.1 Result analysis

As described above, the main difference between the runs was the confidence threshold used for regarding a slot filling candidate as a valid slot filler. As expected, run 1 led to the highest precision and run 5 to the highest recall results. Regarding the F1-score, slot dependent thresholds (run 2, run 3) performed better than fixed thresholds (run 1, run 5). Run 2 led to the highest F1-score, namely 4.7%.

The analyses of the different components (see Section 3.2 and 3.3) already showed that coreference resolution helps to detect more correct slot filling candidates in the sentences. Run 2 and run 4 differed only in the application and omission of the coreference resolution system. While the precision was changed only slightly, the recall value and, hence, the final F1-score were improved by using coreference resolution (run 2). Hence, the system with coreference resolution was able to find more correct slot fillers (higher recall), but it also led to more false positive results (no change in precision). A more detailed comparison of run 2 and run 4 is provided in Table 4. It confirms the statements about the number of correct and false positive slot fillers.

Interestingly, coreference resolution mainly helped for cases where only parts of the name of the entities were mentioned in a sentence (e.g. only the last name of a person or only a short form of a company name). We have expected to also see improvements for pronouns. Reasons for why this was not the case could either be the kind of documents found for the query entities (news articles might refer to an entity rather by its name or an attribute than by a pronoun) or shortcomings of the coreference resolution.

## 5 Conclusion and future work

This paper presented the architecture of the CIS' system for the TAC KBP slot filling evaluation 2014. The system has been developed in a modular way. The paper showed analyses for its different components. This helps to point out the weaknesses of the system and improve it for the next evaluation.

For the next evaluation, we intend to improve all components, especially the candidate extraction and the classification component. The classification component, for example, will be extended by adding further classifiers like neural networks and interpolating their results with the SVM results. Furthermore, the classifiers will be optimized directly on the previous years' evaluation data instead of on distant supervision based validation data. This has been also reported by Roth et al. (2013). In order to improve the training of the classifiers, we also plan to extend and clean the distant supervised training data.

## Acknowledgments

This work was supported by DFG (grant SCHU 2246/4-2).

## References

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. ACM.

Oliver Ferschke, Torsten Zesch, and Iryna Gurevych. 2011. Wikipedia revision toolkit: Efficiently accessing wikipedia's edit history. In *Proceedings of the ACL-HLT 2011 System Demonstrations*, pages 97–102. Association for Computational Linguistics.

E. Gonzàlez, H. Rodríguez, J. Turmo, P. R. Comas, A. Naderi, A. Ageno, E. Sapena, M. Vila, and M. A. Martí. 2012. The TALP participation at TAC-KBP 2012.

Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions and reversals. In *Soviet physics doklady*, volume 10, page 707.

Yan Li, Sijia Chen, Zhihua Zhou, Jie Yin, Hao Luo, Liyin Hong, Weiran Xu, Guang Chen, and Guo Jun. 2012. PRIS at TAC 2012 KBP track.

Fang Liu and Jun Zhao. 2012. Sweat2012: Pattern based English slot filling system for knowledge base population at TAC 2012.

Christopher Malon, Bing Bai, and Kazi Saidul Hasan. 2012. Slot-filling by substring extraction at TAC KBP 2012 (Team Papelo).

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Bonan Min, Xiang Li, Ralph Grishman, and Ang Sun. 2012. New York University 2012 system for KBP slot filling.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Iadh Ounis, Gianni Amati, Vassilis Plachouras, Ben He, Craig Macdonald, and Christina Lioma. 2006. Terrier: A high performance and scalable information retrieval platform. In *Proceedings of ACM SIGIR'06 Workshop on Open Source Information Retrieval (OSIR 2006)*.

Xin Ying Qiu, Xiaoting Li, Weijian Mo, Manli Zheng, and Zhuhe Zheng. 2012. GDUFS at slot filling TAC-KBP 2012.

Benjamin Roth, Tassilo Barth, Michael Wiegand, Mittul Singh, and Dietrich Klakow. 2013. Effective slot filling based on shallow distant supervision methods.

Vasudeva Varma, Bhaskar Ghosh, Mohan Soundararajan, Deepti Aggarwal, and Priya Radhakrishnan. 2012. IIIT Hyderabad at TAC 2012.