University of Florida 2016 Slot Filler Validation system

Miguel Rodríguez
CISE Department
University of Florida
Gainesville, FL 32611, USA
mer@cise.ufl.edu

Daisy Zhe Wang
CISE Department
University of Florida
Gainesville, FL 32611, USA
daisyw@cise.ufl.edu

Abstract

In this paper we present the University of Florida 2016 Slot filler Validation (SFV) system. This is an on going work that started in the previous SFV evaluation where we presented a semi-supervised ensemble learning approach to aggregate the results from multiple slot fillers from the Cold Start track. In 2016 we introduce two major features. First, given the computational complexity of jointly reasoning over the complete set of input KBs, we present a query driven approach that reasons only over the set of candidate answers to a specific query. Our query driven approach also allow our system to be more selective answering 1-hop queries by evaluating only presumably correct answers from its corresponding 0-hop part. The second feature, is an extra layer of ensemble that combines signals from three types of uncertainty: (1) from the extractors, (2) from source documents and (3) from users beliefs. In addition to the new main features, we also used a distance metric to partially disambiguate shallow entity names from different slot filler runs.

1 Introduction

In the Cold Start Slot Filler (CSSF) task, teams were required to construct a knowledge base (KB) by extracting missing attributes of real world entities from a large text corpora. CSSF defines two types of queries, 0-hop and 1-hop, depending on the number of intermediate queries required to give a final answer. Each participating system should output a correct slot filler for every query (entity,relation) pair that can be found in the corpora. The task requires every slot filler to be accompanied by an extraction confidence and its provenance in the original corpus. Table 1 shows various slot fillers for a sample query and the number of runs that agree on the same extraction.

The results of participating CSSF systems motivates the Slot Filler Validation (SFV)track. As shown in Table 1, the collection of slot fillers from different systems for the same query vary greatly. Some runs agree and some others conflict in their answers. Furthermore, when they agree their confidence and provenance may not. SFV aims to automatically validate the result of multiple CSSF systems. The track gathers results in two formats. Filtering, where the SFV system judges every run individually discarding wrong fillers and ensemble, where the output of the SFV system is a new run that aggregates correct results from the input submissions.

In this paper, we provide an overview of our 2016 Slot Filler Validation system. The rest of the paper is outlined as follows. Section 2 describes our techinal approach. In section 3 we describe the evaluation and submitted SFV runs. In section 4 we discuss our official results, show examples of incorrect decisions, and comment on the improvements compared to the original CSSF submissions and our 2015 system.

2 Technical Approach

The SFV task can be casted as a binary classification problem. Given a set of candidate slot fillers for each query, determine the correctness of each. CSSF runs

Slot Filler	Run Count
Thomas	14
Earl Dolby	11
Ray Dolby	10
Dagmar	7
David	6
Musician	5
Ray	5
Inventor/Founder	5
Esther Dolby	3
Emmys	1
Alex Ferguson	1

Table 1: Slot fillers extracted by multiple systems for the query *(Ray Dolby, per:parents)*. The correct answers in the ground truth are *Earl Dolby* and *Esther Dolby*

themselves are the output of such classifier. If a run contains a candidate slot filler, it predicted its correctness with a certain confidence score, otherwise we represent its absence with zero confidence. The matrix $C = \{c_{ij}\}, i = 1, ..., n, j = 1, ..., m$ where n is the total number of candidate slot fillers and m is the number of participating runs, relates all candidate slot filler with its corresponding extractors.

$$C_{ij} = \begin{cases} Conf(j,i) & \text{if } c_i \in R_j \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

For instance, C can be used directly by an ensemble model such as performance weighting (Opitz and Shavlik, 1996), bayesian combination (Buntine, 1992) or meta-combinations such as Stacking (Wolpert, 1992). Simpler ensembles such as majoroty voting can be applied. A stacked ensemble approach to aggregate results from CSSF runs (Viswanathan et al., 2015) uses the assessed queries of previous years as training data to learn weights for the best run of each team participating in more than one year. All other runs are discarded from this approach. In 2015, we proposed to augment the stacking model using Consensus Maximization (Gao et al., 2009). CM is an ensemble model that combines the output of supervised and unsupervised classifiers casted as an optimization problem. Our 2015 system accommodated the first k columns of C as the best run from each system with previous participations. We trained a number of meta-

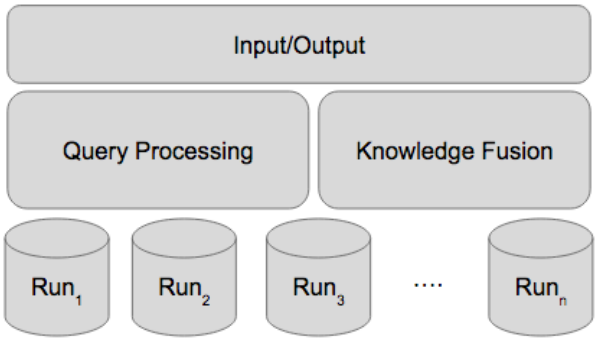

Figure 1: SigmaKB system architecture. The novel components include incorporation of CM Fusion over multiple KBs.

classifiers to provide diversity in the final ensemble and combined their output with the raw predictions from systems with no previous participations. That is columns $k+1, ..., m$ from C. CM objective function propagates conditional probabilities over C penalizing deviations from the initial labels assigned by supervised methods. The 2015 SFV system reasoned jointly over the complete set of queries, that is performed CM on C.

The system we present in 2016 instead of reasoning over the whole C, it reasons only over the subset that answers a specific query. Taking this approach, we make it more similar to a data integration problem. Figure 1 shows our system's architecture. At the bottom there is a data layer were every CSSF run reside in a relational database. In the middle layer we have a query processing module that combined with our fusion model is in charge of the aggregation process. Finally the top layer is the input/output layer in charge of taking queries in SQL, parse it, validate its correctness and pass it to the lower layer. When results are ready they can be retrieved as wel from this layer in JSON format. We named this aggregation system, SigmaKB (Rodríguez et al., 2016).

The Input/Output layer allows the user to submit queries to SigmaKB using SQL, allowing future incorporation of existing relational databases not in RDF format. The SQL layer can be easily augmented with additional systems like SEMPRE (Berant et al., 2013) that enable natural language queries. Query results are returned in JSON format sorted by posterior probability. Provenance is also

included in the results set. SigmaKB provides extractor, document and user provenances.

The query processing layer uses conventional data integration techniques to rewrite the query for each KB, and move the results into the Knowledge Fusion module. The KF module is uses the previous year evaluation data to train meta-combination models. Off line we create a matrix C using equation (1) for the previous evaluation dataset and use the first k (common) systems plus two other signals, the ratio of the total number of documents that provide an answer for the (entity, relation) pair to the number of documents that provide the slot filler, and the square root of the number of documents in the provenance for the (entity, relation, filler) triple. We use this models at query time and combine their output with raw extractions from other systems using our 2015 system CMF.

A new feature added in the 2016 system is the addition of other sources of uncertainty. We explored document uncertainty and user belief. Document uncertainty aims to provide a confidence value taking into account the credibility of the documents where a slot filler was found by the extractors. User belief aims to provide a perspective from the author associated with the extraction, this can be the author of a news article or the user of a web forum. We incorporate this two types of uncertainty using another level of meta-combination that takes extraction, document and user uncertainty as input and output a single probability of correctness for a given triple.

Similar to the matrix C where each candidate slot filler is associated to with the confidence score of its extractors, we create matrices D and U to relate each slot filler with the documents and users associated with the slot filler. Since multiple systems bay have extracted a slot filler from the same document, we take the median value of all confidences that relate an extractor with its corresponding document or user. To fin the posterior probability of each slot filler given D and U we can't use CMF as we did for fusion the extraction confidence since there is no overlap in documents or users with the previous evaluations. Therefore we follow the fully unsupervised approach for aggregating slot filler probabilities by JHU in 2013 (Wang et al.,). We cast an optimization problem that finds a probability distribution among the slot fillers given the confidences in U and D respectively.

$$\min_x \sum_{i=1}^{n} W_i (x_i - y_i)^2 \quad s.t \quad \sum_{i=1}^{n} x_i = 1, x_i > 0 \tag{2}$$

$$W_i = \sum_{j=1}^{m} w_j \tag{3}$$

$$d_i = \frac{1}{W_i} \sum_{j=1}^{m} w_j \frac{1}{2} d_j \tag{4}$$

Where n is the number of candidate slot fillers for a given query, and m the number of documents or users that provide at least one answer for the query. We use the same training strategy to train this meta-combination model and use it at query time to give each triple a final posterior probability.

Another minor addition to the 2016 system is the use of a similarity metric to disambiguate shallow entity names extracted by the input CSSF systems. We used the IDF Token Overlap described in (Galárraga et al., 2014) to calculate distance between pairs of candidate slot fillers. In case we find a confident similarity score between the two strings we merge the feature vectors and treat them as the same slot filler.

Finally, we discuss our strategy to fill in 1-hop query results. First, we obtain answers for the corresponding 0-hop query and use our system to obtain a posterior probability for the set of extractions. We then use the highest ranked answers to perform a set of 1-hop queries using all mentions of the slot filler. We finally select the slot filler with the max number of 1-hop correct fillers.

3 Experimental Evaluation

The SFV evaluation is carried out by a scorer provided for the task that uses a key file which pools all system responses and a manual assessment by human judges. The scorer then uses the assessment as ground truth to calculate precision, recall, and F1 score. These metrics are computed based on

the correct answers, the total number of system responses, and the total number of correct responses in the ground truth. Formally, the metrics are calculated as follows:

$$Recall(R) = Correct/Reference \quad (5)$$

$$Precision(P) = Correct/System \quad (6)$$

$$F1 = 2\frac{PR}{P+R} \quad (7)$$

The scorer reports results for 0-Hop and 1-Hop queries individually, and a general score for the complete submission. Since entities in the proposed query set may have multiple entry points or mentions in the corpora, the reported results are also divided into two types. LDC queries that don't take into account the query entry point and CSSF queries that treats each entry point as as separate query. Filtering and ensemble task are scored separately. For the filtering task, each filtered output is re-scored and the best score obtained among all filtered outputs is kept. The ensemble output is scored as a CSSF output.

To participate in the 2016 SFV task, we submitted three runs. Our first run uses the system described in the previous section, with a small adjustment. There are cases where some runs don't contribute slot filler for a given query, this run discards columns in the subset of C for a given query that don't have any value greater than zero. Our second run, doesn't use the second layer of ensemble and reports directly the posterior probabilities obtained by CMF. The third submitted run takes into account all columns in C for every query.

4 Experimental Results

The official SFV scoring metrics for each of the runs submitted are summarized in Table 2. The best performance of individual runs for each category is also included for comparison. Overall, our system outperformed the best individual runs and thus, achieving the ensemble purpose. Comparing the submitted runs, we can see that the addition of the second layer of fusion does not influence much the results. We think the way we used to capture document and user uncertainty were not appropriate and thus did not improve the results. The most interesting results from our runs is the precision gained by our 3rd run in 1-hop queries at the cost of recall. Since run 1 uses all runs and 1-hop queries are usually answered by a small number of systems with very low precision, by running CMF with all systems included, SigmaKB becomes more selective and avoids including incorrect 1-hop fillers that are penalized for having a wrong 0-hop counterpart.

	P	R	F1	Queries
Run 1	0.3667	0.4239	0.3633	0-Hop
Run 2	0.3694	0.4132	0.3575	LDC
Run 3	0.3065	0.3640	0.3044	
Best Run	0.2716	0.3016	0.264	
Run 1	0.2217	0.2898	0.2340	1-Hop
Run 2	0.1998	0.2683	0.2122	LDC
Run 3	0.2282	0.2319	0.2197	
Best Run	0.1567	0.1977	0.1638	
Run 1	0.3094	0.3709	0.3122	ALL
Run 2	0.3024	0.3560	0.3001	LDC
Run 3	0.2756	0.3118	0.2710	
Best Run	0.2262	0.2605	0.2244	
Run 1	0.3716	0.3589	0.3651	0-Hop
Run 2	0.3831	0.3338	0.3567	CSSF
Run 3	0.3689	0.3368	0.3521	
Best Run	0.4609	0.2437	0.3188	
Run 1	0.1307	0.2786	0.1779	1-Hop
Run 2	0.1279	0.2757	0.1747	CSSF
Run 3	0.4150	0.1789	0.2500	
Best Run	0.2528	0.1320	0.1734	
Run 1	0.2448	0.3320	0.2818	ALL
Run 2	0.2415	0.3143	0.2732	CSSF
Run 3	0.3778	0.2839	0.3242	
Best Run	0.3918	0.2063	0.2703	

Table 2: Results obtained by SigmaKB for LDC and CSSF queries and best run results on each category.

5 Conclusions

This paper presented our on going work on knowledge fusion and the development of SigmaKB. Our system approach consists on combining supervised stacked ensembles with unsupervised ESF outputs, document and user uncertainties. Our system, was able to improve upon individual extractors in the aggregate in general and also individual query types.

According to the literature, the proposed approach is the first one to incorporate document and user uncertainty in the slot filler validation, nevertheless we find the extraction method of uncertainty is not complete and thus doesn't influence our results compared to a single layer fusion. In our future work we will find better ways to capture such uncertainties and effectively add them into our pipeline.

References

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *EMNLP*, volume 2, page 6.

Wray Lindsay Buntine. 1992. *A theory of learning classification rules*. Ph.D. thesis, Citeseer.

Luis Galárraga, Geremy Heitz, Kevin Murphy, and Fabian M Suchanek. 2014. Canonicalizing open knowledge bases. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1679–1688. ACM.

Jing Gao, Feng Liang, Wei Fan, Yizhou Sun, and Jiawei Han. 2009. Graph-based consensus maximization among multiple supervised and unsupervised models. In *Advances in Neural Information Processing Systems*, pages 585–593.

David W Opitz and Jude W Shavlik. 1996. Actively searching for an effective neural network ensemble. *Connection Science*, 8(3-4):337–354.

Miguel Rodríguez, Sean Goldberg, and Daisy Zhe Wang. 2016. Sigmakb: multiple probabilistic knowledge base fusion. *Proceedings of the VLDB Endowment*, 9(13):1577–1580.

Vidhoon Viswanathan, Nazneen Fatema Rajani, and Yinon Bentor Raymond J Mooney. 2015. Stacked ensembles of information extractors for knowledge-base population. In *Proceedings of the 53rd annual meeting on association for computational linguistics. Association for Computational Linguistics*.

I-Jeng Wang, Edwina Liu, Cash Costello, and Christine Piatko. Jhuapl tac-kbp2013 slot filler validation system.

David H Wolpert. 1992. Stacked generalization. *Neural networks*, 5(2):241–259.