University of Florida DSR Lab System for KBP Slot Filler Validation 2015

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

Sean Goldberg
CISE Department
University of Florida
Gainesville, FL 32611, USA
sean@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 a Slot filler Validation (SFV) system that uses a semi-supervised ensemble learning approach to aggregate the results from multiple slot fillers from the Cold Start track. We apply Bipartite Graph-based Consensus Maximization (BGCM) to combine the output of supervised stacked ensemble methods with the output of slot filling runs that can't be trained. By using BGCM we are also able to leverage a small set of assessed fillers to increase the performance of the system. The ensemble results outperformed the best cold start run, the best filtered runs, and other ensemble systems.

1 Introduction

In the 2015 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. Formally, a CSSF system takes a list of entity-relation tuples or queries, and a large text corpus as inputs with the goal being to populate the relations or slots with their correct values. CSSF defines two types of queries (0-hop and 1-hop) depending on the number of required intermediate hops in the knowledge graph to answer the such query. Each participating system should output a correct response (slot filler) for every query that their system can find in the corpora. The task requires every slot filler to be accompanied by an extraction probability denoting 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 track. This track aims to automatically validate the result of multiple CSSF systems. As shown in Table 1, slot fillers from different systems may conflict, thus providing multiple answers for the same query. Even when they agree, their confidence and sources may not be the same. A SFV system take all inputs provided for CSSF (queries and corpus) plus a collection of CSSF. There are two outputs for the SFV track. Filtering, where the SFV system judges every run individually discarding wrong fillers. And ensemble, where the output of the SFV system is a new CSSF run that aggregates correct results from the input submissions.

In this paper, we provide an overview of our 2015 Slot Filler Validation system that aggregates the output of multiple CSSF runs. The preliminary experimental results obtained show a performance increase in 0-hop queries compared to the original CSSF submissions measured using F1 score, but deteriorates the performance of 1-hop queries. Nevertheless, the overall F1 score for the validation task was improved.

2 Technical Approach

The SFV ensemble task can be casted as a binary classification problem. Given a query and a slot filler, determine if the slot filler is correct or not. Since the CSSF runs provided for this task are themselves the output of such classifier, they can be directly used by an ensemble learning system. Ensemble classifiers have been used to enhance the prediction performance compared to us-

Slot Filler	Run Count
Roger Stolle	10
Bennie Thompson	9
Marco W. McMillian	7
Morgan Freeman	6
McMillian	6
Bessie Smith	5
Marco McMillian	3
Robert Johnson	3
Lawrence Reed	3
Meredith	2
Sharon J. Lettman-Hicks	2
Eliza Shook	1

Table 1: Slot fillers extracted by multiple systems for the query *(Clarksdale, gpe:residents_of_city)*. The correct answers in the ground truth are *Morgan Freeman* and *Marco W. McMillian*

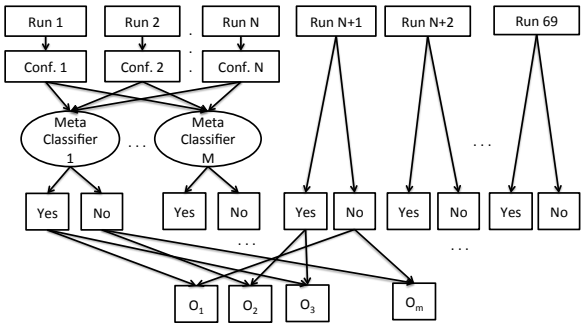

Figure 1: BGCM module input construction from CSSF outputs

ing a single model. The main idea behind ensemble learning is to aggregate the output of multiple weak classifiers to outperform the original ones. An ensemble framework consists of two building blocks, a basic inducer and combiner. Ensemble frameworks can be divided into two classes based on the basic inducer(Rokach, 2010): dependent frameworks where the output of a classifier is used to build the next classifier like AdaBoost (Freund et al., 1996) and independent frameworks where each classifier is built independently like like bagging (Breiman, 1996) or Random Forests (Breiman, 2001). Regardless of the inducer, the outputs are finally aggregated using weight-based combination methods like performance weighting (Opitz and Shavlik, 1996) and Bayesian combination (Buntine, 1992) or meta-combinations methods such as stacking (Wolpert, 1992). Ensemble classification has been successfully applied in a wide variety of applications including spoken language recognition (Ma et al., 2007), sentiment analysis (Whitehead and Yaeger, 2010) and mining concept-drifting from data streams (Wang et al., 2003).

A stacked ensemble approach (Viswanathan et al., 2015) to aggregate results from probabilistic extractions weights each system according to the system's performance on a training set. Since the training set is obtained from the assessed queries of previous years, only systems that participated in more than one edition of slot filling can be used by this approach dismissing slot fillers by new, potentially well ranked systems. Bipartite Graph-based Consensus Maximization (Gao et al., 2009) has the ability to overcome this problem. BGCM is an ensemble model that combines the output of supervised and unsupervised classifiers. BGCM is defined as a constrained optimization problem with an objective function that propagates conditional probabilities over a bipartite graph penalizing deviations from the initial labels assigned by supervised methods. Our SFV system uses multiple stacked ensemble models learned independently as the supervised classifier and combines their output with runs that can not be aggregated by stacked ensembles with BGCM. Furthermore, our SFV system uses BGCM's capability of leveraging the small portion of assessed slot fillers provided for the task. Figure 1 shows how our SFV system constructs a bipartite graph for BGCM from the inputs for the task.

The pipeline used by our BGCM-Based SFV system is shown in Figure 2. The first two steps are completed by the ESF systems that explore the corpus and extracting slot fillers for the given set of queries. In the next step, the runs from teams that participated in previous years are mapped together and ranked using the corresponding assessments. For 2015 runs, the small assessment file provided for the task is used for ranking. The best run of each mapped team is then passes to the feature extraction module and all other runs are passed directly to the BGCM module. The output of mapped runs are then grouped together to extract features.

The features used by our system are the same as

the stacked ensemble system (Viswanathan et al., 2015). The first feature vector contains the probabilities given by each system for the slot filler. In case a system did not provide a slot filler, its probability was set to zero. This feature vector has a total of 10 features. The second feature vector adds a one-hot encoding of the slot name as a nominal feature. The third feature vector adds provenance-based features to determine the level of provenance agreement of extractions. Each response is assigned a document agreement score determined by n/N, where n represents the number of systems that agree on the same document and N the total number of systems that answered the query. The third feature vector, adds an offset score by applying the Jaccard similarity coefficient between the offset of extractions that shared the same source.

Our system learns independent models for each feature vector using a number of stacked ensemble meta-classifiers trained using features from 2013 and 2014 extractions and their corresponding assessments provided by NIST. The stacked ensemble models only use the filler assessment for labeling purposes and disregards the document assessment. The final step of the stacked ensemble module is to produce a set of predictions for each (query, slot filler) tuple in the 2015 data set.

The BGCM module is then fed with the output of the stacked ensemble meta-classifiers. Since BGCM can take advantage of a small set of labeled data to improve the performance of the system, assessed results for 164 queries were fed into BGCM. To the best of our knowledge, no other team has used this assessed data to build their systems since it consists of a very reduced number of assessments. Nevertheless, BGCM can take advantage of it as extra evidence to bias the search procedure.

Finally, the post processing module produces answers for the two task specified by BGCM: filter and ensemble. Filtering is obtained by revisiting each original run and comparing each slot filler with the results obtained by BGCM, there is a match, the slot filler is marked correct, otherwise it is marked as incorrect. For the ensemble output, the post-processing module specifically enforces that all hop-1 queries have a correct hop-0 answer, selects the most probable answer for single value slots if multiple answers are found, and for each slot filler that comes from multiple systems, select the provenance from the extraction with highest probability.

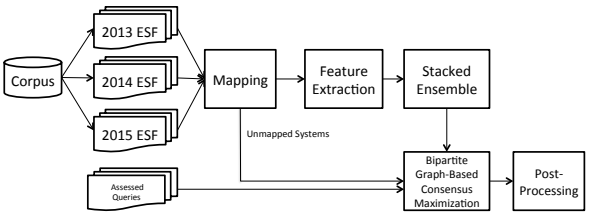

Figure 2: Slot Filler Validation System Pipeline

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 systems 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:

$$Recall(R) = Correct/Reference \quad (1)$$

$$Precision(P) = Correct/System \quad (2)$$

$$F1 = 2\frac{PR}{P+R} \quad (3)$$

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 2015 SFV task, we submitted three runs. All runs use the pipeline described before with 2 types of meta-classifiers, logistic regression and SVM. In the first run, each type of meta-

Method	P	R	F1
LR	0.648	0.335	0.441
LR + REL	0.662	0.343	0.452
LR + PROV + REL	0.634	0.374	0.470
SVM	0.639	0.319	0.425
SVM + REL	0.720	0.299	0.422
SVM + PROV + REL	0.729	0.298	0.423
Best SF 2014	0.585	0.298	0.395
BGCM	0.549	0.538	0.544

Table 2: Results of running our SFV system on the 2014 SFV dataset

classifier is independently trained with the three feature vectors as described in the previous section, stacked, relation and provenance. The supervised part of our first run uses training data extracted from the SF outputs of 2013 and 2014 for a total of 7 runs. The output of the other 62 runs were combined with the supervised outputs in the BGMC module. The second run, trains the same amount of meta-classifiers with extractions from 2014 run only, for a total of 12 runs, and combines them with the other 57 run in the BGMC module. The third run combines the outputs of the meta-classifiers trained for both previous runs, a set of 12 supervised outputs in total, combined with the 57 unused runs in the BMC module.

4 Experimental Results

The system system presented in this paper was developed using 2013 and 2014 data where the SFV task was less complex. Only 0-Hop queries were used, and less queries were asked. The results of running our SFV system trained on 2013 slot fillers and tested on 2014 slot fillers are shown in Table 2. The results show a 7.4% F1 improvement over the best stacked ensemble model, and a 14% F1 improvement over the best SF system for the prediction year. This promising results showed that including unsupervised evidence on top of the meta-classifiers of stacked ensemble is an interesting approach to the SFV problem.

The official SFV scoring metrics for each of the runs submitted are summarized in Table 3. The best performance of individual runs for each category is also included for comparison. For the complete submission, all three submitted runs achieve higher recalls than individual runs while also increasing precision. The overall F-scores also improves from individual runs. The first two boxes in Table 3 summarize the results of 0-Hop runs, where our systems clearly outperforms the best individual run by increasing recall while keeping a high precision. The results for 1-Hop queries summarized in the middle two boxes in Table 3 show a decrease in performance measured by F1. Our system failed to increase recall, although it significantly increased the precision of 1-Hop queries.

	P	R	F1	Queries
Run 1	0.5434	0.4602	0.4984	0-Hop
Run 2	0.5047	0.5157	0.5101	LDC
Run 3	0.4759	0.5361	0.5042	
KB_12_1	0.5048	0.3795	0.4333	
Run 1	0.4375	0.0882	0.1469	1-Hop
Run 2	0.4198	0.1155	0.1812	LDC
Run 3	0.3974	0.1303	0.1962	
SF_03_3	0.2381	0.2941	0.2632	
Run 1	0.5307	0.3247	0.4029	ALL
Run 2	0.4934	0.3698	0.4228	LDC
Run 3	0.4647	0.3882	0.4230	
KB_12_2	0.4373	0.2749	0.3376	
Run 1	0.4857	0.3750	0.4232	0-Hop
Run 2	0.4450	0.4360	0.4405	CSSF
Run 3	0.4130	0.4550	0.4330	
KB_12_1	0.4818	0.3090	0.3765	
Run 1	0.4172	0.0621	0.1081	1-Hop
Run 2	0.3135	0.0773	0.1240	CSSF
Run 3	0.3042	0.0858	0.1339	
SF_03_3	0.2325	0.2537	0.2426	
Run 1	0.4781	0.2520	0.3301	ALL
Run 2	0.4266	0.2951	0.3489	CSSF
Run 3	0.3975	0.3100	0.3483	
KB_12_2	0.4260	0.2170	0.2875	

Table 3: Results obtained by the BGCM-Based SFV system for LDC and CSSF queries and the best CSSF run for each category.

The main idea behind our SFV systems is to take into account the answers from potentially well-ranked runs that traditional stacked ensemble-based systems do not consider. For instance, from the 12

systems from 2015 that our stack ensemble models use, only one is ranked among the top 10 runs. Furthermore, the output of our system is composed only by 6% of fillers coming from the output of supervised learning.

Another interesting trend observed in our 2014 and 205 experiments is the balance achieved between precision and recall. All 2014 baselines are disproportionate (2014 stacked ensembles and 2015 SF). They have high precision, meaning a small number of false positives within the responses. At the same time, they have low recall, meaning a small number of correct facts with respect to the answer key. BGCM maintains or increases the precision of the baseline systems and also significantly increases the recall in order to achieve better F1 results.

5 Conclusions

This paper presented BGCM-based SFV system, a hybrid approach that combines supervised stacked ensembles with unsupervised ESF outputs, and a small number of observed labels to enhance the results of 2015 CSSF results. Our system, was able to improve upon individual extractors in the aggregate in general and specifically 0-Hop queries. According to the literature, the proposed approach is the first one to incorporate a small amount of assessed data to improve system performance in SFV. In general, many automatic knowledge base extractors such as the Never-Ending Language Learner (NELL) (Mitchell et al., 2015) use human guidance in the extraction process. Therefore, the use of Consensus Maximization is also relevant beyond the SFV framework to general knowledge base systems extractions.

References

Leo Breiman. 1996. Bagging predictors. *Machine learning*, 24(2):123–140.

Leo Breiman. 2001. Random forests. *Machine learning*, 45(1):5–32.

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

Yoav Freund, Robert E Schapire, et al. 1996. Experiments with a new boosting algorithm. In *ICML*, volume 96, pages 148–156.

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.

Bin Ma, Haizhou Li, and Rong Tong. 2007. Spoken language recognition using ensemble classifiers. *Audio, Speech, and Language Processing, IEEE Transactions on*, 15(7):2053–2062.

T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. 2015. Never-ending learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15)*.

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

Lior Rokach. 2010. Ensemble-based classifiers. *Artificial Intelligence Review*, 33(1-2):1–39.

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*.

Haixun Wang, Wei Fan, Philip S Yu, and Jiawei Han. 2003. Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 226–235. ACM.

Matthew Whitehead and Larry Yaeger. 2010. Sentiment mining using ensemble classification models. In *Innovations and advances in computer sciences and engineering*, pages 509–514. Springer.

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