Stacked Ensembles of Information Extractors for Knowledge-Base Population by Combining Supervised and Unsupervised Approaches

Nazneen Fatema Rajani **Raymond J. Mooney**
Dept. of Computer Science
University of Texas at Austin
{nrajani,mooney}@cs.utexas.edu

Abstract

The UTAustin team participated in two main tasks this year - the Cold Start Slot Filling (CSSF) task and the Slot-Filler Validation/Ensembling task, which was divided into the filtering and ensembling subtasks. Our system uses stacking to ensemble multiple systems for the KBP slot filling task, as described in our ACL 2015 paper. We expand the stacking approach by allowing the classifier to also utilize additions features that are relevant to making a final decision. Stacking relies on supervised training and hence requires common systems from the 2014 data to be used as training. However, that approach has limitations on performance and therefore we propose a novel approach of combining the supervised approach with an unsupervised approach on the remaining systems. We believe this combination approach gives our best run for the ensembling task. In this paper, we also discuss strategies to handle Cold Start data which comes from multiple hops.

1 Introduction

In 2015, UT Austin was a first time participant in the Slot Filler Validation/Ensembling task of the Text Analysis Conference (TAC) Knowledge Base Population (KBP) evaluation. Every year, many systems participate in the KBP evaluation, in particular, the slot filling task attracts many participants. However, the best performing team has an approximate $F1$ score of 32 and it remains more or less the same each year. Some of these top systems are high precision while others are high recall and thus overall $F1$ remained low. In order to get the best of both these type of systems, we were motivated to ensemble them and expectedly obtained good results. The details of this approach is described in our 2015 ACL paper (Viswanathan et al., 2015). Normal stacking trains a classifier to combine the output of multiple systems using as input features the output and confidence of each individual system. In particular, we use features capturing how well the systems agree about the provenance of the information they extract. Additionally, we also experimented with adding a feature that captures the cosine similarity between the document from which the query is extracted (i.e. the document given in the <doc id>XML tag) and the provenance document from which the slot fill was extracted. This feature helps capture whether the correct query entity is the one actually referenced in the provenance document. This feature boosted our performance even further when tested on the partially evaluated 2015 slot fills.

Our original approach relied on supervised training and hence required common systems from the 2014 data so that we could use their performance data from the previous year's evaluation to train the stacker. However, only 38 of 70 systems from 2015 had also participated in 2014 Cold Start or Slot filling task. Therefore, our initial stacking approach could not utilize about half the systems. To overcome this problem, we use an unsupervised approach to ensemble the remaining 32 systems and propose a novel stacking approach to combine the supervised and unsupervised approaches. We create an additional feature encoding the output of JHU's 2013 unsupervised approach (Wang et al., 2013) to

ensembling, and also add it as an input features to the stacker. We believe this approach gives our best run for the ensembling task.

To handle the round-1 and round-2 hashing problems in our approach, we use either the conservative strategy of throwing away round-2 extractions that do not have a matching round-1 result (used for the filtering task to increase precision) or using a more liberal strategy of including the corresponding round-1 result along with the round-2 result (used for ensembling task to increase recall). Our runs for the filtering task are focused on improving precision while those for ensembling are focused to improve F1, and thus adopt different methodologies based on the different evaluation metrics for the two tasks.

2 Ensembling Slot-Filling Systems

Given a set of query entities and a fixed set of slots, the goal of ensembling is to effectively combine the output of different slot-filling systems. As input, it takes the output of individual systems (in the format described in previous section) containing slot fillers and additional information such as provenance and confidence scores. The output of the ensembling system is similar to the output of an individual system, but it productively aggregates the slot fillers from different systems.

2.1 Algorithm

This section describes our ensembling approach which trains a final binary classifier using features that help judge the reliability and thus correctness of individual slot fills. In a final *post-processing* step, the slot fills that get classified as "correct" by the classifier are kept while the others are set to NIL.

2.1.1 Stacking

Stacking is a popular ensembling methodology in machine learning (Wolpert, 1992) and has been very successful in many applications including the top performing systems in the Netflix competition (Sill et al., 2009). The idea is to employ multiple learners and combine their predictions by training a "meta-classifier" to weight and combine multiple models using their confidence scores as features. By training on a set of supervised data that is disjoint from that used to train the individual models, it learns how to combine their results into an improved ensemble model. We employ a single classifier to train and test on all slot types using an L1-regularized SVM with a linear kernel (Fan et al., 2008).

2.1.2 Using Provenance

As discussed above, each system provides provenance information for every non- NIL slot filler. There are two kinds of provenance provided, the relation provenance and the filler provenance. In our algorithm we only use the filler provenance for a given slot fill. This is because of the changes in the output formats for the tasks over the years. Specifically, the 2013 format is comprised of three provenances namely filler, entity and justification while 2014 format has only two provenances namely filler and relation provenance. There was no English Slot filling task in 2015 but instead only the Cold Start task which had a very output format. However, systems were still ask to provide the filler provenance in their outputs. Hence, we use the filler provenance that is common between 2014 and 2015 formats. As described earlier, every provenance has a *docid* and *startoffset-endoffset* that gives information about the document and offset in the document from where the slot fill has been extracted. In their SFV system, Sammons et al. (2014) effectively use this provenance information to help validate and filter slot fillers. This motivated us to use provenance in our stacking approach as additional features as input to the meta-classifier.

We use provenance in two ways, first using the *docid* information, and second using the *offset* information. We use the docids to define a document-based provenance score in the following way: for a given query and slot, if N systems provide answers and a maximum of n of those systems give the same *docid* in their filler provenance, then the document provenance score for those n slot fills is n/N. Similarly, other slot fills are given lower scores based on the fraction of systems whose provenance document agrees with theirs. Since this provenance score is weighted by the number of systems that refer to the same provenance, it measures the reliability of a slot fill based on the document from where it was extracted.

Our second provenance measure uses *offsets*. The degree of overlap among the various system's offsets can also be a good indicator of the reliability of the

slot fill. The Jaccard similarity coefficient is a statistical measure of similarity between sets and is thus useful in measuring the degree of overlap among the offsets of systems. Slot fills have variable lengths and thus the provenance offset ranges are variable too. A metric such as the Jaccard coefficient captures the overlapping offsets along with normalizing based on the union and thus resolving the problem with variable offset ranges. For a given query and slot, if N systems that attempt to answer have the same *docid* for their document provenance, then the offset provenance (OP) score for a slot fill by a system x is calculated as follows:

$$OP(x) = \frac{1}{N} \times \sum_{i \in N, i \neq x} \frac{|offsets(i) \cap offsets(x)|}{|offsets(i) \cup offsets(x)|}$$

Per our definition, systems that extract slot fills from *different* documents for the same query slot have zero overlap among offsets. We note that the offset provenance is always used along with the document provenance and thus useful in discriminating slot fills extracted from a different document for the same query slot. Like the document provenance score, the offset provenance score is also a weighted feature and is a measure of reliability of a slot fill based on the offsets in the document from where it is extracted. Unlike past SFV systems that use provenance for validation, our approach does not need access to the large corpus of documents from where the slot fills are extracted and is thus very computationally inexpensive.

This year, we also experimented with features that use the query's provenance. The CSSF queries are provided to participants in XML format and has the query entity's ID, name, entity type, i.e., person or organization, the document where the entity appears and beginning and end offsets in the document where that entity appears. This is to help the participants disambiguate query entities that could potentially have same names but are different entities in reality. Following is an example of a query from the CSSF 2015 task:

```
<query id="CSSF15_ENG_0006e06ebf">
  <name>Walmart</name>
  <docid>ad4358e0c4c18e472c13bbc27a6b7ca5</docid>
  <beg>232</beg>
  <end>238</end>
  <enttype>org</enttype>
  <slot0>org:date_dissolved</slot0>
</query>
```

The document in the <doc id>tag can be thought of as the document that is most relevant to the query entity. We call it the query document for further references. Our feature involves measuring the similarity between this query document and the provenance document that is provided by the systems. The documents in the provenance as well as the query document are represented as TF-IDF vectors of fixed length given by the vocabulary size. Thereafter we use cosine similarity between the query document vector and the provenance document vectors. Thus every system that provides a slot fill, provides the provenance for the fill and thus has a similarity score with the query document. If the system does not provide a slot fill then its document similarity score is simply zero.

The idea behind this feature is that if a slot fill is extracted from a document that is very similar to the query document then it would have a high similarity measure. This means that those documents share many words that are relevant to the query and thus there is a high probability that the slot fill extracted is correct. Notice that the document similarity score is 1.0 if the slot fill is extracted from the same document as the query and thus is highly likely to be relevant to the query and thus also be correct.

2.2 Aggregating Confidence Scores for Remaining Systems

Our stacking approach described in Section 2.1.1 relies on shared systems between the years so that it can be trained on one year and tested on the next. Thus it restricts us to only use common systems over the years which would potentially affect our performance. In 2015, only 38 of the 70 systems were shared with 2014 and thus almost half of the systems could not be used by our stacking approach. To overcome this shortcoming, we propose a novel method of combining our stacking approach with an unsupervised approach for the remaining systems. For the systems that are not shared between the years, we use the constrained optimization approach for aggregating the confidence scores given in (Wang et al., 2013). They use an unsupervised technique to aggregate the "raw" confidence scores provided by

systems for each query and slot fill. The algorithm differs slightly for single-valued and list-valued slot fills and is described in detail in their paper. In their algorithm the authors use weights that relies on the performance of the system in the previous year. Since we use their technique on systems that are new and have not participated in previous years, we use uniform weights for all systems.

2.3 Stacking over the Unsupervised Approach

The output from the unsupervised approach is a list of slot fills and a single calibrated confidence score. In their paper the authors directly use the aggregated confidence scores to judge the slot fill. However we propose a unique way of using stacking over the unsupervised approach. We consider the output from the unsupervised approach as one system and add it to the stacker. The stacker is trained on remaining systems from 2014 that weren't shared with 2015. Along with the confidence scores, we also give the slot type as a feature. Additionally, we also tried using the average of provenance scores for unshared systems as a feature to the stacker. However that was not a part of our submitted runs and was experimented with later on.

We combine our original stacking approach with the stacking over unsupervised approach in the following manner. If the slot type is single valued then we simply take the one which has the highest confidence based on the stacker's prediction and if the slot type is list valued then we accept slot fills that cross a certain threshold. This threshold is learnt on 2014 data for each list valued slot type separately and is averaged for all queries. Thus we obtain a threshold for each list value slot type and accept all slot fills belonging to that slot type if it crosses this threshold.

2.4 Eliminating Slot-Filler Aliases

When combining the output of different SF systems, it is possible that some slot-filler entities might overlap with each other. An SF system *A* could extract a filler F_1 for a slot *S* while another SF system *B* extracts another filler F_2 for the same slot *S*. If the extracted fillers F_1 and F_2 are *aliases* (i.e. different names for the same entity), the scoring system for the TAC KBP SF task considers them redundant and penalizes the precision of the system.

In order to eliminate aliases from the output of ensembled system, we employ a technique derived by inverting the scheme used by the LSV SF system (Roth et al., 2013) for query expansion. LSV SF uses a Wikipedia anchor-text model (Roth and Klakow, 2010) to generate aliases for given query entities. By including aliases for query names, the SF systems increase the number of candidate sentences fetched for the query.

To eliminate filler aliases, we apply the same technique to generate aliases for all slot fillers of a given query and slot type. Given a slot filler, we obtain the Wikipedia page that is most likely linked to the filler text. Then, we obtain the anchor texts and their respective counts from all other Wikipedia pages that link to this page. Using these counts, we choose top N (we use N=10 as in LSV) and pick the corresponding anchor texts as aliases for the given slot filler. Using the generated aliases, we then verify if any of the slot fillers are redundant with respect to these aliases. This scheme is not applicable to slot types whose fillers are not entities (like date or age). Therefore, simpler matching schemes are used to eliminate redundancies for these slot types.

3 Experimental Evaluation

This section describes a comprehensive set of experiments evaluating ensembling for the KBP ESF task. Our experiments are divided into two subsets based on the datasets they employ. Since our stacking approach relies on 2014 SFV data for training, we build a dataset of one run for every team that participated in *both* the 2014 and 2015 competitions and call it the *common systems dataset*. There are 38 common systems from 10 common teams that participated in both 2014 and 2015. There are a total of 70 systems from 17 teams for the Cold Start task in 2015. The other dataset comprises of all 2014 SFV systems (including all runs of all 17 teams that participated in 2014). There are 10 systems in the common systems dataset, while there are 65 systems in the all 2014 SFV dataset. Table 1 gives a list of the common systems for 2014 and 2015 tasks. The value in the parenthesis is the number of systems that came from that team.

Common Systems
Stanford (1)
UMass (4)
UW (3)
CMUML (3)
BUPT_PRIS (5)
CIS (5)
ICTCAS (4)
NYU (4)
STARAI (5)
Ugent (4)

Table 1: Common teams for 2013 and 2014 ESF

4 Filtering and Ensembling Tasks

Although the 2015 CSSF task has two hop queries, we do not give special treatment to the hops and all our approaches treats the slot fills from both hops equally. Because of the way the validation script is designed for each hops, we faced round-1 and round-2 hashing problems. To overcome this problem, we designed the following two approaches - the *conservative* approach and the *liberal* approach.

If the stacker predicts that a round-2 slot fill is correct but its equivalent round-1 slot fill is not correct, then in the conservative approach we ignore the round-2 extraction do not include it in our submission. This potentially hurts recall but may improve precision of the slot fills. The filtering subtask focusses on maximizing precision of the SFV slot fills and thus we use the conservative approach for the this subtask.

On the other hand, if the stacker predicts that a round-2 slot fill is correct but its equivalent round-1 slot fill is not correct, then in the liberal approach we include the round-2 extraction along with it's equivalent round-1 extraction in our submission. This approach potentially hurts precision but hopefully improves recall of the slot fills. The ensembling subtask focusses on maximizing F1 of the SFV slot fills and thus we use the liberal approach for the this subtask.

5 Future Work

In our approach for ensembling or filtering the slot fills, we treat both round-1 and round-2 slot fills equally. Thus our stacker is trained on all the slot fills irrespective of which round it belongs to and is also tested on the entire SFV evaluation set. We believe this hurts our performance and thus we plan on looking into ensembling the round-1 and round-2 slot fills separately. Our plans include designing features that would allow us to have a more sophisticated approach for combining the round-1 and round-2 slot fills rather than the naive conservative and liberal approaches discussed earlier.

We also plan on utilizing the *enttype* information that is provided by systems in their slot fills to aid us in judging their correctness. Since our stacker trains on 2014 dataset which comes from the English Slot Filling task as opposed to the CSSF task, we believe our performance is affected because of that and we plan on finding ways to bridge this gap.

Acknowledgments

This research was supported by the DARPA DEFT program under AFRL grant FA8750-13-2-0026.

References

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Benjamin Roth and Dietrich Klakow. 2010. Cross-language retrieval using link-based language models. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 773–774. ACM.

Benjamin Roth, Tassilo Barth, Michael Wiegand, et al. 2013. Effective slot filling based on shallow distant supervision methods. *Proceedings of the Seventh Text Analysis Conference (TAC2013)*.

Mark Sammons, Yangqiu Song, Ruichen Wang, Gourab Kundu, et al. 2014. Overview of UI-CCG systems for event argument extraction, entity discovery and linking, and slot filler validation. *Proceedings of the Seventh Text Analysis Conference (TAC2014)*.

Joseph Sill, Gábor Takács, Lester Mackey, and David Lin. 2009. Feature-weighted linear stacking. *arXiv preprint arXiv:0911.0460*.

Vidhoon Viswanathan, Nazneen Fatema Rajani, Yinon Bentor, and Raymond Mooney. 2015. Stacked ensembles of information extractors for knowledge-base population. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages

177–187, Beijing, China, July. Association for Computational Linguistics.

I-Jeng Wang, Edwina Liu, Cash Costello, and Christine Piatko. 2013. JHUAPL TAC-KBP2013 slot filler validation system. In *Proceedings of the Sixth Text Analysis Conference (TAC2013)*.

David H. Wolpert. 1992. Stacked generalization. *Neural Networks*, 5:241–259.