Stanford at TAC KBP 2016: Sealing Pipeline Leaks and Understanding Chinese

Yuhao Zhang,* Arun Chaganty,* Ashwin Paranjape,*
Danqi Chen,* Jason Bolton,* Peng Qi, Christopher D. Manning
Stanford University; Stanford, CA 94305
{yuhaozhang, chaganty, ashwinpp}@stanford.edu
{danqi, jebolton, pengqi, manning}@stanford.edu

Abstract

We describe Stanford's entries in the TAC KBP 2016 Cold Start Slot Filling and Knowledge Base Population challenge. Our biggest contribution is an entirely new Chinese entity detection and relation extraction system for the new Chinese and cross-lingual relation extraction tracks. This new system consists of several ruled-based relation extractors and a distantly supervised extractor. We also analyze errors produced by our existing mature English KBP system, which leads to several fixes, notably improvements to our patterns-based extractor and neural network model, support for nested mentions and inferred relations. Stanford's 2016 English, Chinese and cross-lingual submissions achieved an overall (macro-averaged LDC-MEAN) F1 of 22.0, 14.2, and 11.2 respectively on the 2016 evaluation data, performing well above the median entries, at 7.5, 13.2 and 8.3 respectively.

1 Introduction

The TAC KBP 2016 challenge introduces two new multilingual tracks apart from English, namely Chinese and Spanish, as well as a new cross-lingual track. Consequently, we refined our existing English KBP system, by conducting thorough error analysis at every step of the pipeline and making incremental improvements. We also developed a new Chinese KBP system by combining pattern-based systems and distantly supervised systems. We then combined the output from these two systems to form our submissions to the cross-lingual track.[1] We describe the details of our contributions in this paper.

Our English KBP system is built on top of Stanford's 2015 KBP slot filling submission system (Angeli et al., 2015). A thorough error analysis of our performance on the 2015 challenge data leads to many minor and a few major system improvements: 1) We enhance our pattern-based extractors by hill-climbing on the dev set. 2) We use a new neural network-based extractor that is optimized for high recall. 3) We enhance our mention detection module by expanding our gazetteers and adding rules to correctly capture nested entites mentions, and we add rule-based extractors to better handle inferred relations. Our final submission system consists of 5 rule-based relation extractors, a self-trained supervised extractor and a supervised neural network extractor.

In addition to the English KBP system, we developed a completely new Chinese KBP system modeled around the English system's architecture. This Chinese KBP system has the following key differences from the English system: 1) We develop a completely new fine-grained Chinese NER system that is optimized for KBP. This new NER system combines a CRF model, a rule-based system and a gazetteer-based system to support 22 NER labels. 2) We use a new gazetteer-based Wikidict entity linker to perform entity linking. This enables fast computation and simple integration of our Chinese and English outputs for the cross-lingual submissions. 3) We implement a completely new Chinese relation extraction system that consists of pattern-based and rule-based extractors and a distantly su-

*The first five authors all made large, roughly equal contributions to the system.

[1] We did not develop a Spanish system.

pervised extractor. These new systems work together to form our final submissions to the Chinese tracks.

Below, we first introduce the overall pipeline and infrastructure of our KBP systems in Section 2. Then we provide detailed descriptions of our multilingual systems: in Section 3 for our English system and Section 4 for our Chinese system. We present the development results and the official evaluation scores of our submissions in Section 5, and finally we analyze our results and point to future directions in Section 6.

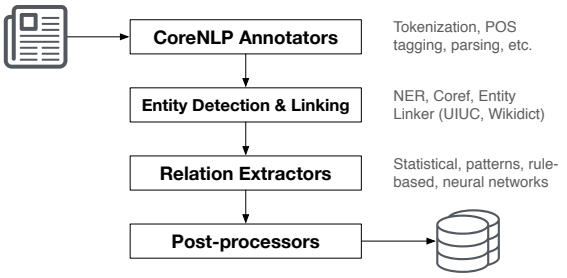

Figure 1: The Stanford KBP system pipeline. The input to the pipeline is a collection of documents, and the output of the pipeline are relation triples stored in database.

2 System Architecture

The architecture of Stanford's 2016 KBP system is largely the same as Stanford's 2015 system, and is described in detail in Angeli et al. (2015).

In earlier years, our slot filling systems used pipelines starting with an information retrieval (IR) component, which takes the query entities and returns relevant textual mentions and corresponding sentences from the corpus. Then these returned mentions and sentences were passed into downstream relation extractors for further processing. While this architecture has the advantages of being lightweight and saving a lot of computation in the preprocessing phase, it suffers from some critical drawbacks: 1) A fair amount of recall is lost at the beginning of the pipeline, due to the limitations of the IR system. 2) Probing the corpus becomes difficult, as the majority of the corpus remains unprocessed. 3) In order to retrieve multiple candidate mentions in a document, the IR system must be run for multiple iterations.

In this context, from last year, we have started to build our KBP system around a relational database, taking inspiration from Angeli et al. (2014b). During development, we store all processed documents from the corpus and intermediate data in the database. As most in-database operations are highly optimized and done in-memory, this architecture offers us a lot of benefits. 1) Since the preprocessing component is completely independent from the query entities, we can now annotate all text from the corpus and make use of all potential candidate mentions. 2) Evaluation now becomes optimized database queries instead of full relation extraction cycles, which enables us to do fast iteration on our algorithms. 3) SQL is a powerful data manipulation language that allows us to calculate data statistics and perform system diagnosis quickly.

At the core of this database-centered architecture are two relational tables: **sentence**, which contains textual information about sentences and supporting annotations (e.g., part-of-speech tagging sequences, dependency parsing, etc.), and **mention**, which contains textual information about entity mentions and supporting information of them (e.g., NER tags, provenances, canonical links, etc.). We now describe how the system makes use of this architecture to pipeline different components to produce final output.

2.1 System Pipeline

Our full system pipeline is pictured in Figure 1. The input to this pipeline is the original full-text TAC KBP corpus. This corpus is directly fed into an annotation component, where a series of Stanford CoreNLP (Manning et al., 2014) annotators, including tokenizer, part-of-speech (POS) tagger and parsers, are run to generate structured annotations of the text. The output of this component is used to populate the **sentence** table as described above.

Subsequently, we run our named entity recognition (NER) and coreference resolution annotators to generate NER tags for each sentence and coreference graphs for each document. Then tokens with NER tags of interest are organized together to form entity mentions. For each extracted entity mention, we run an entity linker over it to generate a canonical entity link used to universally describe this entity. We then use the output of this entity detection

and linking component to populate the **mention** table, where each mention entry is also connected to its corresponding sentence in the **sentence** table.

As we now have all the annotated information about mentions and their corresponding sentences, we then do simple database join operations on the mention table to form our pool of candidate mention pairs. Note that a candidate mention pair (m_1, m_2) is generated with the conditions that both m_1 and m_2 are present in the mention table and that they must co-occur in the same sentence in the original corpus. Afterwards, each candidate mention pair is passed into our relation extractors. The output of the relation extractors are a group of scored triples $(m_1, r, m_2) : p$ where r is either one of the forward relations as defined in the KBP slot descriptions, or a `no_relation` predicate, and p is a score that measures how confident the extractor is about this prediction.

Output triples from the relation extractors are then fed into a series of postprocessors. These postprocessors mainly serve three purposes: 1) Inverse relations are generated from all forward relation predictions. 2) Results from different extractors are merged according to our model ensembling policies. 3) We implement some constraints in the postprocessors to filter out predictions that are obviously wrong according to real-world knowledge, and predictions that contradict with others. A more detailed description of this component is presented in Angeli et al. (2014a). While the relation extractors are core to the entire system, this postprocessor component is also crucial, as it removes some of the salient errors inevitably generated by the upstream extractors to make sure our system has reasonable precision.

2.2 Supporting Infrastructure

We support the pipeline described above with distributed databases, specifically Greenplum DB, set up on two 20-core machines. Doing rapid iterations over the entire pipeline requires intensive large-scale database queries, which greatly benefit from having large memories and fast disk IO speed. Therefore, we set up our machines with 786GB RAM augmented by a 1.2TB PCI-E solid state drive that has a read speed of approximately 2.6GB per second. During development we find this infrastructure setup to be crucial to our quick system testing, problem diagnosis and parameter tuning.

We design both our English and Chinese KBP systems following the same architecture presented above, while the specific implementations of annotators, relation extractors and postprocessors are different across languages. We use separate database instances on different machines for different languages to enable fast development and evaluation.

3 English KBP System

Our English KBP system refines our 2015 system in many aspects. We summarize the implementation of this system and highlight some of the error analysis-driven improvements that we made.

3.1 Data

We only use the TAC KBP 2015 slot filling evaluation queries and assessment files to validate and test our English system, mainly for two reasons: 1) TAC KBP 2015 is the first year to provide multiple entry points for each query entity, thus evaluating on this dataset can give us unbiased estimation of system performance. 2) We use a dataset constructed from previous years' KBP assessment files to train one of our relation extractors.

We divide the 2015 query entities evenly into a dev set and an internal test set, containing 879 and 1104 mention-slot pairs respectively.[2]

3.2 Relation Extractors

In total, we use 7 relation extractors that can be broadly divided into three categories.

Rule-based We have 5 rule-based extractors in total. The first set includes a *Semgrex* pattern system (Chambers et al., 2007) and a *TokensRegex* (Chang and Manning, 2014) pattern system. *Semgrex* patterns operate on the dependency graph of a sentence and trigger a relation prediction once a specific predefined dependency pattern is matched between two entities. In contrast, *TokensRegex* patterns search linearly for specific templates in the word, lemma, POS and NER sequence of a sentence. We reuse all patterns in our 2015 KBP system, and add a number

[2]The imbalance here is due to the different number of mention-slot pairs for each query entity.

of new patterns by hill-climbing on the 2015 dev set. The output of the two pattern extractors is expected to be fairly precise.

Next we have three relation-specific rule-based extractors: *altnames*, *websites* and *gpe-mentions*. *altnames* is an extractor that infers alternate names of organizations and people from coreference chains of a document, and *websites* compares the edit distance between an organization name and an URL to give high-precision predictions of the `org:website` relation. They are described in more detail in Angeli et al. (2015). We explain the *gpe-mentions* extractor in Section 3.3.

Self-trained Supervised We reuse the same self-trained supervised extractor as in our last year's submission system (Angeli et al., 2015). In summary, at the core of this system is a traditional logistic regression-based classifier (LR) with manually-crafted features and a Long Short Term Memory network (LSTM) classifier. We first run the union of our patterns extractors and an Open IE system on the entire corpus. Since these systems are both of high precision, we collect their positive output predictions to form a training dataset. We add this "bootstrapped" training set along with a set of "presumed negative" examples into a pre-collected supervised training set, and use this entire dataset to train the two core classifiers above (LR and LSTM). We then take this output as the new training dataset, and repeat this process for another iteration. In this way, we train our statistical models with output from our own classifiers. We also apply other tricks to avoid class skew and overfitting as described in Angeli et al. (2015).

Neural Network-based Our neural network-based extractor makes use of a Conv-LSTM architecture that combines convolutional layers with an LSTM layer, as pictured in Figure 2. This Conv-LSTM model takes the original sentence as input, and generates embedding vectors for each word through a lookup layer. Then the embedding vector sequence is fed into a convolutional module consisting of three width-3 convolutional layers of 64, 128, and 256 dimension feature maps respectively, a width-2 max-pooling layer, and a width-1 convolutional layer of 128 feature maps. We use ReLU as the nonlinearity function for all convolutional layers. Then output representations from the convolutional component are sequentially fed into an LSTM network. Finally a max-over-time pooling operation is performed over all LSTM hidden states to generate the final sentence representation, which is then passed into the softmax layer for the final prediction. Empirically we find this Conv-LSTM architecture to work better than a simple LSTM model in the KBP setup. Our intuition is that the convolutional operations capture phrasal information that is crucial to the classification of the relation but is not directly captured by the LSTM model.

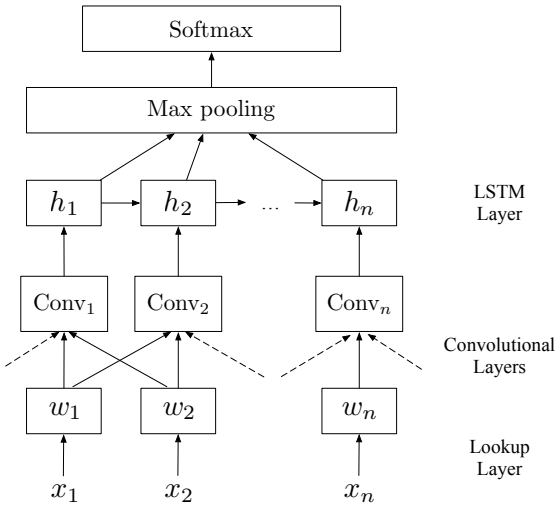

Figure 2: The architecture of the Conv-LSTM neural network relation extractor. Inputs are raw word sequences with entity indicators inserted, while output is the probability distribution over all possible labels for the corresponding subject-object pairs.

Our neural network model is trained on a fully supervised dataset that is constructed from previous years' KBP slot filling assessment files and is labelled by online crowd sourcing. We plan to make this dataset publicly available soon.

3.3 Error Analysis-driven Improvements

The independence of different components and the infrastructure built around use of a relational database enables us to do detailed analysis at every step of the pipeline. We find this analysis to be very helpful at understanding the F_1 score loss at each component, which drives us to make some key improvements in this year's system. In practice, we also monitor the effect on the final evaluation met-

rics, calculated over the development test set.

Improved Fine-grained NER Our analysis shows that the system is missing coverage of many job title and GPE mentions: The recall loss is 1.5% from incomplete job title coverage alone. Therefore, we made three improvements accordingly: 1) We wrote rules to capture nested job title mentions and added them into our gazetteers. For example, previously our system is only able to extract "*Officer*" as a job title when "*Chief Executive Officer*" was present in the sentence. We now extract both of them as entity mentions. 2) We further expand our gazetteers with all known job titles from previous years' KBP assessment files. 3) We add hyphenated location expressions, such as "*Seattle-born*" or "*London-headquartered*", that were previously missing from our systems. As it turns out, this improvement on the NER component alone can offer us about a 1.2 F_1 score boost on our hop-0 dev set queries.

Capture of Hierarchical Mentions Here we define "hierarchical mentions" as GPE mentions within an organization mention. For example, "*Texas*" in "*University of Texas, Austin*" is a GPE mention nested in an organization mention. We extract all mentions of this type by having a special gazetteer for GPEs and looking for GPE mentions within all organization mentions during the annotation phase. Then we implement a special rule-based *gpe-mentions* extrator that handles relations relevant to these entities. This extractor offers us about 1.3 F_1 score boost on the dev set.

Inferred GPE Relations One key drawback in our general pipeline is that extraction is only done at the sentence level,[3] and each mention pair will only be examined by our extractor once. As a result, if a third relation can only be inferred from two relation predictions, it will not be captured by any of our previous extractors. An example here is that in the sentence "*despite efforts by Nashville Mayor Karl Dean and the leaders of other Tennessee cities to derail the legislation*", if our system correctly predicts that "*Karl Dean*" is a resident of "*Nashville*" and we know from world knowledge that "*Nashville*" is a city in "*Tennessee*", we can infer that "*Karl Dean*" is also a resident of "*Tennessee*". An obvious danger of this type of inference is that it can easily bring up our total number of predictions and potentially harm our precision. Therefore, we only apply this inference to GPE-relevant relations where we believe the inference can be made fairly accurately through the incorporation of geographical knowledge.

[3]The use of coreference resolution systems in our pipeline introduces "cross-sentence" interactions that partially mitigate this restriction.

3.4 Model Ensembling

Model ensembling in our system is done by training an SVM classifier over multiple relation extractors as described in Section 3.2. The classifier takes as input a positive predicted relation produced by at least one of the extractors, and outputs a binary label of whether to keep this relation or not. The features we use fall into five categories:

1. An integer represents the count of the extractors that generates this prediction.
2. A one-hot vector that encodes the relation type.
3. A binary vector that encodes the systems that produce this candidate relation.
4. A one-hot vector that encodes the relation-system pair.
5. A vector that has scores at each index representing the confidence score corresponds to that relation-system pair. If a specific index corresponds to a different relation or a system that does not predict this candidate relation, that index will have a score of 0.

For the training objective we experimented with F_1 score, precision at K score (P@K), recall at K score (R@K) and Area Under the ROC Curve (AUC). We find that model ensembling with R@K as objective produces the highest final F_1 score on our dev set, and also performs better than simply taking the union of all the predictions.

4 Chinese KBP System

In addition to an English KBP system, for the 2016 challenge we developed from scratch a Chinese KBP system. Although the two systems largely share the same architectural design, the implementation of the Chinese system is largely limited by the scarcity of available datasets.

4.1 Data

During development we use the TAC KBP 2014 Chinese Pilot Regular Slot Filling task evaluation queries and assessment results to evaluate our system. This dataset contains 103 Chinese query entities. Therefore, we sample 50 entities as the dev set, and use the remaining 53 entities as the internal test set to avoid overfitting.

It is worth noting that the dataset described above has a different structure compared to the 2016 official evaluation. Specifically, in the regular slot filling task, no specific hop-0 and hop-1 slot will be provided at the evaluation time. Instead a system needs to extract all 41 slots for a given query entity. Therefore, during development time we are equivalently evaluating our system only on hop-0 queries by definition.

4.2 Improved Chinese NER System

As the original Stanford CRF-based Chinese NER tagger (Che et al., 2013) only supports 5 NER labels, we need to expand it to accommodate the KBP slot filling task. Our efforts can be categorized as follows:

Enhanced CRF-based NER We retrain our CRF-based NER tagger on the entire Chinese *OntoNotes 5* dataset. In addition, we empirically find that the resulting NER tagger does not generalize well to rare Chinese names (e.g., names of monks). Therefore, we apply a simple data augmentation trick by manually curating a list of 450 Chinese names and randomly replacing `PERSON` entities in a copy of the original dataset with names in this list. We then append this augmented dataset to the original training set and retrain the model. This simple trick gives over 1 F_1 score boost on the dev set. The resulting CRF-based NER tagger supports 7 labels: `PERSON`, `ORGANIZATION`, `LOCATION`, `GPE`, `FACILITY`, `DEMONYM` and `MISC`.

Rule-based Numeric NER We also implement a rule-based NER tagger for numeric entities. This rule-based tagger examines the word and POS sequence of a sentence, and assigns labels to words deterministically. Thereafter, a numeric value normalizer is applied to all extracted numeric entities to normalize the original value into corresponding Arabic numerals, which are then used as canonical entity links. This rule-based tagger supports 6 labels: `NUMBER`, `DATE`, `TIME`, `MONEY`, `PERCENT` and `ORDINAL`.

Gazetteer-based Fine-grained NER Additionally, we curate a gazetteer of common Chinese entities and construct *TokensRegex* patterns (Chang and Manning, 2014) from this gazetteer to label entities. All entities in this gazetteer are originally crawled from the Web and manually filtered by one of the authors. This gazetteer-based tagger adds another 9 NER labels into our system: `COUNTRY`, `CITY`, `STATE_OR_PROVINCE`, `TITLE`, `NATIONALITY`, `IDEOLOGY`, `RELIGION`, `CRIMINAL_CHARGE` and `CAUSE_OF_DEATH`.

4.3 Chinese Entity Linking

Unlike the English system, we apply a gazetteer-based entity linker in our Chinese system. Specifically, we use *Wikidict* as introduced in Spitkovsky and Chang (2012). We observe the original *Wikidict* contains noisy entries, therefore we apply heuristic rules to clean up and expand it. This results in the entire gazetteer having around 4.2 million entries. Moreover, a desirable property of *Wikidict* is that it uses English Wikipedia page URLs as canonical entity links. This largely enables us to naively merge the output from English and Chinese system and submit to the cross-lingual track.

4.4 Relation Extractors

The lack of a clean supervised dataset limited our use of statistical models during the development of Chinese relation extractors. Therefore we follow the following steps: We start with implementing a distantly supervised extractor optimized for recall. Next we add pattern-based extractors that achieve high precision, as well as compensate for missing relations in the distantly supervised training data. Finally, we implement several other relation-specific ruled-based extractors by hill-climbing on the dev set. We present detailed information of each extractor here:

Patterns The development of Chinese *TokensRegex* and *Semgrex* patterns is largely the same as in the English system. We manually add patterns to the

extractors that boost the dev set scores, while monitoring the test set scores to avoid overfitting. The final extractors contain 166 *TokensRegex* patterns and 470 *Semgrex* patterns respectively. This development expended approximately 30 person-hours of work writing patterns over the course of one week. It is also worth noting that we have implemented a new Chinese Universal Dependency (Nivre et al., 2016) converter and empirically find that *Semgrex* patterns on this Universal Dependency scheme work better than the original Stanford Dependency scheme.[4]

Distantly Supervised Extractor We build our distantly supervised extractor based around the Mintz system as described in Mintz et al. (2009). In a distantly supervised extractor, training instances are generated by applying a knowledge base to a corpus and labelling each sentence that contains co-occurrence of relation pairs in the knowledge base as a positive instance for the corresponding relation type. In our system, we acquire a knowledge base by using a combination of Freebase relation tuples and relation tuples extracted from previous KBP assessment results. We apply deterministic heuristic rules to convert the Freebase relation types to the KBP slot types. Applying this knowledge base to the Chinese KBP corpus gives us about 530K positive examples for 34 out of the 41 slot types. To balance training data across relation types, we apply a hard threshold of 5K to limit the number of training examples used for each relation type. This finally gives us around 80K training examples in total. Note that unlike the standard setup, we do not use any negative training examples in our distantly supervised extractor, as we empirically find that gradually adding negative training examples will slightly increase precision but decreases recall substantially. We plan to gain a better understanding of this in our future KBP work.

Other Rule-based Extractors Additionally, we implement an *altnames* extractor and an *org-subsidiaries* extractor. The Chinese *altnames* extractor performs inference over the coreference graph to extract `per:alternate_names` and `org:alternname_names` relations. Our *org-subsidiaries* extractor is based on a key observation that Chinese organization names are often structured in a clearly nested way. For example, in the case of the entity "中国作家协会河北分会", "中国作家协会" is a parent organization of the former and is nested inside the entity. Therefore, the extractor starts with a training phase where it accumulates a gazetteer of possible organizations by going through all extracted organization entities in the corpus. Then during the extraction phase, it examines the surface string of each extracted organization entity, and if a previously seen entity appears as a substring and this substring satisfies a set of lexical constraints (e.g., the suffix falls inside a lexicon), the extractor generates an `org:subsidiaries` relation for the entity pair of this substring and its full string. We optimize these two rule-based extractors to boost recall while preserving the precision.

Model ensembling in the Chinese system is done by taking a union of predictions from all extractors and using a hard per-query prediction count cutoff to select the most confident predictions. To make sure that our system achieves a reasonable precision by having all predictions from our rule-based extractors included in the final output, we manually set the confidence scores of all rule-based extractor predictions to be the maximum value 1. We tune the value of the cutoff using the dev set.

In addition, when optimizing our system for the Slot Filling task, we add a *StringMatch* module at the annotation phase. This module is essentially a simplified version of an IR system. During annotation, this module "peeks" at the evaluation query entities, and searches for words or phrases that can match the surface form of the query entities and makes them mentions. The output of this module is then combined with the output from the original entity detection module to populate the Chinese **mention** table. This component is not applicable to the Cold Start Knowledge Base Population setting.

5 Results

We report dev scores of our systems, as well as the evaluation results on the official 2016 evaluation set. We focus on submissions to the slot filling tracks. For the official evaluation we mainly report on macro-averaged LDC-MEAN scores.

[4]The new converter is made publicly available in the latest Stanford CoreNLP release.

5.1 Development Scores

A summary of development scores of our English systems is shown in Table 1. We want to highlight that by having an ensembled system that is explicitly trained to optimize system recall, we acquire a system that achieves the highest overall recall and F_1 on the dev set.

A summary of development scores of our Chinese systems is shown in Table 2. Note that the ensembled system has the highest recall, while the highest F_1 is achieved by the pattern-based system. We expect the ensembled system to perform better in a recall-bound environment.

5.2 Official Scores

For KBP 2016, Stanford made 5 submissions to the English slot filling track. These submissions were generated by different combinations of extractors and were meant to have different balances between precision and recall. A summary of official scores is shown in Table 3. All scores are reported using the macro-averaged LDC-MEAN calculation. Our submissions were:

Stanford_ENG_1 A model ensemble of all extractors that is expected to have balanced precision and recall on hop-0 queries and medium precision on hop-1 queries.

Stanford_ENG_2 A model ensemble that is optimized for medium precision on hop-0 and hop-1 queries.

Stanford_ENG_3 A model ensemble of all extractors that is expected to have high recall.

Stanford_ENG_4 A model ensemble of patterns, altnames and websites that is expected to have high precision.

Stanford_ENG_5 A model ensemble of all extractors that is balanced for both hop-0 and hop-1.

For the Chinese slot filling track, Stanford made 4 submissions. A summary of official scores is shown in Table 4. Again all scores are reported using the macro-averaged LDC-MEAN calculation. These submissions were:

Stanford_CMN_1 A high-precision model ensemble of all patterns and rule-based extractors. The *StringMatch* module is applied for higher recall of query entities.

Stanford_CMN_2 A model ensemble of all extractors with the *StringMatch* module applied. This is expected to have balanced precision and recall.

Stanford_CMN_3 A high-precision model ensemble of all pattern extractors with the *StringMatch* module turned off. This is our most conservative submission.

Stanford_CMN_4 Same as **Stanford_CMN_1** except that the *StringMatch* module is turned off.

In addition to the slot filling tracks, Stanford also made 3, 2 and 4 submissions to the English, Chinese and cross-lingual KB tracks respectively. In Table 5 we show our best scoring system for each KB track. The best-performing English system is a high-recall ensemble of all extractors, and the best-performing Chinese system is an ensemble of pattern extractors and a distantly supervised extractor. A combination of a balanced English system and a high-recall Chinese system gives us the best cross-lingual scores.

6 Discussion

For both the English slot filling and KB tracks, our highest scoring systems are the high-recall system (Stanford_ENG_3) on the macro-averaged LDC-MEAN metric and the balanced precision-recall system (Stanford_ENG_1, with an F1 score of 18.5) on the micro-averaged LDC-MAX metric. We note that the LDC-MEAN metric equally weights every query, favoring systems that on average do well, even if there are a few queries for which it produces spurious results. By this reasoning, our high-recall system is probably our best system on this highly recall-bound task. We are encouraged to see that in the slot filling track our model ensembling of all extractors did the best amongst our submissions on the LDC-MEAN metric, which it was explicitly trained to maximize.

For the Chinese tracks, on the micro-averaged LDC-MAX metric our highest scoring system is the model ensemble of all rule-based extractors combined the *StringMatch* module (Stanford_CMN_1,

System	Hop-0			Hop-1			All		
	P	R	F_1	P	R	F_1	P	R	F_1
Rule-based	60.9	18.0	27.8	46.0	6.7	11.7	**56.8**	12.6	20.6
Self-trained Supervised	28.6	22.5	25.2	12.6	13.2	12.9	19.7	18.1	18.9
Neural Network-based	30.7	27.7	29.1	9.0	13.5	10.8	17.0	20.8	19.0
Ensembling with Recall@K	–	–	–	–	–	–	37.5	**26.3**	**31.0**

Table 1: Performance of our English systems on the dev set (constructed from the 2015 slot filling assessment files).

System	P	R	F_1
Patterns	**42.4**	26.3	**32.4**
Distantly Supervised	21.8	29.8	25.2
Ensembling (Distantly Supervised + Patterns + Others)	24.0	**34.4**	28.3

Table 2: Performance of our Chinese systems on the dev set (constructed from the 2014 Regular Slot Filling assessment files). Note that all scores are equivalent to hop-0 scores by definition.

System	Hop-0			Hop-1			All		
	P	R	F_1	P	R	F_1	P	R	F_1
Stanford_ENG_1	21.0	19.9	18.5	7.4	8.5	7.4	15.7	15.4	14.2
Stanford_ENG_2	26.7	23.1	22.8	10.3	11.8	10.3	20.3	18.7	17.9
Stanford_ENG_3	26.5	30.2	26.0	15.1	19.5	15.9	**22.0**	**26.0**	**22.0**
Stanford_ENG_4	19.9	16.4	16.6	6.4	5.9	5.9	14.6	12.3	12.4
Stanford_ENG_5	21.0	19.9	18.5	3.2	4.4	3.2	14.0	13.8	12.5

Table 3: Official scores (macro-averaged LDC-MEAN) of Stanford's submissions to the KBP 2016 English slot filling track.

System	Hop-0			Hop-1			All		
	P	R	F_1	P	R	F_1	P	R	F_1
Stanford_CMN_1	15.3	13.2	13.3	4.6	5.5	4.9	10.5	9.7	9.5
Stanford_CMN_2	16.9	21.6	16.2	6.7	10.9	7.7	**12.3**	**16.8**	**12.4**
Stanford_CMN_3	14.2	11.8	12.1	4.6	5.5	4.9	9.9	8.9	8.8
Stanford_CMN_4	14.6	12.5	12.6	4.6	5.5	4.9	10.1	9.3	9.1

Table 4: Official scores (macro-averaged LDC-MEAN) of Stanford's submissions to the KBP 2016 Chinese slot filling track.

Track	Hop-0			Hop-1			All		
	P	R	F_1	P	R	F_1	P	R	F_1
English	25.4	30.2	25.2	17.1	19.2	17.0	22.1	25.9	22.0
Chinese	13.3	22.2	14.5	12.1	18.2	13.7	12.7	20.4	14.2
Cross-lingual	12.9	16.2	13.0	8.9	11.5	9.3	10.9	13.9	11.2

Table 5: Official scores (micro-averaged LDC-MEAN) of Stanford's submissions to the KBP 2016 KB tracks.

with an F1 score of 16.1). We find that this system achieved the highest hop-0 and hop-1 precision among all slot filling systems, while preserving a medium recall. Additionally, compared to the same system without the *StringMatch* module (Stanford_CMN_4, with an F1 score of 15.1), this simple trick gives us about 1 F1 boost, by improving the hop-0 recall of the system. On the macro-averaged LDC-MEAN metric, our highest scoring slot filling system is the model ensemble of all rule-based extractors and the distantly supervised extractor (Stanford_CMN_2). This consolidates our belief that in a recall-bound environment, the system benefits by having a high-recall extractor (i.e., the distantly su-

pervised extractor).

We highlight a few directions for future work. For the English system, we believe that further performance gains can be obtained by exploring better neural architectures for both entity detection and relation classification. For the Chinese system, we believe that: 1) Improving the entity linking module by making use of more data and a context-aware model can greatly improve the system performance. 2) Incorporating an Open IE relation extractor can potentially boost the recall of the existing system. 3) We observe that "zero anaphora" is a much more common grammatical phenomenon in Chinese. Specifically, we find that the subject entity of a sentence is sometimes missing, especially in the newswire context. Therefore having an annotation module that can automatically infer missing subject entities in Chinese can potentially lead to substantial performance gains.

7 Conclusion

In this paper we have presented the design and implementation of Stanford's TAC KBP 2016 multilingual slot filling and knowledge base population systems. The central messages that we want to highlight are: 1) Easy incremental gains at each step help improve the overall performance of our relatively mature English KBP system more effectively than focussing on just one aspect of the pipeline. 2) A Chinese relation extraction system can be created in a relatively short period of time by doing fast iterations on patterns combined with training statistical models on distantly supervised data.

Acknowledgments

The Stanford KBP 2016 team would like to acknowledge Gabor Angeli and Victor Zhong for their useful discussions about the 2016 system and great efforts in the development of Stanford's previous KBP systems.

References

Gabor Angeli, Arun Chaganty, Angel Chang, Kevin Reschke, Julie Tibshirani, Jean Y Wu, Osbert Bastani, Keith Siilats, and Christopher D Manning. 2014a. Stanford's 2013 KBP system.

Gabor Angeli, Sonal Gupta, Melvin Jose, Christopher D Manning, Christopher Ré, Julie Tibshirani, Jean Y Wu, Sen Wu, and Ce Zhang. 2014b. Stanford's 2014 slot filling systems. In *Proceedings of the Seventh Text Analysis Conference (TAC 2014)*.

Gabor Angeli, Victor Zhong, Danqi Chen, Arun Chaganty, Jason Bolton, Melvin Johnson Premkumar, Panupong Pasupat, Sonal Gupta, and Christopher D Manning. 2015. Bootstrapped self training for knowledge base population. In *Proceedings of the Eighth Text Analysis Conference (TAC2015)*.

Nathanael Chambers, Daniel Cer, Trond Grenager, David Hall, Chloe Kiddon, Bill MacCartney, Marie-Catherine De Marneffe, Daniel Ramage, Eric Yeh, and Christopher D Manning. 2007. Learning alignments and leveraging natural logic. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 165–170.

Angel X. Chang and Christopher D. Manning. 2014. TokensRegex: Defining cascaded regular expressions over tokens. Technical Report CSTR 2014-02, Department of Computer Science, Stanford University.

Wanxiang Che, Mengqiu Wang, Christopher D. Manning, and Ting Liu. 2013. Named entity recognition with bilingual constraints. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 52–62.

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*.

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*, pages 1003–1011.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*.

Valentin I Spitkovsky and Angel X Chang. 2012. A cross-lingual dictionary for English Wikipedia concepts. In *LREC*, pages 3168–3175.