Overview of UI-CCG Systems for Event Argument Extraction, Entity Discovery and Linking, and Slot Filler Validation

Mark Sammons[1] Yangqiu Song[1] Ruichen Wang[1] Gourab Kundu[1]
Chen-Tse Tsai[1] Shyam Upadhyay[1] Siddarth Ancha[2] Stephen Mayhew[1]
Dan Roth[1]
[1]Department of Computer Science, University of Illinois at Urbana-Champaign
Urbana, IL 61801
[1]{mssammon,yqsong,rwang11,kundu2,ctsai12,upadhya3,mayhew2,danr}@illinois.edu
[2]Indian Institute of Technology Guwahati, Guwahati - 781039, India
[2]ancha2@iitg.ernet.in

November 11, 2014

Abstract

In this paper, we describe the University of Illinois (UI_CCG) submission to the 2013 TAC KBP Event Argument Extraction (EAE), English Entity Discovery and Linking (EDL), and Slot Filler Validation (SFV) tasks. We developed three separate systems. Our Event Argument Recognition system infers world knowledge from event argument overlaps to improve performance of a recognition/labeling pipeline. Our Entity Discovery and Linking system builds on the Illinois Wikifier, augmenting its candidate identification capabilities and using a clustering algorithm for cross-document coreference. Our Slot Filler Validation system follows an entailment formulation that evaluates each candidate answer based on the evidence present in the source document it refers to.

1 Illinois Event Argument Extraction System

We adopted a two step approach for event extraction, and used a heuristic rule to induce world knowledge to improve performance. For event extraction, we first detected the triggers and then detected the arguments. For world knowledge induction, we computed statistics over ACE 2005 data to identify argument overlaps between event sub-types.

1.1 EAE System Description

The UI_CCG Event Argument Extraction system uses a two-stage approach. The first stage identifies event trigger words, and the second identifies event argument mentions.

1.1.1 Description of Training Data

We used ACE2005[1] as our training data, as its event taxonomy is similar to that of the Event Argument Extraction task. We trained classifiers to recognize event triggers and event argument mentions; in the following sections, we describe the features we used.

To detect the triggers, we train the classifiers based on the trigger features. To detect the arguments, we train the classifiers based on trigger features, mention features, and pair features.

[1]http://www.itl.nist.gov/iad/mig//tests/ace/ace05/

1.1.2 Trigger Features

- Context/lemma/postag (left, right, center) unigram, bigram.
- WordNet extension (hypernym, hyponym, lemma, similar).
- Wether the trigger is in FrameNet, VerbNet, PropBank.
- VerbNet Class.
- Concepts from ESA: We extract concepts using explicit semantic analysis (ESA) [9, 10]. The Wikipedia corpus is the 2013-10-01's dump. We used 200 concepts for each piece of text.
- Word2vec: We extract the distributional representation based on Mikolov's neural network [12, 13]. We train the vector ourselves on the 2013-10-01's Wikipedia dump. The window size was set to be default value (five), and the dimension was set to be 200.

1.1.3 Mention Features

- Context/lemma/postag (left, right, center) unigram, bigram.
- WordNet extension (hypernym, hyponym, lemma, similar).
- Concepts from ESA.
- Word2vec.
- Head word.
- Coreference mention (bag-of-words).
- Named entity types from general named entity recognizer (Illinois NER [17], Stanford NER [8] and OpenNLP[2]).
- Entity type/subtypes from ACE.

1.1.4 Pair Features

- Paths (through relation/node) from Stanford dependency parser [11].
- Paths (through word/postag) from Stanford parse tree.
- Common ancestor (left/right path).
- Position of argument (right, left, overlapped).

1.1.5 Description of the Machine Learning Model

We used a stage wise classification approach to extract the events [1, 2]. The flow of the classification is shown in Fig. 1.

- Step 1.1: We train a binary classifier to detect triggers of events. "True" means a testing word is a trigger.
- Step 1.2: We train a 8-class classifier to classify the triggers into eight types of events. We train a 33-class classifier to classify the triggers into 33 sub-types of events.
- Step 2.1: We train three sets of classifiers for argument detection. One binary classifier based on all the arguments; eight binary classifiers for the eight types; 33 binary classifiers for the 33 sub-types.

[2] `https://opennlp.apache.org/`

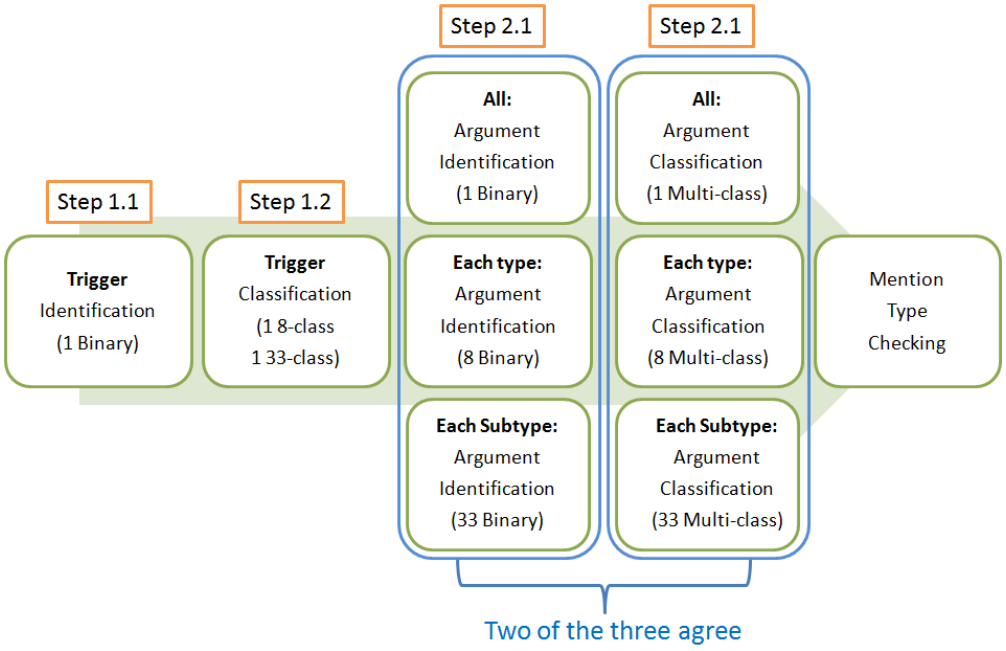

Figure 1: Event Extraction Pipeline.

- Step 2.2: We train three sets of classifiers for argument detection: one multi-class classifier based on all the arguments; eight multi-class classifiers for the eight types; 33 multi-class classifiers for the 33 sub-types.

In Steps 2.1 and 2.2, we implemented a voting mechanism based on three sets of classifiers to make decisions. If two of the three sets of classifiers agree, then we recognize the corresponding mention as the argument of the event.

1.2 Description of the Heuristic Rule of World Knowledge Induction

We observed that a single argument can participate in multiple events simultaneously, and developed a method to automatically determine whether or not an argument can participate in multiple events.

We denote an event mention as $e_i = \{t_i, a_{i,1}, ..., a_{i,m_i}\}$ where t_i is the trigger of e_i, m_i is the number of arguments in e_i, and $a_{i,k}$ is the kth argument of e_i. We denote the sub-type of e_i as l_i, and the role of $a_{i,k}$ as $r_{i,k}$.

We used the following scores to determine whether an argument can be shared between two different events.

1.2.1 Co-occurrence Ratio of Subtypes

We denote the ratio of co-occurrence of sub-types l_a and l_b as:

$$R_e(l_b|l_a) = \frac{\sum_d \sum_{e_i,e_j \in D_d} \delta(l_i = l_a, l_j = l_b)}{\sum_d \sum_{e_i,e_j \in D_d} \delta(l_i = l_a, l_j = l_a)} \quad (1)$$

which is the ratio of the co-occurrence between two sub-types of events and the co-occurrence of the same sub-type. For example, $\sum_d \sum_{e_i,e_j \in D_d} \delta(l_i = \text{Attact}, l_j = \text{Die}) = 2,808$ means there are 2,808 times that event "Attack" co-occurs with event "Die" in the same document, and $\sum_d \sum_{e_i,e_j \in D_d} \delta(l_i = \text{Attact}, l_j = \text{Attact}) = 11,821$ means there are 11,821 times that event "Attack" co-occurs with event "Attack" in the same document. The ratio determines the probability that another event sub-type can co-occur with a given sub-type.

1.2.2 Ratio of Shared Roles by Events

We denote the ratio of co-occurrence of roles r_a and r_b as:

$$R_r(l_b|l_a) = \frac{\sum_d \sum_{l_i=l_a, l_j=l_b, a_{i,k}, a_{j,p} \in D_d} \delta(a_{i,k} = a_{j,p})}{\sum_d \sum_{l_i=l_j=l_a, a_{i,k}, a_{j,p} \in D_d} \delta(a_{i,k} = a_{j,p})} \tag{2}$$

which is the ratio of the shared arguments between two sub-types of events and the shared arguments of the same sub-type. For example,

$$\sum_d \sum_{l_i=\text{Attact}, l_j=\text{Die}, a_{i,k}, a_{j,p} \in D_d} \delta(a_{i,k} = a_{j,p}) = 505,$$

and

$$\sum_d \sum_{l_i=l_j=\text{Attact}, a_{i,k}, a_{j,p} \in D_d} \delta(a_{i,k} = a_{j,p}) = 4,188.$$

Then we use the harmonic mean of R_e and R_c to filter out event sub-types:

$$R(l_b|l_a) = \frac{2R_e(l_b|l_a)R_r(l_b|l_a)}{R_e(l_b|l_a) + R_r(l_b|l_a)}. \tag{3}$$

The for example for "Attack," we can select three relevant event sub-types: "Die:0.2047," "Transport:0.1759," and "Injure:0.1015."

1.2.3 Argument Co-reference Frequency

Finally we count the following co-occurrence between two arguments:

$$\begin{aligned} & C(l_a, r_a, l_b, r_b) \\ =& \sum_d \sum_{l_i=l_a, l_j=l_b, r_{i,k}=r_a, a_{j,p}=r_b \in D_d} \delta(a_{i,k} = a_{j,p}) \end{aligned} \tag{4}$$

to filter out the arguments. For example, for "Attack@Attacker" the shared roles are as following: "Die@Agent:122," "Transfer-Ownership@Buyer:117," "Transport@Person:95," "Injure@Agent:42," "Charge-Indict@Defendant:25," "Sentence@Defendant:24," "End-Position@Person:22," "Arrest-Jail@Person:20," "Convict@Defendant:18," "Phone-Write@Entity:15."

Given the selected relevant sub-types, we then use following roles for possible transfer of arguments: "Die@Agent:122," "Transport@Person:95," "Injure@Agent:42."

Then for two event mentions, if they have shared a same mention, and there are other mentions satisfying $C(l_a, r_a, l_b, r_b) > \lambda_1$ and $R(l_b|l_a) > \lambda_2$, then we transfer the arguments, where the λ_1 and λ_2 parameters are empirically set.

1.3 EAE Results

1.3.1 Results on Development Data

We randomly selected 40 documents from newswire articles for testing and the rest are split as 90% for training and 10% for validation. We used LibLinear [7] and used the L_2-norm regularized logistic regression as the classifier. From Table 1 we can see that a two step approach (Steps 1.1 and 1.2) results in a better F1 score, while using a one step approach (combination of Steps 1.1 and 1.2) has better precision. Comparing Tables 2 and 3, we can see that using a voting mechanism with different level of semantics can improve the results for the one-step approach, which is a combination of Step 2.1 and 2.2. Moreover, similar to trigger classification, the one step approach will have better precision while two step approach can be tuned to have better F1 scores.

	Combine Steps 1.1 and 1.2		
	P	R	F
Identification + Classification	78.69	54.78	64.60
	Separate Steps 1.1 and 1.2		
	P	R	F
Identification	67.04	65.20	66.11
Identification + Classification	68.53	61.48	64.82

Table 1: Trigger Detection Results on ACE2005.

	Combine Steps 2.1 and 2.2		
	P	R	F
Identification + Classification	63.32	34.65	44.79
	Separate Steps 2.1 and 2.2		
	P	R	F
Identification	56.26	49.49	52.66
Identification + Classification	51.40	43.73	47.26

Table 2: Argument Detection Results without Voting on ACE2005. (Only use 33 multi-class classifiers for identification and classification.)

1.3.2 TAC KBP 2014 Event Track

The evaluation of TAC KBP is based on argument detection, which is the F1 score over all the event arguments associated with event types, argument roles, and realis labels. We present the evaluation results of five trials on the 127 released documents in Table 4. For the five trials, we used the following experimental settings:

- Trial #1: For mention detection, we trained a mention detection model based on ACE 2004/2005 (only on entities except values and time), and we also incorporated named entity recognition results. For trigger detection, we set the identification threshold (for Step 1.1) as 0.3. We use a combination of Steps 2.1 and 2.2 for argument detection.
- Trial #2: For mention detection, we trained a mention detection model based on ACE 2004/2005 (only on entities except values and time), and we also incorporated named entity recognition results. For trigger detection, we set the identification threshold (for Step 1.1) as 0.3. We use separate Steps 2.1 and 2.2 for argument detection. The argument identification thresholds for each event subtypes are determined with best F1 based on validation set.
- Trial #3: For mention detection, we trained a mention detection model based on ACE 2004/2005 (only on entities except values and time), and we also incorporated named entity recognition results. For trigger detection, we set the identification threshold (for Step 1.1) as 0.3. We use separate Steps 2.1 and 2.2 for argument detection. The argument identification thresholds are fixed to 0.1.
- Trial #4: Similar setting as Trial #1. We also incorporate the output of SRL arguments as mention detection results.
- Trial #5: Similar setting as Trial #2. We also incorporate the output of SRL arguments as mention detection results.

From Table 4 we see that incorporating the arguments from SRL, the recall can be improved while precision may be hurt.

	Combine Steps 2.1 and 2.2		
	P	R	F
Identification + Classification	66.29	35.01	45.82
	Separate Steps 2.1 and 2.2		
	P	R	F
Identification	59.14	45.44	51.40
Identification + Classification	54.56	40.74	46.65

Table 3: Argument Detection Results with Voting on ACE2005. (Voting follows Fig. 1.)

	P	R	F
LDC	0.77	0.29	0.42
Rank #1	0.43	0.24	0.31
Trial #1	0.29	0.09	0.14
Trial #2	0.23	0.12	0.16
Trial #3	0.16	0.17	0.18
Trial #4	0.26	0.11	0.15
Trial #5	0.20	0.15	0.17

Table 4: TAC KBP 2014 Event Track Results.)

1.4 EAE Discussion and Conclusions

In this work, we have tried a supervised machine learning based event extraction system. Different from previous stage-wise event extraction systems [1, 2], we introduce a voting mechanism for the final decision. Inspired by the evidence that there are three levels of semantics/topics of events, we designed three different argument detection learning systems. For a final decision, only when two of them agree on an argument, we output that argument. The higher level of topics can get more training labeled data but vaguer definition of argument roles, while lower level of topics will be more specific but with less training data. Using a voting system can balance these different level of semantics.

One major problem of our system is that we focused more on the learning aspects, and less on the mention detection and realis classification. Our mention detection is only trained on the entities of ACE 2004/2005, but ignored values and time. Therefore, we had to incorporate entities from other named entity recognition output and Semantic Role Labeler [15] output directly (not as features for our classifiers). We plan to have more evaluation on the mention detection component in the near future. Moreover, our realis label prediction is relatively simple. We regard the data with "not general" and "past" labels in ACE 2005 as "actual." We haven't checked the pilot data for a validation of the correctness. Moreover, the features used for realis labels are bag-of-words and POS tags. We also need to try richer features for this sub-task.

In summary, we have a supervised learning based system for event argument extraction. The voting mechanism can be a general framework for many information extraction tasks, provided that the semantics/topics of the task can be organized in a hierarchy. We also used a heuristic method to induce world knowledge from ACE 2005 data, with the hope that this knowledge can be used to identify arguments that are shared between events. Although this is not directly evaluated by the task, we are very interested in some further study on this direction for the future work.

2 Illinois Entity Discovery and Linking System

Our system is based on last year's Entity-Linking system [3] with some modifications. The core parts of our system are based on the Illinois Wikifier [16, 4] and a cross document co-reference algorithm which deals with NIL clustering and which refines the outputs of wikifier. We focused on enhancing candidate generation for entity mentions to improve recall.

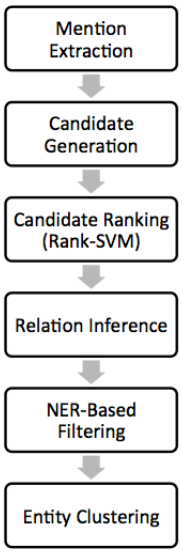

Figure 2: Entity Discovery and Linking pipeline

2.1 EDL System Description

Our system pipeline is depicted in Figure 2. The first four stages enhance corresponding components of the Illinois Wikfier. The key components are:

1. **Mention extraction** identifies mentions in the query document which should be linked to the knowledge base.
2. **Candidate generation** generates a high-recall set of candidate wikipedia titles for each mention.
3. **Candidate ranking** uses a ranking SVM, trained on Wikipedia documents, to rank the candidates for each mention. This uses local features based on the context for each mention under consideration.
4. **Relational inference** re-ranks the top candidates for each mention by considering relations between mentions. The relations are inferred either from text or from a knowledge base (we use DBPedia[3]). We formulate the inference step as an ILP, as described in [4]. The relational inference step captures global interactions between mentions in the same document.
5. **Named Entity Based Filtering** filters out mentions which are linked to wikipedia titles which do not refer to any named entity of interest, viz. PER,GPE,ORG.
6. **Entity Clustering** Clusters the mentions linked by the above stages using the L3M algorithm [19]. Here we don't use wikifier outputs to do clustering.

2.1.1 Key Changes

In this section we discuss some key components which are new or important to this year's task, relative to last year's Entity Linking system.

- Mention Extraction
  We use a high recall mention extraction procedure, which includes all mentions identified by Illinois Named Entity Recognizerand also noun phrases and sub-phrases from Illinois shallow parser [14].

[3] `http://wiki.dbpedia.org/`

- Candidate Generation
  We augmented the candidate generation algorithm in our previous system using the following three approaches. The aim is to improve the recall for candidate generation for downstream ranking, and ensure we do not miss surface forms which are not hyperlinked in Wikipedia.

  1. Freebase-based candidate expansion
     Wikipedia often does not contain all possible nicknames and aliases of named entities. To account for such variations, we query freebase to obtain candidates. For every mention with less than 2 candidates, we query freebase to get possible wikipedia titles it can refer to. For eg., for the nickname "Arnie", freebase will return "Arnold Schwarzenegger" as one possible entity. We add all the candidates for the freebase suggested entity to our candidate set. Namely, all candidate titles for "Arnold Schwarzenegger", will be added to the set. To avoid explosion of the candidate set, we only perform this step for mentions which have less than 2 candidates prior to this stage.
  2. Acronym candidate expansion
     For an acronym mention, we try to find all possible expansions for the acronym in the same document. The candidates of the identified expansion are added into the candidate set of the acronym mention.
  3. Spelling checking
     Many documents originating from discussion forums have spelling mistakes which hurts candidate generation. To remedy this, we correct spelling errors, using the list of Wikipedia titles as our vocabulary. Our system lists corrections ranked by their edit-distance from the surface form in the text. All possible corrections are then added to the candidate set.

- Post-wikification Filtering
  Because we are only asked to output named entities of three types this year, after wikifying every possible mention, we use Illinois NER to identify mentions which have type PER, LOC, or ORG.

- Entity Clustering (Cross Document Co-reference)
  The algorithm is basically the same as the one used in last year's system, except we added word similarity features using Illinois Named Entity Similarity tool (NESim) and Lexical Level Matching (LLM) [6]. Clustering mentions which involve acronyms also takes into account its possible expansion when making the decision to merge clusters. When merging mention clusters, we use the mention with the longest surface form as the representative for the cluster.

2.2 EDL Results

2.2.1 Performance on Development data

Since our Wikifier does not train on the provided training data, we use all training documents as our development set. The performance of our system on the development set and the 2014 EDL test set is as following:

	WikiF1	CEAFm precision	CEAFm recall	CEAFm F1
dev.	36.3	52.6	65.7	58.4
testing	34.9	47.5	62.6	54.0

WikiF1 is called "strong types link match" in the output of evaluation script. It measures the performance of wikifier (no NIL clustering). A mention is considered correct if mention boundaries, entity type, and Wikipedia title are all correct.

2.3 EDL Discussion and Conclusions

One important observation from this task is using an existing NER system is clearly not good enough. We may need to re-train or adapt an NER system to perform well on mention boundary detection and also typing the extracted mentions. One important extension of the basic Named Entity Recognition task is to handle nested mentions. In preparing for this task, we focused on improving our Wikifier on last year's queries. We

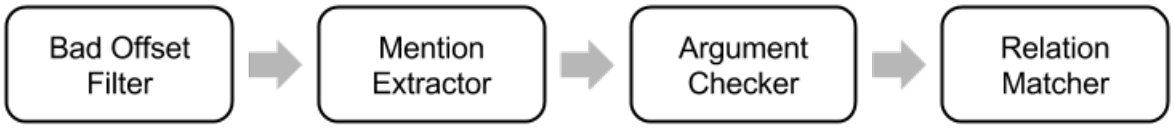

Figure 3: SFV Pipeline

mainly worked on increasing the coverage of candidate generation and improving ranking accuracy. Although we had tried some joint inference between NER and Wikifier, we couldn't improve our NER performance before the task ended. We can see the difference between performance when mentions are given or not from the result of diagnostic evaluation. We got 70.2 on B^3 F1 when the mentions are given. However, when mentions are not given it's only around 50.

3 Illinois Slot Filler Validation System

The UI_CCG SFV team developed two separate systems for the Slot Filler Validation (SFV) task. The first models SFV as an entailment problem [5]: for each query, it finds matches in the relevant document for the query arguments, then determines whether or not the relevant predicate connects them. The second uses agreement between different systems/system runs to identify which queries are expected to represent reliable predictions.

3.1 SFV System Overview

3.1.1 SFV as Entailment

The Illinois SFV Entailment system considers each query independently, and does not use any information provided about the source Slot Filler system that produced the query.

The first stage of the Illinois SFV system is the Bad Offset Filter. (See Figure 3 for a visual description of the system). First, the query's filler offsets are used to verify that a match for the filler value can be found in the document in the neighborhood of the claimed match. Queries for which a good candidate match cannot be found are rejected (labeled as non-entailing). We found that many offsets were reported incorrectly, and were thus a good indicator of the correctness of the query.

Next, the Mention Extractor infers possible matches for arguments in the candidate query's reference document from the outputs of the Illinois NER, Shallow Parser, and Part of Speech [18] taggers. In an ideal world, we could have used the subject and object offsets from the input queries, but we found that they were too noisy to be trusted. For each candidate match, we compared against the input subject and the input filler to decide if it should be a candidate subject match site, or a candidate object match site, respectively. If we believed that the entities being compared were named entities (in this case, names of people or organizations or countries), we used our NESim tool to compare. This left us with two lists: one of subject match sites, and one of object match sites.

We experimented with several different methods of selecting subject/object pairs most likely to hold the answer to the query. One way is to collect all possible subject/object pairs; this results in a relatively large number of pairs, often with nonsensical results, such as pairs which span the entire document. Another approach is to select only those pairs with a short distance between elements. Another insight we had was that the filler is usually provided verbatim from the document text (with the exception of those relations that took dates as arguments; dates are typically normalized). This meant that we could be very strict in matching objects, and then entertain a small number (usually one) of object sites. Then, using the object sites as anchors, we selected only those subject sites that appeared within one sentence on either side of the object. We refer to this final approach as the locality heuristic.

The Argument Matcher then discards candidate pairs with low scores, or which violate a set of relation-specific constraints (for example, a location of death needs to have type "location", not "number" or "person"). This was difficult for those relations which had vague object types, most notably the "title" relation. The remaining pairs are used to identify candidate match sites for the query relation.

Rule	Description
`CONV 4 @@@ OBJ; +SUBJ`	A convention based rule. This matches when the subject directly follows the object. For example, "president John Smith"
`city_of_death 1 h @@@ died; in`	This matches when the relation is `city_of_death` and the keywords "died" and "in" are found in any order between some pair of subject and object (in any order).
`alternate_names 5 h @@@ go; by; +OBJ; anti:.`	This matches when the relation is `alternate_names` and the object directly follows a passage that begins with the subject, contains the word "go", and ends with the word "by". This fails (anti-match) if there is a period in the passage.

Table 5: Some example rules (actual format).

The Relation Matcher uses the candidate match-sites produced earlier in the pipeline to identify candidate text spans to check for groundings of the query relation. Relations are matched based on hand-written patterns encoding lexical terms and argument positions (see Table 5 for examples).

Rule terms are optionally lemmatized to improve generalization; document terms are also optionally lemmatized as needed at comparison time. It is also possible to use more general word similarity techniques (such as Illinois' WNSim [6]). In practice, we did not use this approach as it allowed rules to match too broadly, and made it difficult to analyze errors.

The stricter matching requirement allowed us to take a more principled approach to error analysis because it was much easier to see where the errors were being made. Then, when this was clear, it was possible to write more rules to cover more cases. Our rule-filtering approach (described below) ensures that too many rules is never a problem.

The Rule patterns use a simple grammar that optionally allows argument positions to be fixed relative to some or all rule terms, and disjunctions of lexical terms at a given match location. Text from the document is selected by taking all words enclosed by the match-site, and also a window on either side of the span, the width of which controlled by two parameters. In our experiments, this window was fixed at 12 words. Only those rules corresponding to the relation in question are applied, and if any rule score exceeds a threshold, it is said to have "fired". This is the threshold reported in Tables 6, 7, and 8.

The system architecture reports individual rule performance on a labeled data set, which allows the rules to be filtered to retain the most useful candidates. Additionally, the entailment system has a parameter that can be tuned to allow stricter or more relaxed matching.

3.1.2 Trust-based approach to SFV

UI_CCG's second SFV system is a trust-based system. The intuition here is that knowledge of the source of a filler is a valuable signal regarding that filler's correctness. This is implemented using the simplest possible algorithm: majority voting.

In the task definition, slots are divided into two sets: single-valued slots, which have only a single correct answer, and list-valued slots, which may assume multiple correct answers. We differentiate based on slot type.

The algorithm is as follows. First, fillers are grouped by query+slot. As such, all fillers in a given group can be seen as answering the same question. Then, in each group, we create a histogram of fillers sorted by number of occurrences. Every filler in the group is then labeled either True or False by the following procedure. Every filler is by default False. If the most popular filler occurs more than 5 times, then every occurrence of that filler is set to True. If the second most popular filler occurs more than 10 times, then every occurrence of that filler is set to True.

(In this situation, the first two most popular fillers are allowed to be True even in the case of single-valued slots. In practice, this allows for higher recall.)

This procedure is continued only if the slot is list-valued, including the third most popular filler at a threshold of 11, and the fourth most popular at a threshold of 15. These values were tuned by running the

algorithm on all of the 2013 data.

As a preprocessing step, all fillers which do not have reasonable offsets are immediately marked as False (the first step in Section 3.1.1).

3.1.3 Development Data Used

Both UI_CCG SFV systems were developed using the 2013 TAC Slot Filling and Slot Filler Validation data.

For the entailment-based approach, the rule set from our 2013 SFV system was extended using human analysis of a 4,000-example subset of the 2013 data ("4kdev"). A second 4,000 example set was used to validate this rule set ("4ktest"). We also split the entire 2013 data set into two halves, "25dev" and "25ktest" for final parameter tuning.

Initial system configurations were developed on 4kdev and validated using 4ktest. The rule filtering was tested on the 25kdev and 25ktest data sets, and the rule match threshold was also finalized using these two data sets.

For the 2014 task, the final filtered rule set was generated using statistics from the entire 2013 data set.

3.2 SFV Results

We evaluate our SFV systems on the pooled English Slot Filler system outputs. Based on our entailment-oriented perspective, we assume that the evaluation allows inexact filler matches (but not incorrect filler matches) and that it marks redundant filler values as being correct.

Table 6 lists performance on five data sets of the entailment-based SFV system and a single version of the trust-based system. The entailment-based SFV system is evaluated using the 4,000 example development sets ("4kdev" and "4ktest"), both of which use rules developed using and values tuned on the 4kdev set. This system is also evaluated on our development/test split of the entire 2013 corpus ("25kdev" and "25ktest"), for which the rule set was filtered and the match threshold tuned using the 25kdev set and all other parameters were tuned on the 4kdev data set. Performance is also listed for this system using all 2013 data, where the whole data set was used to generate the filtered rule list, the threshold was that used for the 25ktest evaluation, and all other parameters were tuned on 4kdev. The system performance measured by F1 is fairly robust for a wide range of threshold values, but the trade-off between precision and recall does vary with threshold. Naturally, for lower thresholds, recall is higher while at higher thresholds, recall falls but precision improves. The bias of rule coverage towards cases seen in development data can be seen from the relative performance on the 4kdev and 4ktest sets.

We also present results on all 2013 data for the Trust-based system. The Trust-based approach significantly outperforms the entailment-based system, showing the significant value of the additional information provided by having multiple independent systems make predictions.

System Configuration	Precision	Recall	F1
Baseline – always say 'YES'	0.254	1.000	0.405
Entailment, 4kdev, threshold 0.1	0.523	0.672	0.588
Entailment, 4ktest, threshold 0.1	0.463	0.588	0.518
Entailment, 25kdev, threshold 0.5	0.500	0.619	0.553
Entailment, 25ktest, threshold 0.5	0.481	0.658	0.556
Entailment, all 2013, threshold 0.5	0.495	0.658	0.565
Trust-based, all 2013	**0.570**	**0.688**	**0.624**

Table 6: **Performance on KBP 2013 data.**

UI_CCG submitted 5 runs for the 2014 Slot Filler Validation task. Two different cut-off thresholds were used, with the higher threshold tending to produce a higher precision, lower recall behavior. For each threshold, we ran two versions of the system: one with a filter based on identifying a sufficiently close match to the reported filler value, and the other without that filter.

We report results for the trust-based system and the different configurations of the rule-based SFV system using two versions of the 2014 corpus. Table 7 presents results for a "strict" corpus, which is derived from

the assessment results by labeling only assessments marking the filler as "correct". Table 8 presents results for a "relaxed" corpus, where fillers marked as "inexact" and "redundant" by the assessors are also labeled as "correct".

System Configuration	Precision	Recall	F1
Baseline – always say 'YES'	0.226	1.000	0.369
Threshold 0.5, no filter	0.437	**0.515**	0.472
Threshold 0.5, filter bad offsets	0.457	0.507	**0.481**
Threshold 0.9, no filter	0.451	0.439	0.445
Threshold 0.9, filter bad offsets	**0.468**	0.431	0.449
Trust-based, all 2013	0.467	0.447	0.457

Table 7: **Performance on KBP 2014 data, "strict" corpus.**

System Configuration	Precision	Recall	F1
Baseline – always say 'YES'	0.266	1.000	0.421
Threshold 0.5, no filter	0.503	**0.502**	0.502
Threshold 0.5, filter bad offsets	0.523	0.491	**0.507**
Threshold 0.9, no filter	0.528	0.435	0.477
Threshold 0.9, filter bad offsets	**0.545**	0.425	0.478
Trust-based, all 2013	0.535	0.435	0.480

Table 8: **Performance on KBP 2014 data, "relaxed" corpus.**

3.3 SFV Discussion and Conclusions

From the entailment perspective on the Slot Filler Validation task, the "relaxed" evaluation seems more relevant, as the queries are considered in isolation and with no attempt to match to a reference knowledge base. It is unsurprising, therefore, that the rule-based SFV system performs better on the "relaxed" version of the corpus (although the simple baseline is also stronger for this corpus).

The locality heuristic – requiring a grounding of the subject/relation/ argument triple within a limited span of four sentences centered on the query's filler value – significantly improves system performance. It is appealing for two reasons. First, it allows a precision-first approach to system development, where the errors of the system can be better understood. Improvements can be made by integrating new resources that might be expected to improve recall: for example, we anticipate that adding coreference links to the preprocessing stage will improve the current system. Second, it improves performance by limiting the number of comparisons the system must make when identifying match sites for the query subject and filler value.

The simple trust-based system suffers a much larger drop in performance between the 2013 data and 2014 data. Presumably this is an effect of system diversity. However, the strong performance on the 2013 data compared to the entailment-based system suggests that there is significant potential in combining signals from multiple systems.

References

[1] David Ahn. The stages of event extraction. In *Workshop on Annotating and Reasoning About Time and Events*, pages 1–8, 2006.

[2] Chen Chen and Vincent Ng. Joint modeling for chinese event extraction with rich linguistic features. In *COLING*, pages 529–544, 2012.

[3] X. Cheng, B. Chen, R. Samdani, K. Chang, Z. Fei, M. Sammons, J. Wieting, S. Roy, C. Wang, and D. Roth. Illinois cognitive computation group ui-ccg tac 2013 entity linking and slot filler validation systems. In *TAC*, 2013.

[4] X. Cheng and D. Roth. Relational inference for wikification. In *EMNLP*, 2013.

[5] I. Dagan, D. Roth, M. Sammons, and F. Zanzotto. *Recognizing Textual Entailment: Models and Applications*. Morgan and Claypool, 7 2013.

[6] Q. Do, D. Roth, M. Sammons, Y. Tu, and V. Vydiswaran. Robust, light-weight approaches to compute lexical similarity. Technical report, Computer Science Department, University of Illinois, 2009.

[7] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 2008.

[8] J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL*, 2005.

[9] Evgeniy Gabrilovich and Shaul Markovitch. Feature generation for text categorization using world knowledge. In *IJCAI*, pages 1048–1053, 2005.

[10] Evgeniy Gabrilovich and Shaul Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJCAI*, 2007.

[11] D. Klein and C. D. Manning. Fast exact inference with a factored model for natural language parsing. In *NIPS*, 2003.

[12] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119. 2013.

[13] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751, 2013.

[14] V. Punyakanok and D. Roth. The use of classifiers in sequential inference. In *NIPS*, 2001.

[15] V. Punyakanok, D. Roth, and W. Yih. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 2008.

[16] L. Ratinov, D. Downey, M. Anderson, and D. Roth. Local and global algorithms for disambiguation to wikipedia. In *ACL*, 2011.

[17] L. Ratinov and D. Roth. Design challenges and misconceptions in named entity recognition. In *Proc. of the Annual Conference on Computational Natural Language Learning (CoNLL)*, 2009.

[18] D. Roth and D. Zelenko. Part of speech tagging using a network of linear separators. In *COLING-ACL, The 17th International Conference on Computational Linguistics*, 1998.

[19] R. Samdani, K. Chang, and D. Roth. A discriminative latent variable model for online clustering. In *ICML*, 2014.