Illinois CCG TAC 2015 Event Nugget, Entity Discovery and Linking, and Slot Filler Validation Systems

Mark Sammons and **Haoruo Peng** and **Yangqiu Song** and **Shyam Upadhyay** and **Chen-Tse Tsai** and **Pavankumar Reddy** and **Subhro Roy** and **Dan Roth**

Department of Computer Science, University of Illinois, Urbana-Champaign.
{hpeng7,yqsong,upadhya3,ctsai12,mssammon,muddire2,sroy9,danr}
@illinois.edu

Abstract

This paper describes University of Illinois's Cognitive Computation Group (UI-CCG)'s submissions for three TAC tracks: Event Nugget Detection/Co-reference; Entity Discovery and Linking (EDL); and Slot Filler Validation (SFV).

The Event Nugget Detection and Co-reference system employs a supervised model for event nugget detection with rich lexical and semantic features while we experiment with both supervised and unsupervised event co-reference methods. We also utilize ACE2005 data as an additional source and use several domain adaptation techniques to improve our system's performance.

The Entity Discovery and Linking system focuses on solving the Spanish subtask. The system uses Google Translation to translate Spanish documents into English and then apply Illinois Wikifier to identify entity mentions and disambiguate them to Wikipedia entries. It outperforms other participants on both linking and clusteringevaluations.

The Illinois SFV system treats the task as an entailment problem, seeking to identify for each individual query whether or not the proposed answer is valid based on the information contained in the query document. The system builds on those of previous years, and uses a machine learning component to try to extract cues from unmarked relations in the context of the query relation.

The three systems described here were developed as separate systems.

1 Event Nugget Detection and Co-reference

In this section, we describe our submission to the TAC KBP event task. Our team participated in the TAC KBP Event Nugget (EN) track. It includes three sub-tasks: event nugget detection, event co-reference based on gold and predicted event nuggets. Our system uses a supervised model for event nugget detection with rich lexical and semantic features. For event co-reference model, we experiment with both supervised and unsupervised methods. For supervised models, we train a classifier to model the similarity between each event nugget pair while we also ESA representations (Gabrilovich and Markovitch, 2007; Song and Roth, 2015) to compute this similarity in an unsupervised fashion.

We show each module of our system in the following sub-sections and discuss several techniques that we employ.

1.1 Event Nugget Detection

We use a stage wise classification approach to extract all events (Ahn, 2006; Chen and Ng, 2012). We first train a 34-class classifier (33 event subtypes and one non-event class) to detect event nuggets and classify them into different types. We apply it on each token. Features for this supervised classifier includes lexical features, features from parser, *Named Entity Recognition* (NER), *Semantic Role Labelling* (SRL), entity co-reference and WordNet, and other semantic features from *Explicit Semantic Analysis* (ESA) (Gabrilovich and Markovitch, 2005; Gabrilovich and Markovitch, 2007) and Brown Clusters (Brown et al., 1992). We then apply a classifer using the same set of rich features on each detected event nuggets to get *REALIS* information (ACTUAL, GENERIC or OTHER).

Features They can be summarized in the following categories:

1. Lexical features: context (part-of-speech tag and lemma) of a candidate token in a window size of 5 and 20, plus their conjunctions.

2. Seed features: we use 140 seeds for event triggers following a previous work (Bronstein et al., 2015). We consider whether a candidate token is a seed or not (also its type if it matches) and conjunction of the matched seed and context seeds (also their types).

3. Parse Tree features: path from a candidate token to root, number of its right/left siblings and their categories, and paths connecting a candidate token with other seeds or named entities.

4. NER features: named entities and their types within a window size of 20 of a candidate token.

5. SRL features: whether a candidate token is VerbSRL/NomSRL predicate and its role, its conjunction with SRL relation names and the conjunction of the SRL relation name and the NER types in the context.

6. Coref features: co-referred entities with the candidate token, and their conjunction with both the candidate token and named entities in the context.

7. ESA features: top 50 ESA concepts for each candidate token.

8. Brown cluster features: brown cluster vector of prefix length 4, 6, 10 and 20.

9. Wordnet features: hypernym, hyponym, entailment words and derived words of both the candidate token and its context, and also the wordnet relations between the candidate token and seed words.

10. Other features: whether a candidate token is in Framenet/Propbank or is a deverbal noun.

Learning Model We choose *Support Vector Machine* (SVM) to train both event nugget detection classifier and realis classifier. We use L2 loss and set C as 0.1 after tuning on a development set. We use Illinois NLP packages for NER[1], SRL[2], and Entity Co-reference[3].

Domain Adaptation Apart from the KBP training data, we use ACE2005 as an additional source of our training data. The ACE event taxonomy is similar to that of the KBP task. To enable the domain adaptation from ACE to KBP, we employ the following techniques.

1. We view event triggers in ACE annotations as event nuggets in the KBP task.

2. We apply a deterministic rule to convert ACE realis information to KBP formulation. Specifically, we combine "Genericity.Past" and "Tense.Past" in ACE to be "Actual" in KBP. We also use"Genericity.Generic" directly as "Generic" and "Tense.Unspecified" (and sometimes also "Tense.Future") as "Others".

3. As ACE and KBP have different data distributions based on event types, we use resampling in ACE to match the event nugget type distribution in KBP. There is also notable mismatch of the density of events between ACE and KBP. On average, each sentence contains 0.34 events in ACE, while in KBP, the statistics is 0.82, which is significantly larger. Thus, we also use subsampling to get a subset of the negative training examples in ACE to have a similar positive and negative training example ratio in KBP.

Results We have two development datasets, one from ACE2005 dataset while the other is from KBP data. On ACE2005, we select 40 documents from newswire articles for testing and the rest for training. We only use this *ACE development set* to evaluate our performance on ACE. For KBP data, we also select 30 documents (20% of the available data) as the developmemt set. These selected documents contain genres of both news articles and discussion forums. We use this *KBP development set* to test performance on models trained on KBP data and ACE-KBP combined data using domain adaptation techniques. Results on the two development sets are shown in Table 1. The overall score on the KBP development set shows that it is best to train on ACE-KBP combined data without doing resampling and subsampling techniques. However, the sampling technique improves recall

[1] http://cogcomp.cs.illinois.edu/page/software_view/NETagger

[2] http://cogcomp.cs.illinois.edu/page/software_view/SRL

[3] http://cogcomp.cs.illinois.edu/page/software_view/Coref

on detection, which may have a bigger impact on test data.

For the KBP event nugget detection task, we submit the following three runs.

1. Trial One: Models trained on all KBP data

2. Trial Two: Models trained on all ACE-KBP combined data without doing resampling and subsampling.

3. Trial Three: Models trained on all ACE-KBP combined data with doing resampling and subsampling.

Results are shown in Table 2. The overall score on the KBP test set shows that it is best to train on ACE-KBP combined data with doing resampling and subsampling techniques.

Table 1: Event Nugget Detection Results on Development Sets.

	Precision	Recall	micro-F1
ACE-dev (trained on ACE)			
Detection	76.53	71.64	74.00
Type	71.22	66.59	68.83
Realis	75.24	70.41	72.74
Type+Realis	69.99	65.43	67.63
KBP-dev (trained on KBP)			
Detection	72.50	50.73	59.69
Type	51.46	36.01	42.37
Realis	44.11	30.87	36.32
Type+Realis	33.89	23.72	27.91
KBP-dev (trained on ACE-KBP w/o sampling)			
Detection	74.80	49.89	59.86
Type	54.25	36.18	43.41
Realis	48.30	32.21	38.65
Type+Realis	**36.82**	**24.56**	**29.47**
KBP-dev (trained on ACE-KBP w sampling)			
Detection	74.08	**51.28**	**60.60**
Type	52.86	36.59	43.24
Realis	45.48	31.48	37.21
Type+Realis	35.68	24.70	29.19

1.2 Event Co-reference

We employ a event-pair model for event co-reference, which is similar to the mention-pair co-reference model (Denis and Baldridge, 2007; Bengtson and Roth, 2008) in entity coreference. We model the similarity between event nugget pairs either by a supervised model or in an unsupervied fashion. We then employ a greedy clustering method to put every event nugget into co-reference chains. We first make a decision on each event nugget pair (whether they are linked or not) and then put all linked event nuggets into the same event co-reference chain. We now discuss the details of our system.

Table 2: Event Nugget Detection Results on the KBP Test Set.

	Precision	Recall	micro-F1
Trial One			
Detection	84.49	43.44	57.38
Type	70.32	36.16	47.76
Realis	58.87	30.27	39.98
Type+Realis	**50.16**	25.79	34.07
Trial Two			
Detection	83.52	45.03	58.51
Type	69.29	37.36	48.54
Realis	58.56	31.57	41.03
Type+Realis	49.45	26.66	34.64
Trial Three			
Detection	82.22	46.99	59.80
Type	67.95	38.83	49.42
Realis	58.94	33.68	42.87
Type+Realis	49.88	**28.50**	**36.28**

Supervised Method - Features They can be put into the folliwing categories:

1. Event Nugget Features: all features for event nugget detection (defined in Section 1.1) and their conjuntions between two events nuggets.

2. Event Argument Features: we get event arguments directly through SRL. We first extract the sentence containing an event nugget, and then use SRL to extract SRL arguments (specfically A1 and A2). We treat them as event arguments. Though the event arguments we get are not precise, they are sufficient for event co-reference (as later supported by our results). We apply all event nugget detection features on the arguments (defined in Section 1.1) and their conjuntions between arguments of two events nuggets.

3. Event Entity Features: we get event entities directly through entity co-reference. We run entity co-reference on the whole document.

Then, similar to the construction of event argument features, we extract sentences containing event nuggets, and then use entity mentions (as annotated by co-reference) in these sentences as event entities. We apply all event nugget detection features on the entities (defined in Section 1.1) and their conjuntions between entities of two events nuggets.

4. Pair-wise Features: distance, ESA similarities of two events nuggets and number of co-referent entity mentions of two events nuggets.

Supervised Method - Learning Model We train our supervised event-pair model using SVM. We choose L2 loss and set C as 1 after tuning on a development set. We use Illinois NLP packages for SRL and Entity Co-reference (the same pakcages explained in Section 1.1).

Supervised Method - Domain Adaptation Similar to event nugget detection, we use ACE2005 as an additional source of our training data. To enable the domain adaptation from ACE to KBP, we employ the following techniques.

1. As ACE has gold entity co-reference annotations, we use them during training on ACE data. However, we use Illinois Entity Co-reference package to get co-reference annotations on KBP data.

2. There is a notable mismatch of the length of event co-reference chains between ACE and KBP. On average, each event co-reference chain contains 1.30 events in ACE, while in KBP, the statistics is 1.75, which is much larger. We use subsampling to get a subset of the negative training event nugget pairs in ACE to have a similar positive and negative training example ratio in KBP.

Unsupervised Method We find that we can actually model the similarity between event nugget pairs in an unsupervised fashion. We first get the ESA vector representation for the event nugget, event arguments and entity mentions within the same sentence of the nugget. We then use simple concatenation of these vectors to represent each event nugget. We can then directly model the similarity of two events as the cosine similarity between their ESA representations.

Results We evaluate on the KBP development set explained in Section 1.1 as well as the final KBP test set. For the KBP event nugget co-reference task, we submit the following three runs.

1. Trial One: Models trained on KBP data

2. Trial Two: Models trained on ACE-KBP combined data without doing subsampling.

3. Trial Three: Models trained on ACE-KBP combined data with doing subsampling.

Results on the two development sets are shown in Table 3 while results on the final test set is shown in Table 4. In both tables, "AVG" stands for *CoNLL Average*, which is the average score of MUC, B^3, $CEAF_e$, $CEAF_m$ and BLANC. Though the performance of the unsupervised method is lower than supervised methods, it is competitive considering its simplicity. On the KBP test set, our best results rank 2nd in event co-reference task based on gold events while rank 3rd based on predicted events.

2 Entity Discovery and Linking

In this section, we describe our submission to the Tri-lingual Entity Discovery and Linking task. We participated in the Spanish sub-task where the system is required to identify named entity mentions in the given Spanish documents and also disambiguate them to the entities in FreeBase. Our translation based approach achieves the best linking and clustering scores on the final evaluation dataset. Figure 1 shows a schematic view of our pipeline.

2.1 System Pipeline

Our system is based on Google Translation and Illinois Wikifier (Ratinov et al., 2011; Cheng and Roth, 2013) along with some heuristic algorithms to align the English mentions back to the Spanish documents. The details of the pipeline are as follows:

1. **Translation** We use Google Translation to translate the Spanish documents in to English, which allows us to leverage the well-developed English Wikifier to discover and disambiguate entity mentions in the documents.

2. **Wikification** The Illinois Wikifier uses the Illinois Named Entity Tagger (Ratinov and Roth, 2009) to extract the named entity mentions in the documents and then try to link

Table 3: Event Nugget Coreference Results on the KBP Development Set.

	MUC	B^3	$CEAF_e$	$CEAF_m$	BLANC	AVG
Gold Event Nuggets						
Trial One	67.39	74.55	59.97	60.51	67.37	65.96
Trial Two	66.48	75.75	60.99	63.25	70.37	67.37
Trial Three	69.08	76.54	61.49	63.77	71.10	**68.40**
Unsupervised	64.25	73.04	60.16	61.11	69.81	65.67
Predicted Event Nuggets						
Trial One	43.66	53.86	44.57	46.41	34.51	44.60
Trial Two	39.61	48.97	50.48	47.63	39.61	49.04
Trial Three	47.73	58.73	48.96	50.48	40.43	**49.19**
Unsupervised	38.63	52.03	43.78	45.90	32.64	42.60

Table 4: Event Nugget Coreference Results on the KBP Test Set.

	MUC	B^3	$CEAF_e$	$CEAF_m$	BLANC	AVG
Gold Event Nuggets						
Trial One	52.23	72.80	64.47	57.53	50.68	59.54
Trial Two	52.82	69.91	61.78	54.46	48.38	57.47
Trial Three	63.78	83.75	75.81	74.08	73.99	**74.28**
Predicted Event Nuggets						
Trial One	52.41	67.88	59.55	61.01	55.07	59.18
Trial Two	52.78	68.31	60.48	61.60	55.90	59.81
Trial Three	53.29	68.80	60.78	61.86	56.92	**60.33**

each mention to a title in English Wikipedia. In addition, we try to modify the titles of the PERSON mention after wikification, since a shorter mention may have higher ambiguity. For example, if "Chris" and "Chris Kyle" are two mentions in a document, the entity which "Chris Kyle" links to is more likely to be the correct one since "Chris" is very ambiguous to Wikifier. Under the assumption that these two mentions refer to the same person, we modify the title of "Chris" to "Chris Kyle"'s title.

3. **Spanish Mention Discovering** Google Translation does not provide word/phrase alignments of the source and target languages. This poses a challenge of mapping the English mentions back to the Spanish documents. We use the following algorithm to align mentions:

   - Given a mention in English, we produce possible Spanish translations by the English mention surface, the Google translated English string, the Spanish Wikipedia title (through the language link in Wikipedia if the mention is linked to an English Wikipedia title), and all the strings which are redirected to the Spanish title.
   - For each n-grams of the Spanish document, we calculate the edit distance to the possible Spanish translations obtained in the previous step and then return the first n-gram which has not been mapped to any English text and has the lowest edit distance at most 1.

   For example, if the English mention is "USA", by the first step of the algorithm, we generate the following lowercased Spanish translations: "usa" (the English surface form), "ee.uu" (the translation of "usa"), "estados unidos" (the Spanish Wikipedia title), "ee.uu.", "eeuu", "e.u.a.", "estadounidense", "ee. uu.", "ee uu", "unión americana", "eua", "estadounidenses", and "united states of america" (strings redirected to the Spanish Wikipedia title). We then match all the above possible translations with n-grams in the Spanish document by the second step. We

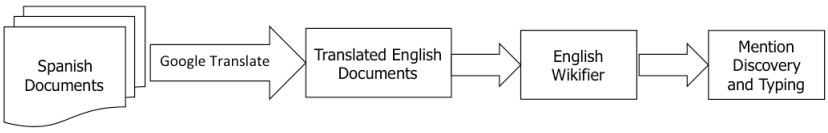

Figure 1: System Pipeline for EDL Task

can see that Wikipedia provides many useful information at this stage.

4. **FreeBase ID Mapping** In principle, the mapping between Wikipedia titles to the corresponding FreeBase MIDs is deterministic. However, due to the difference of versions used in the Wikifier and the FreeBase, we could not find the corresponding FreeBase MIDs for some Wikipedia titles. We apply the following two-step approach to find the FreeBase MID. First, we directly query the FreeBase MID of a given Wikipedia title using FreeBase's MQL API. If the query fails to find a FreeBase entry, we query the FreeBase search API by the Wikipedia title and use the first returned FreeBase entity as the answer.

5. **Mention Filtering** Since only five types of name entities are of the interest this year, we train a binary classifier based on the FreeBase types to remove the mentions which do not belong to these five types. When constructing training examples, we use the gold annotations of the English and Spanish training documents as positive examples, whereas the negative examples are generated from our system predictions which do not overlap with any gold mentions.

6. **Named Entity Typing** The Illinois Named Entity Recognizer only covers three entity types. We train a 5-class classifier based on the FreeBase types to classify each mention into one of the five entity types. For the NIL mentions, since we could not obtain FreeBase types for them, we simply use the types provided by the Named Entity Recognizer.

7. **Entity Clustering** As the number of NIL mentions are fewer in this year's documents, we simply cluster all mentions in a document by the Jaccard similarity of the surface strings.

Approach	Precision	Recall	F1
Development documents			
Stanford NER	69.4	68.4	68.9
FreeLing NER	72.2	71.5	71.8
Illinois CCG	**81.4**	**76.6**	**79.0**
Evaluation documents			
Stanford NER	61.6	61.9	61.8
FreeLing NER	58.7	65.0	61.7
Illinois CCG	**73.6**	**70.9**	**72.2**

Table 5: A comparison of mention extraction performance. We compare two off-the-shelf Spanish NER packages with our system. We only evaluate the mention spans since the entity types used in these two NERs are different from our task. Note that the results of our system is after the third step of the pipeline (Spanish Mention Discovering).

Measure	Prec.	Recall	F1
Development documents			
mention spans	85.5	79.6	82.4
mention spans+type	79.8	74.3	76.9
mention spans+link	78.5	73.1	75.5
mention CEAF	81.6	76.1	78.8
Evaluation documents			
mention spans	80.1	77.2	78.7
mention spans+type	76.1	73.4	74.7
mention spans+link	72.5	69.9	71.2
mention CEAF	75.0	72.3	73.7

Table 6: Performance of UI_CCG Spanish entity linking system on the development documents and the final evaluation documents. We got the first place on linking (mention spans+link) and clustering (mention CEAF) on the evaluation documents.

2.2 Results

In the following results, "development documents" refers to all the Spanish training documents this year. We observe the performance on these documents to guide the development of algorithms. All the training of Wikifier ranking model, entity type classifier, and the filtering

classifier are done on the English documents of this and last year's EDL tasks. The final held-out evaluation documents are referred to "evaluation documents" in all the results.

Comparing Mention Extraction Mention extraction is a very important step since it is usually the first step of a pipeline thus sets the ceiling of the final performance. An immediate baseline of mention extraction is by applying a Spanish Named Entity Recognizer (NER). In Table 5, we compare our system with two off-the-shelf NERs: Stanford[4] and FreeLing[5]. For our system, we evaluate the results after the third step of the pipeline (Spanish Mention Discovery), so no mention filtering is applied. It is basically mapping the results of English NER (used by Wikifier) on the translated documents back to the Spanish documents by the algorithm described in the third step of Section 2.1. Note that in this evaluation, we exclude the author mentions within XML tags since these mentions can be easily extracted by regular expressions and we only input plain text (without XML tags) to the two NERs. In addition, only mention spans are evaluated as the entity types output by NERs are different from the types used in the shared task.

We can see that the proposed UI_CCG system outperforms both NERs significantly on both development and evaluation documents. From some error analysis we see that Spanish NERs perform poorly if the text is in bad format, for example, some paragraphs are all uppercased or all lowercased from the discussion forum. On the other hand, Google Translation tries to format the text better, therefore makes the text easier for the English NER.

End-to-end Evaluation The overall performance of our system is summarized in Table 6. We can see that the performance of Wikifier is around 90 (the ratio between mention spans+link to mention spans), which indicates the mentions are not hard in this year's documents. For the final evaluation, our system performs the best among all the participants in terms of the linking performance (mention spans+link) and clustering performance (mention CEAF).

[4] http://nlp.stanford.edu/software/CRF-NER.shtml

[5] http://nlp.lsi.upc.edu/freeling/index.php

3 Slot Filler Validation

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

3.1 The 2015 SFV Task

The 2015 TAC SFV task is significantly different from the previous years as the queries are generated from a Cold Start Knowledge Base Population task rather than a Slot Filler task. This adds a number of explicit inverse relations to the previous Slot Filler/SFV task, as well as a new entity type (Geopolitical Entity, or GPE). The corpus of source documents is much smaller ($50,000$ documents), and much more skewed towards forum documents (about 80%). It is also very much larger than 2013 and 2014 data: 1.2M Slot Filler System outputs compared to 50K each in 2013 and 2014. This larger scale adds significant complications, making a straightforward reliance on more sophisticated NLP tools problematic. In addition, queries were split into one- and two-hop queries, with the two-hop queries being treated as a single step (i.e. if the second hop is correctly extracted but based on an incorrect first hop extraction, it is judged incorrect).

3.2 Rule-Based System

The 2015 Illinois SFV system was based on the system from the TAC 2014 SFV track (Sammons et al., 2014). In order to manage the much larger corpus, the system processed the query documents with lighter-weight NLP tools (Illinois POS tagger (Roth and Zelenko, 1998), Shallow Parser (Punyakanok and Roth, 2001; Punyakanok and Roth, 2005), Named Entity Recognizer (Ratinov and Roth, 2009), and Illinois Temporal Expression Extractor (Zhao et al., 2012)).

The first stage of the Illinois rule-based SFV system is the Argument Checker. After identifying candidate subject and filler matches in the source document close to the specified filler provenance, the query is rejected if good matches for the subject and filler are not found. For each candidate matching argument pair for the query subject and object, the system, applies a set of handwritten rules appropriate for the query relation. If no match is found, the query is rejected. For this year's task we used a union of the rules identified as useful for the 2013 and 2014 SFV corpora. To

attempt to answer queries involving the new inverse relations, we copied rules from the corresponding original relations and where they were sensitive to the relative position of the arguments, modified them accordingly.

The SFV system also identifies other entities/values in the neighborhood of the query match that have the same types as the query arguments. It then applies rules for the query relation and a set of relations identified as "competing" with that relation to identify relations that may hold for these other entities. The resulting relation match information is passed on to the system decision component.

3.3 Machine Learning component

The final stage of the SFV system uses a machine learning approach. Using all of the 2013 and half of the 2014 SFV corpora as training data, we identified coarse features that could potentially improve SFV performance. These included whether or not competing relations were identified that also covered query arguments, and which (if true) might preclude the query relation. We tried several different machine learning algorithms, and mainly used Naive Bayes for its efficiency. One run uses Linear SVM.

We developed four separate data sets: for the 2013 and 2014 SFV data, we split each into two halves, ensuring there were no tuples in the first half that appeared in the second half. We then used one half of each year as a training/development set, and the other half for testing. We conducted some experiments where we trained on all 2013 data plus the 2014 training data, and evaluated on the 2014 data. We used the models trained this way for the SFV system machine learning component when predicting labels for the 2015 data.

We used the preliminary assessment results provided by NIST to determine basic choices about the final system configuration: whether or not to try to filter queries using the new inverse relations, and whether or not to try to change round 2 queries based on round 1 predictions.

3.4 Results

Table 7 shows results on the training and development data we used for this task. The results for the basic SFV system (rule-based system without learned component) are comparable to last year's system performance, and the learning component gives a significant boost to performance. Results for the rule-based system on the 2014 data are slightly lower than last year, presumably because the 2015 system does not use the Illinois Wikifier (Cheng and Roth, 2013) in order to speed up processing time. When the learning component is added, performance approaches that of the 2014 system.

Table 8 shows results using the preliminary assessments on several different system configurations for round one SFV queries, while Table 9 shows the performance for round two SFV queries. The results show that the SFV system is too strict on round one queries, as the original system performance is quite high. For round two queries, the SFV system does better, even though it is relying on the round one predictions to filter round two queries that would otherwise have been labeled as entailing.

System Configuration	Precision	Recall	F1
Baseline 2013 test – always say 'YES'	0.300	1.000	0.462
Basic SFV System 2013	0.499	0.623	0.554
Learned System (Naive Bayes) 2013	0.487	0.753	0.591
Baseline 2014 test – always say 'YES'	0.295	1.000	0.455
Basic SFV System 2014	0.445	0.542	0.489
Learned System 2014	0.427	0.610	0.503

Table 7: **Performance on 2013 and 2014 SFV Test data** For 2014 Test, learned model is trained on all of 2013 and half of 2014.

System Configuration	Precision	Recall	F1
Baseline – always say 'YES'	0.335	1.000	0.502
Basic SFV System	0.328	0.693	0.445
Learned System, Naive Bayes	0.335	0.607	0.401
Learned System, SVM	0.337	0.624	0.438

Table 8: **Performance on preliminary Round One assessed 2015 data**

System Configuration	Precision	Recall	F1
Baseline – always say 'YES'	0.251	1.000	0.401
Basic SFV System	0.420	0.260	0.323
Learned System, Naive Bayes	0.674	0.444	0.536
Learned System, SVM	0.606	0.396	0.479

Table 9: **Performance on preliminary Round Two assessed 2015 data**

3.5 Discussion

While the system does relatively well on 2013 and 2014 data sets, performance on the 2015 data does not improve over the original cold start systems. This is in part because these systems are performing better on round one queries than on comparable Slot Filler queries than in previous years.

Error analysis indicates some shortcomings of the Illinois SFV system rule sets. For example:

Query: Poland gpe:residents-of-country Stefan Niesiolowski
Text: Stefan Niesiolowski, who is a deputy Speaker of the Polish parliament, told a TV audience in June that same-sex families are abnormal and described lesbian couples with children as a "serious pathology."

To capture this kind of example reliably, the rule-based approach requires world knowledge that people who hold positions in certain kinds of national organizations are typically residents of that country.

Other errors indicate that the rules are not specific enough, as they do not account for syntactic structure:

Query: George Zimmerman per:spouse Trayvon Martin
Text: Shellie Zimmerman, wife of George Zimmerman, charged with murdering Trayvon Martin, was arrested Tuesday on one count of perjury, the Seminole County, Fla.

Here, the system matches the pattern "wife of [SUBJECT]" and a good match for the query filler is found close by, but would ideally have syntactic constraints that would prevent the rule from firing when the context does not support the inference.

Acknowledgments

This work was supported by the US Defense Advanced Research Projects Agency (DARPA) under the DEFT and LORELEI programs. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

References

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

E. Bengtson and D. Roth. 2008. Understanding the value of features for coreference resolution. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 10.

Ofer Bronstein, Ido Dagan, Qi Li, Heng Ji, and Anette Frank. 2015. Seed-based event trigger labeling: How far can event descriptions get us? *Volume 2: Short Papers*, page 372.

Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jenifer C. Lai, and Robert L. Mercer. 1992. Class-Based n-gram Models of Natural Language. *Computational Linguistics*, 18(4):467–479.

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

X. Cheng and D. Roth. 2013. Relational inference for wikification. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

P. Denis and J. Baldridge. 2007. Joint determination of anaphoricity and coreference resolution using integer programming. In *Proceedings of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*.

Evgeniy Gabrilovich and Shaul Markovitch. 2005. Feature generation for text categorization using world knowledge. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1048–1053.

Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

V. Punyakanok and D. Roth. 2001. The use of classifiers in sequential inference. In *Proc. of the Conference on Neural Information Processing Systems (NIPS)*, pages 995–1001. MIT Press.

V. Punyakanok and D. Roth. 2005. Inference with classifiers: The phrase identification problem. Technical report.

L. Ratinov and D. Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proc. of the Conference on Computational Natural Language Learning (CoNLL)*, 6.

L. Ratinov, D. Roth, D. Downey, and M. Anderson. 2011. Local and global algorithms for disambiguation to wikipedia. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

D. Roth and D. Zelenko. 1998. Part of speech tagging using a network of linear separators. In *Coling-Acl, The 17th International Conference on Computational Linguistics*, pages 1136–1142.

M. Sammons, Y. Song, R. Wang, G. Kundu, C.-T. Tsai, S. Upadhyay, S. Mayhew, D. Roth, and S. Ancha. 2014. Overview of ui-ccg systems for event argument extraction, entity discovery and linking, and slot filler validation. In *Proc. of the Text Analysis Conference (TAC)*.

Y. Song and D. Roth. 2015. Unsupervised sparse vector densification for short text similarity. In *Proc. of the Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 5.

R. Zhao, Q. Do, and D. Roth. 2012. A robust shallow temporal reasoning system. In *Proc. of the Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL) Demo*, 6.