Illinois Cognitive Computation Group UI-CCG TAC 2013 Entity Linking and Slot Filler Validation Systems

Xiao Cheng, Bingling Chen, Rajhans Samdani,
Kai-Wei Chang, Zhiye Fei, Mark Sammons,
John Wieting, Subhro Roy, Chizheng Wang, and **Dan Roth**
Department of Computer Science, University of Illinois at Urbana-Champaign
Urbana, IL 61801
{cheng88,chen386,rsamdan2,kchang10,zfei2,mssammon,
wieting2,sroy9,cwang86,danr}
@illinois.edu

Abstract

In this paper, we describe the University of Illinois (UI_CCG) submission to the 2013 TAC KBP English Entity Linking (EL) and Slot Filler Validation (SFV) tasks. We developed two separate systems. Our Entity Linking system integrates an improved version of the Illinois Wikifier with additional functionality to identify and cluster entity mentions that do not correspond to entries in the reference knowledge base. 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 Entity Linking System

The goal of the TAC KBP English Entity Linking (EL) task is to cluster name entity mentions in a document and either link them to entities in a knowledge base (KB), or assign them to a non-KB entry (NIL) with a unique NIL ID.

Our Entity Linking system has two main components: (1) the Wikifier[1] (Cheng and Roth, 2013; Ratinov et al., 2011) as the underlying knowledge base linking engine, linking all mentions in a given text to the entire Wikipedia (a superset of the TAC KBP Knowledge Base); and (2) a cross-document coreference resolution system based on the Best Latent Left-Linking (L^3M) approach (Samdani et al., 2012; Chang et al., 2013). During testing, the system runs the linking and clustering components separately and then combines the results.

[1] http://cogcomp.cs.illinois.edu/page/software_view/33

It is important to note that our Wikifier component is not retrained on TAC data and our L^3M clustering is trained and tuned on the 2012 Entity Linking queries, optimizing for the B^3 F1 metric.

1.1 EL System Description

The UI_CCG EL system first preprocesses the query mentions and documents. Then it applies the Wikifier to link the mentions to Wikipedia entities and applies cross-document coreference resolution system to cluster mentions into groups. The final decision is provided by a voting scheme based on the clustering and wikification results. Figure 1 shows the overall system architecture. We next describe each stage in detail.

1.1.1 Preprocessing

We note that due to the excessive amount of time needed to exhaustively annotate the referenced documents, we purposefully transform the input document according to the query. Specifically, in the preprocessing stage we truncate the longer documents and only keep, for each long document, the starting paragraph, the paragraphs surrounding query mentions and, if length permits, the sentences that contain possible coreference mentions of the query mention. The goal is to keep the text short while minimizing the loss of critical context needed to make the linking decision.

The linking system consists of 3 stages: Candidate Generation, Ranking, and Clustering. The first two stages are accomplished using the Wikifier and the last stage uses the L^3M component.

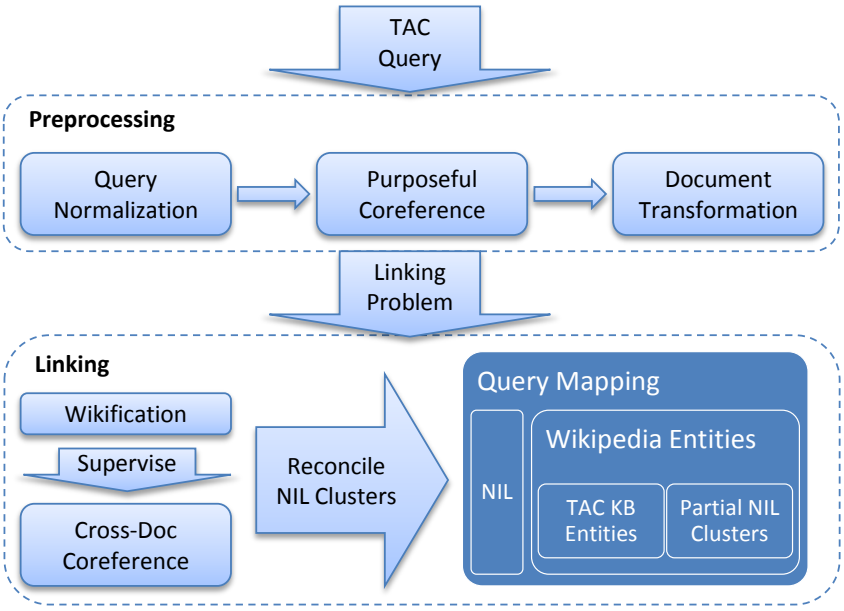

Figure 1: Overall UI_CCG Entity Linking system architecture

1.1.2 Wikification

We run the Wikifier[2] to get a list of confident Wikipedia links for each query and the associated confidence scores. The Wikifier performs the following steps.

Candidate Generation We first generate candidates for each mention by taking the top 20 Wikipedia pages that were linked based on the normalized mention surface. We also search lexically in the arguments of DBPedia infobox relations and other candidates in the same document when certain textual relations are found (Cheng and Roth, 2013). In addition to the query mention, we also generate candidates for each NER mention and for noun phrase sub-chunks of length up to 5. Furthermore, we consider as mentions those phrases that are comprised of two such mentions when connected by prepositions, conjunctions and punctuation marks. These strings are matched exactly against all anchor text links in Wikipedia, against page titles and redirect page titles.

[2]The Wikifier was trained on a pre-2011 Wikipedia dump (Ratinov et al., 2011) with redirects from May 2013 and relations from DBPedia 3.8

Candidate Ranking We now rank the candidates following these essential steps:

- The first step is mainly based on the prior $P(entity|mention)$ and $P(mention|entity)$ (for a given page in Wikipedia, the distribution over the various text spans hyper-linked to it and vice versa).
- Then, we rank the candidates again based on the context compatibility by considering TFIDF similarity of groups of words around the mention and each candidate, using multiple window sizes. We then rank the top candidates based on the similarity of the link structures of their corresponding Wikipedia pages.
- Lastly, we identify important textual relations and enforce relational constraints (Cheng and Roth, 2013), including coreference relations, by matching relation arguments against Wikipedia infobox (DBPedia) and page-link relations, on the entire document. Let e_i^k denote the kth candidate for the ith mention. This stage promotes and demotes pair $r_{ij}^{(k,l)}$ of entity candidates e_i^k and e_j^l to maximize the following objective function of linking coherency, where s_i^k and $w_{ij}^{(k,l)}$ are priors for entity candidate e_i^k

and candidate pair $r_{ij}^{(k,l)}$ respectively:

$$\Gamma_D = \arg\max_{\Gamma} \sum_i \sum_k s_i^k e_i^k + \sum_{i,j} \sum_{k,l} w_{ij}^{(k,l)} r_{ij}^{(k,l)}$$

$$\begin{aligned}
s.t. \quad & r_{ij}^{(k,l)} \in \{0,1\} && \text{Integral constraints} \\
& e_i^k \in \{0,1\} && \text{Integral constraints} \\
& \forall i \textstyle\sum_k e_i^k = 1 && \text{Unique solution} \\
& 2r_{ij}^{(k,l)} \leq e_i^k + e_j^l && \text{Relation definition}
\end{aligned}$$

1.1.3 Cross-Document Coreference Resolution

We consider using a coreference resolution system to group query mentions into clusters. We experimented with a range of pairwise clustering classifiers. The L^3M classifier (Samdani et al., 2012) performed best on the 2012 development set; in this case we trained it on contextual and surface string similarity features. In particular, the string similarity is based on NESim[3] (Do et al., 2009) and takes into account acronyms and edit distance; context features include words with different window sizes (8, 16, 24 words etc.).

We assume the mentions arrive at the classifier in a streaming fashion and only mentions to the left of the current mention are considered "seen". We experimented with the following approaches:

Trivial Link all mentions to separate clusters

AllLink We compute the pairwise score between the current mention and all other mentions (whether seen or unseen), and link it to the cluster with the highest pairwise score.

SumLink We compute the normalized sum of pairwise scores between the current mention and seen entities in each cluster. We then put the current mention in the cluster with the highest normalized score.

$\mathbf{L^3M}$ This approach is similar to SumLink, and we compute the score using a latent EM classifier as in (Samdani et al., 2012).

[3] http://cogcomp.cs.illinois.edu/page/software_view/22

Since singleton clusters (clusters of size 1) dominate the cluster distribution in our development data, trivial clustering achieves a very competitive baseline. To test our system's robustness under different cluster size distributions, we also tested the performance with certain ratios of singleton clusters removed from the data.

To address the lack of NIL cluster training data, we also used the Wikifier component to generate around 10k training queries for the cross-document clustering.

The performance of the different clustering approaches described above on development data is summarized in Table 1.

	Clustering Algorithm				
$\mathcal{R}$	Trivial	AllLink	L^3M	L^3M_s	SumLink
On TAC 2012 Data					
100%	64.54	**86.49**	83.92	82.94	86.52
75%	87.82	88.36	**89.19**	89.14	88.49
50%	92.73	92.88	**93.14**	93.12	93.08
25%	94.72	94.72	**94.89**	94.86	94.86
0%	95.84	95.84	**95.95**	95.94	95.88
On Wikifier Generated Data					
100%	43.53	79.77	**86.73**	82.68	85.45
75%	55.77	82.09	**88.09**	84.74	87.23
50%	64.38	83.60	**89.05**	86.48	88.25
25%	69.89	85.27	**89.62**	87.43	88.82
0%	73.82	86.40	**90.18**	-	89.31

Table 1: Clustering performance on development data. $\mathcal{R}$ refers to the percentage of singleton clusters removed from training/testing data. The performance measure is the standard B^3F1 regardless of whether the entity is NIL. The L^3M_s metric is obtained by removing surface string features.

1.1.4 Voting and Generating Final Output

We consider the following procedure to cluster query mentions into groups and to assign the ID of the KB entry that the mentions in the cluster refer to, or a NIL ID if such a KB entry does not exist.

From Wikifier, for each query i, we get a list of confident Wikipedia titles $\{t\}$ with corresponding ranking scores $r_i(t)$ and a linker score l_i for whether the answer t_i should be NIL. Then, we consider jointly deciding the references of query mentions in

the same cluster by the voting schemes described in (Ratinov and Roth, 2011). This approach provides a more robust answer when evaluating on the development set. We consider the following two ways of voting:

Max The most confident candidate entity across all query references:

$$t^* = \arg\max_t \{r_i(t) | \forall i, l_i \geq 0\} \quad (1)$$

Sum The most confident candidate entity by summing all its ranking score across all query references:

$$t^* = \arg\max_t \sum_{\forall i, l_i \geq 0} r_i(t) \quad (2)$$

We used the $l_i \geq 0$ threshold for all our runs.

Note that the assignment to a cluster can be NIL if $l_i < 0$ for all mentions i in the cluster. In such cases, we assign the cluster to a non-KB entry with a unique NIL ID, otherwise the cluster is mapped to a Wikipedia entry but the entry does not appears in the provided knowledge base, in which case we also assign a unique NIL ID.

We combine the output of the Wikifier obtained above with the cross-document coreference clusters by taking the most confident candidate in the cluster. On the development set, the Wikifier and the cross-document coreference system achieve around 75% agreement in terms of the pair-wise cluster membership test.

1.2 EL Results

This section describes the results of experiments we ran on our Entity Linking system on the development data and on the evaluation data.

1.2.1 Results on development data

We used the 2012 TAC Entity Linking data to develop our system. Table 2 summarizes the system performance on this data set.

Note that if we use the query offset exactly, the performance drops significantly. This phenomenon demonstrates the importance of better coreference resolution for the Entity Linking task, as the first two approaches, which use a voting procedure to introduce candidates from other mentions, help the performance.

	Metrics			
Approach	MA	B^3 Precision	B^3 Recall	B^3 F1
Max	75.8	70.5	72.1	71.3
Sum	75.8	70.6	72.2	71.4
Exact	71.5	64.9	68.0	66.4

Table 2: System performance on 2012 Entity Linking queries. Sum and Max are runs as explained in Sec. 1 and the Exact run is obtained by taking the top answer of the exact query mention.

1.2.2 Performance on TAC 2013 Evaluation Queries

We also include the official results for TAC 2013 Entity Linking evaluation: The official result for our submitted 5 runs is presented in Table 3

	Metrics			
Approach	MA	B^3+ Prec.	B^3+ Recall	B^3+ F1
Max	79.6	77.0	60.5	67.8
Sum	79.8	77.2	60.5	67.8
Norm. Max	80.6	77.9	62.0	69.1
Norm. Sum	**80.7**	**78.2**	62.0	69.2
No Thres.	80.5	77.6	**62.8**	**69.4**

Table 3: Official system performance on 2013 Entity Linking Evaluation queries. *No Thres.* run uses the Max voting method and ignores negative linker score. Sum and Max are runs as explained in Sec. 1. Runs with the prefix *Norm.* incorporate query normalization.

A more detailed performance break-down by different document domains and query types is summarized in Table 4.

1.3 EL Discussion and Conclusions

1.4 Error Analysis

To understand our system performance better, we also analyze our system errors on the 2013 gold evaluation data by checking mismatched query clusters in Table 5.

Error Type	Percent	Main Causes	Gold	Error
Link to NIL	19.74	spelling mistakes, nick names	*['Merica]* → *United_States*	NIL
Mismatch	63.82	coreference, entity typing	Bears or *[Seattle]* fans → NIL(*Seattle Seahawks*)	Seattle
Misc.	16.44	candidate typing etc.	*[Great Britain]* → *United Kingdom*	Great Britain

Table 5: Entity Linking errors by types. For Link to NIL type errors we fail to identify a known concept. For mismatch errors both system output and gold annotation are either NIL or non-NIL with different ID.

	Approach				
Domain	Max	Sum	N.Max	N.Sum	N.T.
All	67.8	67.8	69.1	69.2	69.4
KB	65.6	65.8	67.9	68.2	68.6
NIL	70	69.7	70	69.8	70
NW	75.4	75.5	76	76.1	77
WB	64	64.2	64.3	64.4	63.9
DF	57.4	57.2	60.4	60.3	60
PER	69.9	70.2	70.2	70.6	70.8
ORG	63.9	63.7	63.8	63.7	63.5
GPE	69.2	69.1	72.5	72.4	73.3

Table 4: Official system performance break-down. The metric used is the official modified B^3F1 (Ji et al., 2011). N.T. refers to No Threshold. N.Max and N.Sum refers to Normalized Max and Normalized Sum respectively as described in Table 3

Due to the differences in task definitions (for example, only named entities of types PER, ORG and GPE are queried in the KBP Entity Linking task), we need better adaptation of Wikification tools to the TAC Entity Linking domain. One possibility in the future would be extracting high quality entity types and restricting output to only the relevant named entity types. In Table 4, we can see that

- Our system performs worst on the ORG type queries and the corresponding performance is insensitive to different approaches. The cause is revealed by the *Mismatch* error in Table 5. In the Wikification task we would always link *Seattle* in the *[Seattle] Seahawks* to the Seattle City (GPE), whereas in Entity Linking it should link to the team (ORG) Seattle Seahawks.
- We did not use any resources external to Wikipedia to perform systematic spell checking for the queries. This hurts the performance on the DF (Discussion Forum) data significantly, as the language use from this domain is usually informal and error-prone.

2 Illinois Slot Filler Validation System

The Slot Filler Validation (SFV) task resembles, when each candidate answer (query) is considered separately, the problem of Recognizing Textual Entailment (RTE). We approach the SFV task by finding matches in the relevant document for the query arguments and predicate and then determining whether together they satisfy the desired relation. If all these conditions are met sufficiently well, the system predicts that the proposed answer is entailed, and therefore correct.

2.1 SFV System Overview

The UI-CCG SFV system has five stages: 0) use NLP tools to preprocess the document text; 1) find matches for query arguments in the proposed document; 2) check argument compatibility with the relation named in the query; 3) find matches for relations in the vicinity of matched arguments using hand-written rules; and 4) make a decision based on the evidence from stages $1-3$. The architecture is summarized in Figure 2.

2.1.1 Preprocessing

The TAC KBP document collection was first processed to remove XML and HTML markup and stored in a Lucene database. At run-time, the relevant document text is retrieved from this database and processed with a range of Illinois NLP tools. First, the text is "cleaned" to remove problematic character sequences. Then, using the Curator (Clarke et al., 2012), the text is processed

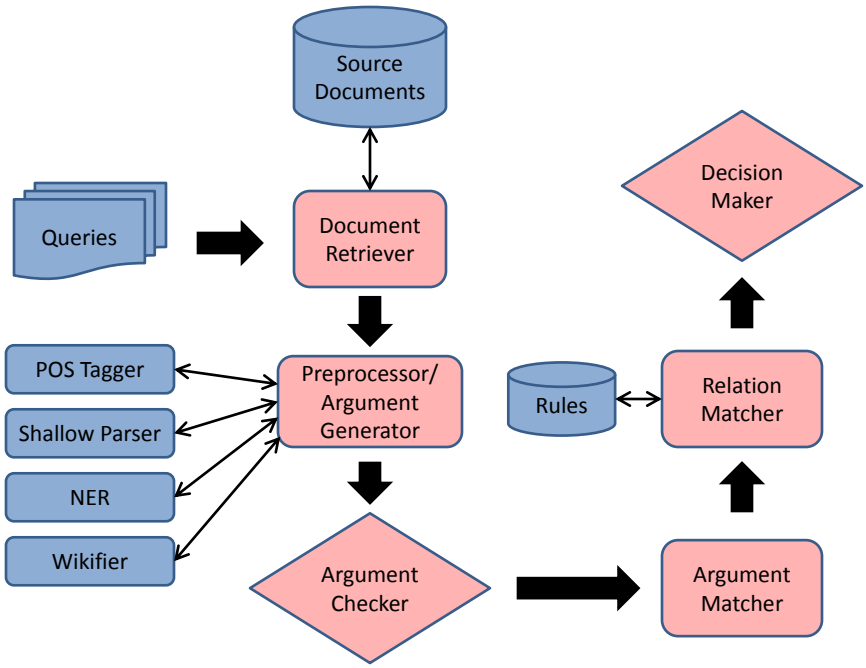

Figure 2: UI_CCG Slot Filler Validation system architecture

with the Illinois tokenizer, POS tagger (Roth and Zelenko, 1998), Shallow Parser (Punyakanok and Roth, 2001), Named Entity Recognizer (Ratinov and Roth, 2009), and Wikifier (Ratinov et al., 2011; Cheng and Roth, 2013).

The document text is often noisy, especially when drawn from the News Group and Discussion Forum subsets of the KBP data. We implemented some heuristics to clean up character sequences likely to cause errors in the preprocessing tools. The problematic sequences appear to come from several distinct sources:

- Creative use of punctuation for formatting. This was prevalent in the weblog and newsgroup data sets, and two common instances consisted of repeated punctuation to indicate document section boundaries; and use of characters such as asterisks or tildes to represent bullet point markers.
- Complete or partial inclusion of XML/HTML markup. This was found mainly in KBP system responses, presumably due to imperfect character offset calculations by participating Slot Filler systems.
- Imperfect translation between character encodings. The observed results of these problems were the use of unexpected characters or character sequences for section breaks (in the case of documents presented as single newspaper articles, but which consist of a set of topically connected, but factually unrelated short items), or of other characters such as quotation marks or non-standard spacing or dashes.
- Over-long documents. The cause of this problem was mainly that we lacked a parser to handle forum data, which comprised long sequences of related posts. In a few other cases, apparent processing errors in whatever method was used to generate the source corpus led to some instances that concatenated multiple news documents in a single file.

Some of the resulting malformed character sequences create problems for the NLP tools applied during preprocessing. To mitigate these problems, all documents are first processed using the following heuristics: characters were mapped to their ascii

equivalents where possible, and removed otherwise; quotations were normalized; and sequences of repeated punctuation characters were removed. In addition, all xml/html markup was removed from query slot fillers.

This significantly increased the number of query documents that the system could process. For the 2012 data, NLP processing errors resulted in $4,395$ queries being ignored (automatically labeled as incorrect) before the cleanup heuristics were introduced. After they were introduced, the number of documents that could not be processed fell to $2,007$ – a reduction of over 50%.

To address the problem of long documents, articles whose text exceeded $100,000$ characters were truncated at the nearest sentence boundary to that character limit. This is obviously not an ideal solution, and in future work we plan to write a forum parser that segments forum sequences in an intuitive way.

2.1.2 Candidate Argument Generation

The SFV query data includes a very limited set of types for query subjects (Person or Organization). Moreover, many of the relations specified by the task imply type constraints on both arguments. We identified a number of relevant coarse-grained argument types (Person, Organization, Location, State, Province, City, Nationality, Date, and Number) and wrote a set of simple constraints that the different relations imposed on the possible types of their arguments. These constraints jointly specified links between subject and object types: for example, *date_founded* requires an Organization as its subject and a Date as its object, while the relation *employee_or_member_of* allows two cases: one where the subject is a Person and the object is an Organization; and the other where the subject and object are either Organizations or Locations (since we did not have a reliable NLP component to identify Geopolitical entities, which might be labeled as either Locations or Organizations). We used these constraints to identify necessary argument type resources, as well as a means to filter out inappropriate candidate argument matches in the candidate slot filler source document. To further increase recall we also allowed some constituents to pass through the filter even if they violated the constraint. This was permitted as long as the constituent could not be classified as any other type.

Candidate argument boundaries are generated from constituents identified by the Named Entity Recognizer, shallow parser, and Wikifier. Where possible, types are inferred from the constituent labels or, in the case of dates and numbers, using regular expressions to detect appropriate patterns. For sub-types of Location, we experimented with using gazetteers that we hoped would be sufficiently exhaustive – for example, for cities, states, and countries. However, we found that only our Cities resource had sufficient coverage to allow its use as part of the argument labeling system.

Through early error analysis on the 2011 RTE/Slot Filler Validation data, we found that the system missed a significant number of candidate arguments in the source text, either because no constituent covered the relevant span, or because no type was found for a constituent that covered the matching span. We therefore extended the candidate identification component to search for exact matches for query subjects and objects in the source text, and creating the appropriate argument constituent when exact matches are found. If an unlabeled argument constituent is found, the system checks all labeled constituents found in the document for one with the same surface form, and if one is found, its label is attached to the unlabeled constituent.

We consider these steps to be examples of *Purposeful Inference*, by which we mean adapting the system's interpretation of a source document based on the specified information need. This strategy potentially introduces noise, as system slot fillers are often incorrect, but is a high-precision means of increasing recall for the query entities, whose type is necessarily compatible with the surface form. In the case of object entities, it increases recall at the early stages of the system, deferring the entailment decision to later stages.

2.1.3 Argument Matching

To identify whether or not a candidate argument in a source document matched an argument specified in a query, the Illinois Named Entity Similarity tool (NESim) (Do et al., 2009) is used. This tool models some typical variations of name representation, including acronyms, nicknames, and partial

name matching. Those candidates that match with sufficiently high score are retained, and the rest are removed.

NESim is a high-recall matching resource, and potentially leads to numerous false positives in the case of partial name matches or acronym matches. For both the query subject and the object, the system therefore requires that at least one candidate anywhere in the document is a "strong" match with the argument (matches 75% or more of the tokens in the span); this mitigates NESim's relaxed matching rules by avoiding last-name-only matches, or homographic acronyms to match incorrect expanded names.

The system next generates all pairs of subject-object candidates. Each pair is then checked for compatibility using the constraints described in Section 2.1.2. Although some relations (like charges) have no obvious object type, we found that we could still improve results by including constraints such as "not a date", "not a number", etc. This component is designed to be high recall, filtering out the most obvious mistakes.

2.1.4 Relation Matching

The Relation Matching stage uses hand-coded rules that specify lexical patterns that can account for argument position. Each rule is associated with a specific relation, so rule arguments are subject to the type constraints imposed via the argument checker. Rule lexical terms are lemmatized, and text tokens are lemmatized at the time of comparison to abstract over inflected forms.

Rules were generated by the authors based on intuition, and extended using error analysis of two sets of 3000 examples from the 2011 and 2012 Slot Filler Validation data sets. Some rules are duplicated across relation types. The system currently uses about 600 rules, although an improved rule syntax would reduce duplication and would therefore significantly reduce the number of rules. Figure 3 shows examples of the two main types of rule: Adjacency rules, which focus on highly localized patterns that tend to occur in the immediate vicinity of relation arguments; and Standard rules, which encode patterns that are less localized.

In Rule 1, "adj:" terms fix the positions of the subject (SUBJ) or object (OBJ) argument relative to the lexical elements of the rule. This rule encodes the pattern in which an alternate name is given in parentheses immediately after another name.

In Rule 2, the lexical terms are lemmas that must be matched in the vicinity of two arguments that match the query subject and object.

Match precision is controlled via a set of parameters to allow emphasis on precision or on recall of different aspects of the rule match. For example: How far away can the subject/object be from a rule match term? Do all the rule components have to match? Does the order of components matter? Can arguments be matched separately? The last parameter behaves as a weak proxy for coreference; more is said about this in the Discussion (Section 2.5).

2.1.5 Decision

The Decision component integrates the argument and relation match information and assigns the final label to the query.

We tried two versions of the decision component. The first, hereafter referred to as the *rule-based* system, uses a deterministic procedure with a tuned threshold to label queries: development data (in the form of a subset of the 2011 RTE SFV data or of the 2012 KBP SFV data) is used to optimize a threshold that determines how many rule terms must be matched to infer that the relation holds. Presently, a single threshold is learned for all rules and all relations.

The second version, which we call the *learning-based* system, uses a machine learning algorithm to train an SVM classifier that is used to predict the query label. Optimal parameters were tuned for the SVM using grid search, and during training the positive examples were given 6 times the weight of the negative examples to overcome the imbalanced data set.

This learning-based system encodes features based on previous component decisions by the argument matcher and the relation identifier's rule applications. The strategy aims to learn how to correct mistakes made by the rule-based system, and to expand the coverage of the rule based system. Each feature was conjoined with the query relation and so one weight vector was used to learn the parameters for all relations. The features used included whether or not a rule was triggered; and whether a sentence

Rule 1: **alternate_names @@@ adj:OBJ; (; adj:SUBJ;)**

Target example: **Marion Robert Morrison (John Wayne)**

Rule 2: **charges @@@ serve; years; for**

Target example: **Wilson served seven years for armed robbery**

Figure 3: Examples of Slot Filler Validation rules. The leftmost term names the relation to which the rule applies, while the terms to the right of "@@@" specify the rule itself (see section 2.1.4). For each rule, the "Target example" indicates a text span that would match the rule.

had ended between the query arguments when a rule was triggered. Additional features used to expand the coverage of the system included the minimum distance between the query arguments when a rule was not triggered as well as the unigrams and POS tags between the query argument pairs when they were both in the same sentence.

We also experimented with a more expressive feature set which had features for particular rules in order to try and learn optimal conditions for specific rules instead of just the relations. In addition to the previous features, these included the maximum distance between all matching components of a rule and the subject and object as well as which tokens of the specific rule were actually found in the document.

2.2 SFV Results

To develop and train our Slot Filler Validation system, we used data from the 2011 RTE Slot Filler Validation task and from the 2012 KBP Slot Filler assessed results. This section describes the selection and use of development, training, and test data from previous TAC KBP-related tasks, and the performance on the 2012 and 2013 tasks.

2.3 Training/Development Data

To develop the UI-CCG system, we used the 2011 Slot Filler Validation data from the TAC RTE track, and generated a comparable data set ("2012 SFV") from the 2012 KBP Slot Filler queries, system outputs, and TAC annotator assessments. For final evaluation of the learning-based system, we split the 2012 data into two halves, a training set and a test set, keeping the proportions of queries for each relation type comparable to the data set as a whole. We developed rules and features initially on a 3000-example sample of the 2011 RTE SFV data. We then generated a 3000-example sample from the 2012 KBP data and analyzed the errors on this data set to improve rules and features. For the rule-based system without learning, we tuned a decision threshold on the 3000-example subset and evaluated on the entire 2012 data set. We also ran the system with a higher threshold to produce a more conservative output.

For the system with the learning component, during development we trained the system on the 3000-example 2012 SFV subset and evaluated on a second 3000-example subset. When we had identified two good feature sets, we trained on the training set of the 2012 data and evaluated performance on the test set.

2.4 System Performance

The results for the four system configurations on the entire 2012 SFV data set are reported in Table 6, with the baseline being the performance if no SFV results are filtered. The learning-based systems were evaluated only on the test set of the 2012 data. We also show the results of 10-fold cross-validation on the entire 2012 corpus.

The results for these same system configurations on the entire 2013 SFV data set are reported in Table 7. The last two entries refer to training on half of the 2013 data and testing on the remainder.

2.5 SFV Discussion and Conclusions

The results in the previous section indicate that our overall approach is reasonably successful at improv-

System Configuration	Precision	Recall	F1
Baseline – always say 'YES'	0.181	1.000	0.307
Argument Checker only	0.184	0.986	0.310
Argument match plus Argument Checker	0.280	0.798	0.415
Rules, no learning, low threshold (0.55)	0.449	0.667	0.537
Rules, no learning, high threshold (0.85)	0.475	0.558	0.513
Learning, coarse features	0.402	0.766	0.527
Learning, expressive features	0.402	0.687	0.507
Learning, coarse features, 10 fold CV	0.447	0.827	0.581
Learning, expressive features, 10 fold CV	0.489	0.775	0.600

Table 6: **Performance on KBP 2012 data.** "Argument Checker only": say "NO" if argument checker fires, otherwise say "YES"; "Argument Match plus Argument Checker": say "YES" if both arguments match and Argument Checker does not fire, don't use relation matching

System Configuration	Precision	Recall	F1
Baseline – always say 'YES'	0.248	1.000	0.398
Argument Checker only	0.254	0.967	0.402
Argument match plus Argument Checker	0.305	0.825	0.446
Rules, no learning, low threshold (0.55)	0.349	0.708	0.467
Rules, no learning, high threshold (0.85)	0.367	0.604	0.456
Learning, best features	0.338	0.783	0.472
Learning, most expressive features	0.330	0.398	0.361
Learning, best features, 2013 Dev	0.400	0.863	0.547
Learning, most expressive features, 2013 Dev	0.469	0.604	0.528

Table 7: **Performance on KBP 2013 data of system configurations developed and tuned on KBP 2012 data with the exception of the last two learning systems which were trained on half of the 2013 data.** "Argument Checker only": say "NO" if argument checker fires, otherwise say "YES"; "Argument Match plus Argument Checker": say "YES" if both arguments match and Argument Checker does not fire, don't use relation matching

ing the raw KBP Slot Filler results. In this section, we assess specific aspects of the UI_CCG Slot Filler Validation system and propose future work for each.

2.5.1 Assessing the Textual Entailment Approach

The entailment-based approach we have used for slot filler validation addresses two key challenges: applying entailment recognition at large scale (tens of thousands of documents) and recognizing entailed relations from entire documents. These arise from the properties of the KBP slot filler task, but are representative of typical Information Extraction needs.

The system performance is good enough to improve over raw Slot Filler system behavior, but there is clearly a lot of room to improve precision and even recall. Moreover, the current system is oriented toward filtering results from other NLP systems, and not towards querying a large document collection itself.

We plan to extend our system by accounting for local inference operations beyond the local matching capabilities incorporated into the system thus far. The current system, tuned for higher recall, will serve as a filtering step, and the deeper system will be applied to relatively few candidate documents in cases where simpler methods are unreliable.

2.5.2 Rule-based Approach

The rule-based approach, and the rule encoding that we used, is very straightforward but quite effective for the simple relations specified in the Slot Filler task. The rule syntax is a convenient way to represent surface variations in representation of many relations, and constitutes a straightforward mechanism for extending the system to recognize new relations.

It is clear that the syntax we used to encode rules can be improved to reduce duplication (for example, by assigning numbers to rules, and mapping individual rules to multiple compatible relations). However, we also believe that further abstraction encoding multi-word-expressions such as named entities, and local syntactic structure such as appositions, possessive constructions, and modifier-head structures would improve expressiveness and specificity.

We evaluated two models of rule application: in one, both arguments had to be sufficiently close to a single set of terms that matched the rule; in the other, the arguments had to be sufficiently close to *some* set of terms that matched the rule. In the latter case, the result was a crude approximation of co-reference: in Figure 4 the term "died" matches in two locations, each near a strong match for the subject or object, but not both. This heuristic also introduces errors, but we found that overall there was a significant improvement in F1 using this relaxed model.

Finally, we observe that each relation match decision is presently independent of all other such decisions, but that when multiple compatible entities are in close proximity (see Figure 5), some relations may impose constraints on others. In the example shown, "Josh Smith" is in the "title" relation with "CEO", and so cannot also be in the "title" relation with "Vice President of Operations".

We plan to extend the decision model to account for these additional constraints, which we feel will be easier with the more expressive rule syntax described above and also by further clustering the rules in a way that each cluster will have its own set of features. For instance, those rules which include a verb and an object should have SRL features, while those rules in which an adjective is modifying a noun would benefit from features derived from dependency parsing.

2.5.3 Lexical Similarity

One potential source of improvement would be to generalize the rules by using lexical similarity. We experimented with a version of WNSim (Do et al., 2009), modified to account for derivationally related words, as a WordNet similarity metric. We used a threshold that was deemed optimal from previous RTE tasks to expand the vocabulary specified by the rules. However, on the 2012 data, the performance of the rule based system dropped from an F1 of 0.537 to 0.482. As expected the recall increased from 0.667 to 0.729, but the precision fell from 0.449 to 0.360. To improve our system, we plan to explore other ways of using lexical similarity in order to improve our results.

QUERY: Vitaly Ginzburg cause_of_death heart failure

DOCUMENT: Nobel Physics prize winner **Vitaly Ginzburg**, who helped develop the Soviet hydrogen bomb, has **died** at age 93, the Russian Academy of Sciences said Monday. Ginzburg said the bomb "saved" his life during an anti-Jewish campaign. But he had drawn controversy in recent years with fierce public criticism of the Russian Orthodox Church, which has enjoyed surging popularity and political influence since the fall of the atheist Communist regime. "He **died** from **heart failure**," Irina Presnyakova, a spokeswoman for the Russian Academy of Sciences...

Figure 4: Example of "relaxed" rule matching allowing coreference-like behavior.

The award was accepted by [CEO] [Josh Smith] and [Vice President of Operations] [Celia Jones].

Figure 5: Example of contraints arising from multiple relations.

2.5.4 Learning-based approach

Learning was difficult in this task. There are many relations and the dataset is significantly skewed towards negative examples, resulting in relatively few positive examples for many relations. In addition, our results suggest that data from 2012 was significantly different from that of 2013 as training on 2012 did not generalize well to the 2013 data.

Not surprisingly, the more expressive feature set increased precision and lowered recall. The improved results on the 10-fold cross validation evaluation (see Table 6) suggest that this feature set works well when more training data is available.

Figure 6 illustrates a query that the rule-based system incorrectly predicts to be entailed, but which the learning system correctly labels as not entailed. The rule in question is overly general, and essentially encodes a proximity-based relation match. Two mechanisms in the learning approach may account for this improved behavior: down-weighting of an overly noisy rule, or recognition of lexical items (such as commas) that have a strong anti-correlation with a correct rule match.

2.5.5 Comparing the Rule-based and Learning-based approaches

For the Slot Filler Validation use case, the rule-based approach has the appealing characteristic of being fairly robust: performance degrades from the 2012 to the 2013 data set, but is still significantly above baseline, and degrades less than the learning-based approach. Given the challenges to a learning model – limited per-relation training data and a domain shift due to the changes in participating KBP Slot Filler systems from year to year – this approach is a reasonable starting point. In a scenario where the application domain is static, the learning approach is clearly superior. At present, the features used in the learning-based model are necessarily simple. To avoid sparsity problems, more advanced training methods will have to be investigated and more robust feature sets that are more sophisticated and abstract will be required. The use of rule applications as the basis for such features seems reasonable, provided the rules are not overly specific.

2.5.6 Summary

The application of a Textual Entailment Recognition model to the task of Slot Filler Validation is an intuitive one, but the scale of the task makes application of "deeper" NLP resources such as coreference resolvers and semantic role labelers problematic. Our approach is a straight-forward one, but performs well on the filtering task, even when there is a significant domain shift (as Slot Filler system behaviors change from year to year). The current framework allows for new relation recognition resources to be added via a simple rule syntax, and appears well-suited to development of a deeper analysis component that is applied very selectively after most of the relevant document content is filtered by earlier components.

RULE MATCHED: [[OBJ]] [[*]] [[*]] [[SUBJ]] Antis : .

QUERY: Patricia Neal origin Greek

DOCUMENT: ...a stroke. "Frequently my life has been likened to a *Greek* [tragedy] [,]" *Neal* wrote, "and the actress in me cannot deny that comparison..."

Figure 6: Example of query that learning-based system predicts correctly, but the rule-based system does not.

References

K.-W. Chang, R. Samdani, and D. Roth. 2013. A constrained latent variable model for coreference resolution. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

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

J. Clarke, V. Srikumar, M. Sammons, and D. Roth. 2012. An nlp curator (or: How i learned to stop worrying and love nlp pipelines). In *LREC*, 5.

Q. Do, D. Roth, M. Sammons, Y. Tu, and V. Vydiswaran. 2009. Robust, light-weight approaches to compute lexical similarity. Technical report.

Heng Ji, Ralph Grishman, and Hoa Trang Dang. 2011. Overview of the tac 2011 knowledge base population track. In *Fourth Text Analysis Conference (TAC 2011)*.

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

L. Ratinov and D. Roth. 2009. Design challenges and misconceptions in named entity recognition. In *CoNLL*.

L. Ratinov and D. Roth. 2011. Glow tac-kbp 2011 entity linking system.

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*.

R. Samdani, M. Chang, and D. Roth. 2012. Unified expectation maximization. In *Proc. of the Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.