CUNY-BLENDER TAC-KBP2010 Entity Linking and Slot Filling System Description

Zheng Chen, Suzanne Tamang, Adam Lee, Xiang Li, Wen-Pin Lin, Matthew Snover, Javier Artiles, Marissa Passantino, Heng Ji[*]

Computer Science Department
Queens College and Graduate Center
City University of New York
New York, NY 11367, USA
[*]hengji@cs.qc.cuny.edu

Abstract

The CUNY-BLENDER team participated in the following tasks in TAC-KBP2010: Regular Entity Linking, Regular Slot Filling and Surprise Slot Filling task (per:disease slot).

In the TAC-KBP program, the entity linking task is considered as independent from or a pre-processing step of the slot filling task. Previous efforts on this task mainly focus on utilizing the entity surface information and the sentence/document-level contextual information of the entity. Very little work has attempted using the slot filling results as feedback features to enhance entity linking. In the KBP2010 evaluation, the CUNY-BLENDER entity linking system explored the slot filling attributes that may potentially help disambiguate entity mentions. Evaluation results show that this feedback approach can achieve 9.1% absolute improvement on micro-average accuracy over the baseline using vector space model.

For Regular Slot Filling we describe two bottom-up Information Extraction style pipelines and a top-down Question Answering style pipeline. Experiment results have shown that these pipelines are complementary and can be combined in a statistical re-ranking model. In addition, we present several novel approaches to enhance these pipelines, including query expansion, Markov Logic Networks based cross-slot/cross-system reasoning. Finally, as a diagnostic test, we also measured the impact of using external knowledge base and Wikipedia text mining on Slot Filling.

1 Introduction

The CUNY-BLENDER team participated in the Regular Entity Linking, Regular Slot Filling and Surprise Slot Filling (per:disease slot) tasks in the TAC Knowledge Base Population 2010 track. We submitted two runs for regular entity linking, one is based on VSM model using tf-idf similarity between context document of the query and knowledge base (KB) text, and the other run is to incorporate some semantic features extracted from slot filling into the VSM model. In this paper we will focus more on the description of our slot filling system which achieved more competitive results.

The Slot Filling task requires a system to automatically distill information from a large document collection and return answers for a query entity with specified attributes (‘slots’), and use them to expand the Wikipedia infoboxes while validating the answer with a cited document from the collection. Many existing techniques could be used to solve this task such as (1) mapping Information Extraction (IE) results to slot answers (Bikel et al., 2009), (2) training new pattern-matching systems, or (3) using a Question Answering (QA) system to fill slots (Li et al., 2009). There has been a question whether bottom-up style IE methods or top-down style QA methods are more suitable for slot filling. While these methods each have their own inherent strengths and weaknesses, we decided to investigate the extent to which automatic combination of these techniques might exceed their individual limitations. We will compare the detailed techniques in these pipelines and the lessons we have learned from experiments and error analysis. In general we found that the supervised IE pipeline performs worse than the other two because about many instances require cross-sentence and cross-slot inferences which are beyond traditional IE task. The pattern matching pipeline achieves the highest recall while the QA pipeline has the highest precision. We shall further demonstrate that these three pipelines are complementary and can be combined effectively in a novel statistical re-ranking module based on Maximum Entropy. The features used in learning-to-rank include local evidences such as baseline confidence, name type, slot type, gazetteer constraints and dependency parsing.

In addition to this core combined system, several novel approaches were used to enhance these pipelines, including query expansion and Markov Logic Networks based cross-slot/cross-system reasoning. In the slot filling task, each slot is often dependent on other slots, but the previous systems usually ignore these dependencies and process each slot individually. We developed a reasoning component to approach a real world acceptable answer in which all slot dependencies are satisfied. All of these new methods achieved significant improvements over the baseline pipelines.

Finally, for diagnostic analysis, we will briefly present the impact of using external knowledge base and Wikipedia text mining on Slot Filling. For the surprise slot filling task we will show that using a small list of common disease names can perform as well as a much larger list.

2 Entity Linking System Overview

Figure 1 depicts the general procedure of our approach.

3 Entity Linking Approach

3.1 Preprocessing

In the preprocessing step we use Lucene[1] to index the knowledge base according to the following fields (Table 1):

[1]http://lucene.apache.org/

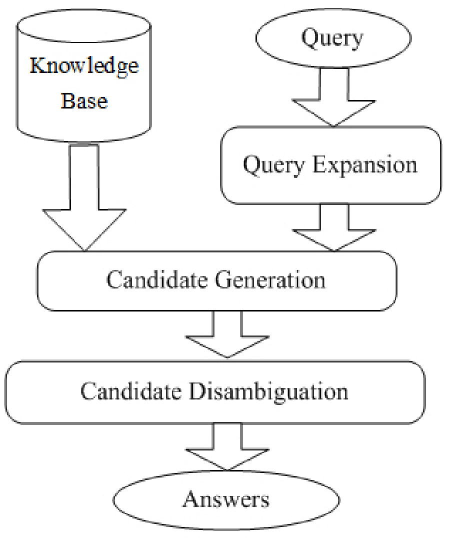

Figure 1: Framework of CUNY_BLENDER Entity Linking System

Field	Description
title	the entity name in wiki format
id	a unique KB node ID, e.g., E000001
type	the entity type, PER, ORG, GPE or UKN
name	the entity name in plain format
facts	attribute-value pairs extracted from infobox
text	KB text

Table 1: Fields Indexed in the Knowledge Base

In the candidate disambiguation step we will be using the field “name” to retrieve candidate KB nodes and the field “text” to retrieve KB text.

Then we index the source text corpus by three fields: *docID*, *title*, and *text*.

Finally we use the Java Wikipedia Library Toolkit[2] to process the Wikipedia English dump[3] obtained in July, 2010. Processed results are stored in a MySQL database which allows us to access the vast amount of entity knowledge (e.g., redirections, disambiguation tables) instantly.

3.2 Query Expansion

It is usually not sufficient to retrieve candidate KB nodes just using the query’s name string. Therefore, we expand the query in various ways. We

[2]http://www.ukp.tu-darmstadt.de/software/jwpl/

[3]http://download.wikimedia.org/

use the following notations: Q means a query, *Q.querystring* for the query's name string, *Q.text* for the query's context text, *Q.fullname* for query's full name if *Q.querystring* is an acronym and *Q.expandset* for the query's expanded set.

1. Initially, add *Q.querystring* into *Q.expandset*.
2. If *Q.querystring* contains all capital letters, we consider it as an acronym. We expand it to *Q.fullname* by searching the text span around *Q.querystring* in *Q.text*. *Q.fullname* matches the pattern of "*Q.fullname* (*Q.querystring*)" and concatenate all the capital letters in *Q.fullname* to form *Q.querystring*. We add *Q.fullname* into *Q.expandset* if *Q.fullname* is not empty.
3. Get the Wikipedia page *P* whose title is *Q.querystring*. If *P* is a redirect page, we add the target page's title into *Q.expandset*.
4. Get the Wikipedia page *P* whose title is "*Q.querystring (disambiguation)*". If *P* exists, we get the list of Wikipedia page links in *P* which refer to different meanings of *Q.querystring*. We add the titles of those target pages into *Q.expandset*.
5. If at this point, *Q.expandset* only contains *Q.querystring* which means the name string may be a misspelling. We get the suggested string ("did you mean" functionality) using the spellchecker in Lucene. If such string exists, we add it into *Q.expandset*.
6. If at his point, *Q.expandset* still only contains *Q.querystring*, we apply a fuzzy search in the knowledge base, and retrieve the names which share at least two common words with *Q.querystring*. If any such names exist, we add them into *Q.expandset*.

3.3 Candidate Generation

For each string in *Q.expandset*, we form a *PhraseQuery* by treating the words in the string as an "AND" relation. We then form a *BooleanQuery* by concatenating the *PhraseQueries* as an "OR" relation and use Lucene to retrieve top 100 candidate KB nodes from the knowledge base (only searching the field "name" in the index). We use the notation *Q.candidateset*=$\{Q.kb_1, Q.kb_2, \ldots, Q.kb_K\}$ to represent the set of candidate KB nodes.

3.4 Candidate Disambiguation

If *Q.candidateset* is empty, which means that we cannot find any candidate KB nodes, we simply return NIL as the answer. If *Q.candidateset* only contains one item, we select it as the final answer. When there are multiple items in the set, we should disambiguate the candidates and find the most probable one as the answer; meanwhile we need to distinguish NIL and non-NIL if possible.

We implemented two approaches for candidate disambiguation as follows.

3.4.1 Basic VSM Model: Bag-of-Word(BOW)

The intuition behind the basic vector space model (VSM) is that the more similar (based on word co-occurrence information) between the KB text $Q.kb_k.text$ with the context text of the query *Q.text*, the more likely the KB node refers to the query. We constructed this baseline model based on bag-of-word features, with stop words removed.

Once we computed every similarity for $Q.kb_k$ $(1 \le k \le K)$, we select the candidate with the largest similarity. In order to distinguish NIL from non-NIL, we introduce a threshold T. If the largest similarity is lower than T, the system returns NIL, otherwise returns the candidate.

3.4.2 Enhanced VSM Using Attributes

The intuition behind enhanced VSM is that entities can be disambiguated by their attributes, e.g., the "Michael Jordan" as a basketball player can be differentiated from the "Michael Jordan" as a politician by the attribute of "title". We define the *profile* of an entity as a list of attribute-value pairs. We use attribute vectors $A_1 = \{a_{11}, \ldots, a_{1n}\}$ and $A_2 = \{a_{21}, \ldots, a_{1n}\}$ to represent *Q.profile* and $Q.kb_k.profile$ respectively. Each item (e.g., a_{11}) in the vector is a set of values for some type of attribute (some attributes are single-valued, and some attributes are list-valued).

The KBP2010 Slot Filling task defines 26 attributes for PERSON, and 16 attributes for ORGANIZATION. Therefore attribute vectors can be formed by applying slot filling. For *Q*, it is usually not enough to extract any information from a single context text *Q.text*, hence, top K similar documents can be retrieved from the source text corpus which contains *Q.querystring* and then slot filling is applied on the set of documents. The other issue concerned with *Q* is that *Q* does not give the entity type explicitly; therefore, entity tagger should be applied to tag the entity type of *Q* in *Q.text*. The entity type determines whether person or organization attributes should be extracted. For $Q.kb_k$, slot filling can be applied solely on the KB text $Q.kb_k.text$ because the KB text is usually rich in information of the entity. Furthermore, $Q.kb_k$ explicitly tells the entity type.

The similarity between $Q.profile$ and $Q.kb_k.profile$ is computed as:

$$sim(Q.profile, Q.kb_k.profile) = \sum_{i=1}^{n} f\left(a_{1i}, a_{2i}\right)/n$$

where $f(a_{1i},\ a_{2i})$ is a boolean function which returns 1 if a_{1i} and a_{2i} share any similar values (similar is defined as equal, substring or a standard Levenshtein distance between the two strings smaller than 2); otherwise, the function returns 0.

In our implementation we have simplified the computation of the boolean function by skipping slot filling on the clustered set of documents. We assume that slot filling on the clustered set of documents may be much harder than slot filling on $Q.kb_k.text$ because the sentences in $Q.kb_k.text$ are much more coherently organized. For each value in a_{2i} extracted from $Q.kb_k.text$, we check whether *Q.querystring* and the value occur simultaneously in some document from the clustered documents of *Q*. If it is true, the function returns 1. We check all the values in a_{2i}, if there is not any such document, the function returns 0.

We derive a hybrid similarity by combining the similarity from BOW and from attributes:

$$sim_{hybrid} = \alpha sim\left(Q.text, Q.kb_k.text\right) +$$
$$\left(1 - \alpha\right) sim(Q.profile, Q.kb_k.profile)$$

4 Slot Filling System Overview

The general procedure of our approach is shown in Figure 2. Our approach begins with an initial query processing stage where query expansion techniques are used to improve recall. Each query is then processed along three different pipelines, representing three alternative approaches to the KBP task: IE, pattern matching and QA. The best answer candidate sets are generated from each of the individual pipelines and are combined in a statistical re-ranker. The resulting answer set, along with confidence values are then processed by a cross-slot reasoning system, resulting in the final system outputs.

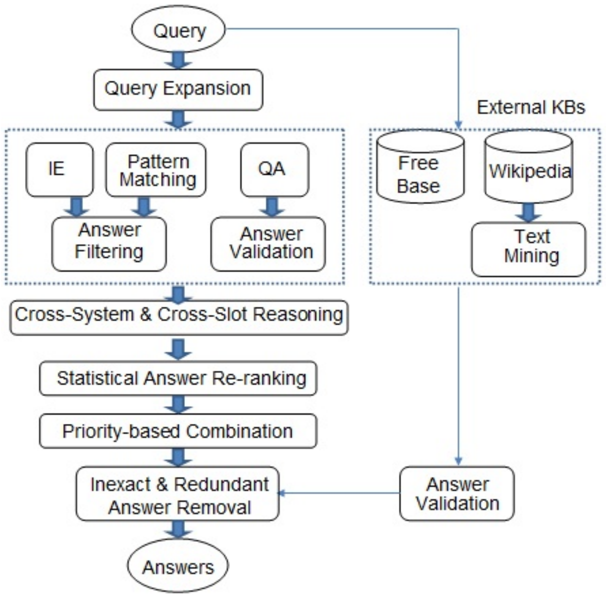

Figure 2: Slot Filling System Overview

5 Slot Filling Baselines

We developed a multi-system approach that includes four pipelines for KBP regular slot filling. These pipelines are organized in two forms: bottom-up IE based approaches that extract all possible attributes for a given query and then fill in the slots by mapping and inference (sections 5.1-5.3); and a top-down QA based approach that searches for answers constructed from target entities and slot types (section 5.4). In addition, we exploited an external knowledge base (Freebase) and automatic Wikipedia text mining for answer validation (section 5.5).

5.1 Pattern Matching

In the pattern matching approach, we learn and rank patterns from query-answer *(q-a)* pairs, and then apply these patterns to find answers to unseen queries. For example, given the pair *(Michael Jackson, 50)* for the *per:age* slot, we can extract sentences in which *Michael Jackson* and *50* co-occur:

Michael Jackson died at the age of 50; Michael Jackson (50).

A pattern can be constructed as:

$\{Q\}$ *died at the age of* $\{A\}$*;* $\{Q\}$ *(*$\{A\}$*).*

This approach consists of the following steps:

(1) Selection of query-answer pairs

We extract *q-a* pairs from the facts listed in Wikipedia infobox by some mappings infobox fields to KBP slots. *q-a* pairs are split into two sets: half for pattern extraction, and the other half for pattern assessment.

(2) Pattern extraction

For each *q-a* pair from the training set, we use a search engine to retrieve the top 1000 documents in the source collection, and select thoses sentences in which the query and answer co-occur. In addition to populating static patterns for different *q-a* pairs, we also apply entity type replacement and regular expressions to make patterns as general as possible.

(3) Pattern assessment

For each *q-a* pair from the development set, we search the top 1000 documents and pick out the sentences in which the query occurs. We apply patterns for each sentence and if it can be matched, extract the entity as the answer. We sort the patterns in descending order of precision (matching rate), and filter those with precision below a threshold.

(4) Pattern matching

To obtain candidate answers for slot filling, we locate those sentences where *q* occurs, apply the patterns generated by step (3) and extract the answer which matches any pattern. We then rank the answers according to the sum of precisions of all patterns that produce the answer.

5.2 Supervised Feature-based IE Mapping

We apply an English IE system (Grishman et al., 2005; Ji and Grishman, 2008) to extract relations and events defined in NIST Automatic Content Extraction Program (ACE 2005) . Relation extraction and event extraction are based on maximum entropy models, incorporating diverse lexical, syntactic, semantic and ontological knowledge. We apply the following mappings between ACE relations/events and KBP slots:

(1) Given a 3-tuple $\{$*emi, emj, r*$\}$ from relation extraction which indicates that the entity mentions *emi* and *emj* holds a relation *r*, and if *r* matches a slot type *r* and *emi* matches the query entity *q* in slot filling, then the answer in the incomplete 3-tuple $\{$*q, a, r*$\}$ for slot filling is *emj*.

(2) Given a 3-tuple $\{$*t, arg, e*$\}$ and *arg* = $\{$*emi*$\}$ from event extraction which indicates that the trigger word *t* indicates an event type *e* and the involving arguments in *arg* include *emi*, *emj*, and so on. If the event type *e* matches a slot type *e*, *emi* matches the query entity *q* in slot filling and *emj* satisfies the role constraint, then the answer is *emj*.

ACE2005 defines 6 main relation types and 18 subtypes; 8 event types and 33 subtypes. Table 2 shows the possible mappings from KBP2010 slots to ACE 2005 relations/events.

5.3 Answer Filtering

We set a low threshold to include more candidate answers, and then remove the following errors to distill the best answers from both of the IE and pattern matching pipelines:

- answers with inappropriate entity types (e.g. the family members lots should have person answers);
- erroneous answers that don't match dictionary constraints (e.g., the country dictionary for slot *per:country_of_birth*);
- answers that are similar with the query names (except for alternative name slots);
- some slot types (e.g. *per:parents and per:children*) cannot share the same answer, remove the answer with lower confidence;
- remove alternative names and title answers with low confidence;
- inappropriate answers whose dependency parsing paths to the query do not satisfy certain constraints (e.g., for slots *org:subsidiaries*,

KBP 2010 slots	ACE2005 relations/ events
per:*of_birth	event: be-born
per:*_of_death	event: die
per:*_of_residence, per:religion	relation:citizen-resident-religion-ethnicity
per:school_attended	relation:student-alum
per:member_of	relation:membership, relation:sports-affiliation
per:employee_of	relation:employment
per:spouse, per:children, per:parents per:siblings, per:other_family	relation:family, event: marry, event:divorce
per:charges	event:charge-indict, event:convict
org:members, org:member_of	relation:membership
org:subsidiaries, org:parents	relation:subsidiary
org:founded_by	relation:founder
org:founded	event:start-org
org:dissolved	event:end-org, event:declare-bankruptcy
org:shareholders	relation:investor-shareholder
org:*_of_headquarters	relation:org-location
org:top_members/ employees	per:employment, per:membership, per:ownership

Table 2: Mappings from KBP2010 slots to ACE2005 relations/events

org:parents, the query and the answer should not have a conjunction relation);

5.4 Question Answering

We also apply an open domain QA system, OpenEphyra (Schlaefer et al., 2007) to retrieve candidate answers for the slot filling task. Since candidate answers must be validated, additional answer processing is necessary to determine the candidate answer's relevance and retrieve the corresponding docid(s) from the source collection.

To estimate the relevance, *R*, of a *q-a* pair we use the joint probability of observing both the query and answer by means of the answer pattern probability:

$$P\,(q,\,a) = P\,(q\ \mathit{NEAR}\ a)$$

where NEAR is defined as *q-a* existing within the same sentence. At the sentence level, we calculate the frequency of *q-a* pair occurrence in the reference corpus and modify the related Corrected Conditional Probability (CCP) formula to assess relevance:

$$R(q,a) = \mathit{frequency}(q\ \mathit{NEAR}\ a) \times \#(\mathit{total\ sentences}) \div (\mathit{frequency}(q) \times \mathit{frequency}(a))$$

After the relevance scores are calculated, the values for each slot are rescaled to be between 0 and 1 in order to facilitate the comparison of relevance values among different slots.

5.5 External Knowlege Base Search and Validation

We exploited external knowledge bases to discover and validate more complete and reasonable answers.

The first resource we used is Freebase ((Bollacker et al., 2008)), which harvests information from many open data sets (for instance Wikipedia and MusicBrainz), as well as from user contributions. In Freebase, there are many different categories of data according to different criteria, for example, "*American football*" is one category, which also includes many specific divisions (*football coach*, *football player*, *player game statistics* and etc). We selected some categories and mapped them to corresponding KBP slot types, as shown in Table 3. For example, we could map the coach of a football team to the slot *org:topmembers_ employees*. As we can see from Table 3, most of the Freebase entries involve celebrities and only cover 17 KBP slot types in total. Some slots need additional processing, for example, the place slots from Freebase need to be further divided into countries/states/cities.

In addition, we also conducted automatic Wikipedia text mining by running the above three pipelines on Wikipedia texts and storing the results as an offline knowledge base.

For a given query *q* and a slot type, we search this offline knowledge base. If a candidate answer *a* is obtained, we try to validate the pair $\{q,\,a\}$ for relevance to the source text collection using the same search method and co-occurrence metric as the QA pipeline. Since Freebase provides accurate and up-to-date information, we set the confidence of $\{q,\,a\}$ pairs obtained from Freebase to 1. For the answer obtained from Wikipedia texts, we increase the confidence of $\{q,a\}$ if *q* and *a* co-occurred in the same Wikipedia text.

In the final combined system, we assign the answers obtained from this external knowledge base

Table 3: Mappings from Freebase to KBP Slots

Freebase slots	KBP slots	#entries
academic_post	per:title	299
cyclist		3593
employment_tenure		75592
golfer		1542
government_position_held		13941
person		296718
profession		363554
US_president		43
US_vice_president		47
baseball_historical_managerial_position	org:top_members/employees, per:member/employee_of	263
baseball_manager		31
basketball_coach		138
company_founder		5277
football_coach		38
organization_committee_membership		142
organization_founder		910
baseball_historical_roster_position	per:employee_of	475
basketball_historical_roster_position		247
boxer		4342
employment_tenure		100674
cricket_coach		11
cause_of_death	per:cause_of_death	25729
company_name_change	org:alternate_names	787
company_shareholder	org:shareholders	2354
founding_figure	per:religion	102
person		33162
marriage	per:spouse	12877
organization_board_membership	org:top_members/employees, per:member_of	5004
organization_membership	per:member_of	4416
person	per:children, per:parents	38398
person (ethnicity)	per:origin	30846
person (nationality)		370421
place_lived	per:residence	164053
sibling_relationship	per:siblings	8229
subsidiary_relationship	org:subsidiaries	6333
person	per:place_of_birth	303653
person	per:date_of_birth	763897

higher priorities than baseline pipelines.

6 Slot Filling Enhancement

We enhance the above pipelines based on the following extensions. We hypothesize that cleverly designed query expansion techniques (section 6.1) will improve the recall of candidate answers to the query. In addition, most slot filling methods often produce logically incoherent answers. We design a novel cross-slot reasoning approach based on Markov Logic Networks (MLN) to further refine the quality of answers and predict new answers. Furthermore, we take advantage of the redundancy from multiple pipelines to conduct cross-system reasoning (section 6.2). We applied several heuristic rules (section 6.3) to remove inexact and redundant answers compared to the knowledge base, as required by the task definition.

6.1 Query Expansion

6.1.1 Name expansion

The query name may be mentioned in its alternative names in the corpus, thus, name expansion can help improve the recall of slot filling. Wikipedia uses redirect links to indicate navigations among pages that mention the same entity. For example, the entity name "*Seyed Ali Khamenei*" is redirected to "*Ali Khamenei*". We mine redirect links from our Wikipedia database (a static copy retrieved on March, 2010) and use them to form extra query names.

Similar to query expansion in Entity Linking (section 3.2), a query name in slot filling is expanded by taking all the following cases into consideration:

1) If the query name is an acronym, search in the context doc and find the full name near the acronym;

2) Find the redirect name by a redirect dictionary from Wikipedia;

3) If the query name is an organization, find the acronym by connecting the capital letters in the name, and validate the acronym by searching in the source text corpus;

4) If the query name is a person name, expand it with the last name.

Step 2 above normally produces very reliable answers for *org:alternative_name slots* as well, and so they are propagated into the final output directly.

6.1.2 Template expansion

In order to generate informative natural language questions from each pair of {query name, slot type}, we developed the following expansion methods.

We generated 68 question templates for the 16 organization slots and 68 question templates for the 26 person slots. For example, the following semantically equivalent questions are generated for the *org:founded_by* slot type:

- Who founded {org}?
- Who established {org}?
- {org} was created by who?

During candidate answer generation, the tag {org} is replaced by the target. On average, each target value produced an initial set of 112 candidates. After filtering with stop words and sufficient query-answer co-occurrence in the source collection, each query generates an average of 4.9 for the baseline results, which suggests a very high rate of spurious results from the QA module. For this reason, query expansion is a necessary step in the QA pipeline. A rough estimate using a small set of queries without enhanced expansion suggests the impact of this step on recall leads to approximately a four-fold improvement.

6.2 Cross-system and Cross-slot Reasoning

In the slot filling task, each slot is often dependent on other slots. In particular, the family slots include such dependency relationships (e.g. *X is per:children of Y* $\rightarrow$ *Y is per:parents of X*; *X is per:spouse of Y* $\rightarrow$ *Y is not likely to be per:siblings of X*). Therefore we develop a reasoning component to approach a real world acceptable answer in which all slot dependencies are satisfied. Similarly, we can design propagation rules to enhance recall, for example, *X's per:date_of_birth is Y* $\rightarrow$ *X has per:age which is approximately (the current year Y).*

We noticed that heuristic inferences are highly dependent on the order of applying rules, and the performance may have been limited by the thresholds which may over fit a small development corpus. We use Markov Logic Networks (Richardson and Domingos, 2006), a statistical relational learning language, to model these inference rules more declaratively. Markov Logic extends first order logic in that it adds a weight to each first order logic formula, allowing for violation of those formulas with some penalty. We use the Alchemy toolkit (Kok et al., 2007) to encode inference rules such as those based on traversing family trees.

Markov Logic will make it possible to compactly specify probability distributions over these complex relational inferences, and easily capture non-deterministic (soft) rules that tend to hold among slots but do not have to. We incorporate hard rules such as *name/date/number/title* format constraints for slots, as well as soft rules such as *per:birthofdate* to *per:age* propagation.

Finally we apply the following additional heuristic rules to improve the precision of slot filling:

- **Format Validation**

  (1) *per:date_of_birth* and *per:date_of_death*: Answer has to be in a date format (e.g. mm/dd/yyyy)

  (2) *per:age*: Answer has to evaluate to an integer that is between 0 and 150.

  (3) *org:number_of_employees/members* and *org:founded*: Answer has to be an integer.

- **Gazetteer based Validation**

We apply gazetteers to filter out incorrect answers for the following slots: *per:country_of_birth*, *per:city_of_birth*, *per:origin*, *per:country_of_death*, *per:city_of_death*, *per:countries_of_residence*, *per:cities_of_residence*, *per:title*, *org:country_of_headquarters*, and *org:city_of_headquarters*.

- **Regular Expression based Generation**

  We apply the following high-precision regular expression based patterns to obtain answers:

  (1) *org:alternate_names*: ANSWER (QUERY_NAME)

  Comparing to the query name expansion technique, this pattern has the advantage of obtaining answers include tokens which are not represented by the abbreviation name, e.g. *Gay, Lesbian and Straight Education Network (GLSEN)*.

  (2) *org:website*:

  http://*QUERY_NAME*.(gov|com|edu|org) This pattern found most of the website answers by checking whether the query name exists within the web address.

  (3) *per:age*: QUERY_NAME, ANSWER,

  Some age answers were missed by the general pattern learning component due to name tagging errors. This pattern can discover many high-precision answers.

- **Regular Expression based Filtering**

  In the slot filling task, a system is required not only to return the answer, but also a context document to support the answer. So it is also important to validate the document or context sentences. We have implemented the following regular expression based rules to filter out incorrect answers as well as contexts.

  (1) ***org:subsidiaries* Filtering**

  If a candidate answer match one of the following patterns, then remove this answer for the *org:subsidiaries* slot:

  (a) Context sentence contains the word ”contract”

  (b) QUERY_NAME * to ANSWER

  For example, in the sentence “*Halliburton referred questions to Petrobras, which has said it has backup copies of the information*”, “*Petrobras*” is unlikely to be the answer of the query “*Halliburton*”.

  (c) QUERY_NAME *competitor|has been|and * ANSWER

  For example, “*Schlumberger Ltd.*” is the competitor instead of a subsidiary of “*Halliburton*” in the following sentence: “*Last week, Halliburton's chief competitor, Schlumberger Ltd., said its third-quarter profit jumped 35 percent, but weaker-than- expected results in North America drove its shares down sharply.*”

  (d) ..., QUERY, ...ANSWER, ...

  If the query and answer candidate appears in the middle of a very long comma delimited list, then the answer is unlikely to be a correct subsidiary.

  (2) ***per:employee_of* Filtering**

  (a) QUERY_NAME told ANSWER

  When the query is a news agency, the slot filling pipelines cannot distinguish the reporter and the witness well. In these cases, if the query and answer are involved in a “*tell*” event, then we remove the answer candidate. For example, in the following sentence “*David Banda*” is not an employee of “*AFP*”:

  “I only met Madonna and her husband once in court when we were signing the first documents for adoption in 2006,” Yohane Banda told AFP by telephone from his village of Mchinji, 110 kilometres (70 miles) from the capital Lilongwe.

  (3) ***org:number_of_employees/members* Filtering**

  (a) QUERY_NAME in ANSWER

  Sometimes the baseline pipelines cannot distinguish numbers from years, and thus generated incorrect answers for the *org:number_of_employees/members* slot. We use this simple pattern to remove some errors. For example, in the sentence “*Previous* *estimates from India's National AIDS Control Organisation (NACO) had put the HIV caseload at 5.2 million while UNAIDS in 2006 estimated 5.7 million cases.*", "*2006*" is not *org:number_of_employees/members* of "*UNAIDS*".

(4) ***org:top_members-employees*** **Filtering**

(a) QUERY_NAME * said ANSWER

Dependency relations are generally crucial to filter out incorrect family and employment slot answers. However when the dependency parser fails, we apply some hard constraints such as the above pattern to filter out some incorrect answers for *org:top_members-employees*. For example, in "*A FEMA spokeswoman, Allsee Tobias, said Anna Johns...*", "*Anna Johns*" is not a top employee of "*FEMA*".

(5) **Structured Data Identification and Answer Filtering**

In the web data, some structured data such as tables lost their original forms, which may lead to many spurious answers. We apply some specific patterns to automatically detect these structures and filter out errors. For example, in the following sentence "*Samsung Z720 @ 150usd Samsung Ultra Edition 5.9 @ 270usd Samsung SGHP310 @ 155usd Samsung SGHi830 @ 190usd Samsung SGHi718 @ 190uad Samsung SGHi600 @180usd Samsung SGHF700 @ 200usd Samsung SGHB600 @ 300usd Samsung i760 @ 250uad*", we rely on the common word "*usd*" to detect the tabular form and remove "*SGHB*" from the answer list for the query "*Samsung*", which was mistakenly recognized as an organization name in the baseline pipelines.

6.3 Inexact and Redundant Answer Removal

We handled the following specific rule based on within-document entity coreference resolution to convert inexact answers to full strings.

- if $\{a, b\}$ are coreferential, while a is a full name while b is not, generate a as the answer to replace b.

In addition, we applied substring matching and Levenshtein distance to remove redundancy from the following different sources.

- 1) Redundancy between answer and knowledge Base entry
- 2) Redundancy between query and answer
- 3) Cross-sentence answers

For (1) and (2), we remove the answer directly, for example, the answer "*David Lesar*" is removed because "*David J. Lesar*" exists in the knowledge base with the same slot type for the same query; In (3) we keep unique answer candidates for list-value slots, while keep the answer with the highest confidence for single-value slots.

Our redundancy removal methods effectively removed a lot of incorrect answers but also some correct answers for alternative name slots due to step (2). We considered substring matched names not as alternative answers (e.g. "*IBM Corp.*" and "*IBM*"). But this definition is not clear in the 2010 annotation guidelines. In the 2010 assessment guidelines which are used for human annotators, most substring matched names are considered as correct answers.

7 Slot Filling System Combination

We apply a novel statistical re-ranking method to re-rank the IE, QA and pattern pipelines because we can incorporate rich features, and then apply a final priority based combination step to combine with the external knowledge base search pipeline and NYU system (Grishman and Min , 2010).

7.1 Statistical Re-ranking

We develop a Maximum Entropy (MaxEnt) based supervised re-ranking model to re-rank candidate answers for the same slot. We incorporate the following semantics and global statistics based features into the re-ranker:

- Baseline confidence.
- Answer Validation Score.

  This feature provides confidence information based on *q-a* co-occurrence of the query and the answer hypothesis in the source document collection.

- Answer Name Type.

  We incorporate the name type of the candidate answer as an additional feature for re-ranking.

- Slot Type.

  Using the KBP slot type as a feature allows us to re-rank slot types so that our QA system is more likely to get correct answers for a slot type with a higher confidence.

- Dependency Parse Tree of the main context sentence.

- Length of Dependency Parse.

Using the value of each individual system's confidence as the initial probability, we use the feature weights obtained from the trained MaxEnt model to boost the confidence of answers that are more likely to be correct, and decrease the system confidence of those answers more likely to be false. In this way, we can effectively mitigate the impact of errors produced by co-occurrence. For example, the answer "*1976*" did not have a high co-occurrence with the query "*Moro National Liberation Front*", but was bumped up by the re-ranker based on the slot type feature org:founded.

7.2 Priority based Combination

After the above stastical re-ranking, we keep the answers with re-ranking confidence values above a threshold optimized from our development set. However, there may be still some conflicting answers produced from different pipelines or systems. In the KBP2010 evaluation, we submitted two runs which are combined with NYU slot filling system. For each single-value slot, we choose one answer from all of the systems based on the following priority order: External knowledge base search $>$ NYU $>$ IE $>$ QA $>$ Pattern matching pipeline, which is optimized based on the precision values obtained from the development set. For each list-value slot we merge all posible unique answers.

8 Surprise Slot Filling

CUNY-BLENDER team participated the per:disease slot only in the surprise slot filling task.

We first retrieve relevant sentences based on co-occurence of a query (and its coreferential mentions) and most common disease names from a list of common treatment and disease derived from the UMLS, a Metathesaurous of health and biomedical concepts. If the query and the candidate answer co-occured, then we used the same relevance metric employed by the QA pipeline for answer validation against the source collection.

We then apply a filtering scheme based on confliction resolution rules with the organization_founded slot, intervene word length constraint, query/answer order and dependency parsing features.

In general we found that if we only use a smaller list including common disease names, we obtained the same answers as using the entire large list. Most errors were caused by the confusion between per:charity vs. per:disease, and between query:disease vs. family member of query:disease. Most errors we observed could have been avoided by incorporating sentence level analysis. For example, many instances where *q* is a well known or affluent individual and has some done philanthropic work related to a lethal illness or disease, such as *AIDS* or *cancer*. Sentence level analysis can be used to exclude certain event types, require a disease related trigger or other more fine-grained method. extended to at least include those slot answers that were not in our candidate answer set.

Corpus	Source	#mentions		
		PER	ORG	GPE
Training	2009 Training	627	2710	567
	Web data	500	500	500
Testing	Newswire	500	500	500
	Web data	250	250	250

Table 4: Training and Testing for Entity Linking

9 Entity Linking Experiments

9.1 Data and Scoring Metric

For our experiments we use the KBP2010 entity linking evaluation data for testing, and use the entity linking training data released before the evaluation for tuning parameters (as shown in Table 4).

We evaluated the results of our experiments with the official scoring metric in KBP2010 (Micro-average Accuracy).

Query Expansion Strategy	training (Coverage)	testing (Coverage)
step (1)	84.2%	84.9%
(1)+(2)	87.7%	86.7%
(1)+(2)+(3)	92.1%	94.3%
(1)+(2)+(3)+ (3)	95.2%	95.0%
(1)+(2)+(3)+(3)+(5)	96.1%	96.0%
(1)+(2)+(3)+(3)+(5)+(6)	96.9%	96.8%

Table 5: Impact of Query Expansion

9.2 Impact of Query Expansion

In section 3.2, each step from (1) to (6) may introduce more candidates into *Q.candidateset*, and thus may have more chances to obtain the non-NIL answer.

The *Coverage* evaluates the effectiveness of each step in identifying the non-NIL answers. *Coverage* is computed as follows:

$$Coverage = \frac{\#(non - NIL\ answer \in Q.candidateset)}{\#non - NIL\ answer}$$

The experiments were conducted by incrementally adding a step each time. The results on training and testing corpus are shown in Table 5 (experiments for testing corpus were done after KBP2010 keys were released).

Table 5 shows that as we incrementally add steps, the coverage for both training and testing corpus improves. However, we observe that step (3) (redirect page) leads to more gains on the testing corpus than on the training corpus (7.6% gain on testing versus 4.4% training) while step (3) (disambiguation page) brings more gains on training than on testing (3.1% on training versus 0.7% on testing).

By analyzing the query mentions that fail to cover non-NIL answer, we understand that some queries are extremely challenging. Besides some difficult cases discussed in (McNamee and Dang, 2009), we observe two other cases that may need more advanced techniques for resolution:

1. co-reference - the answer is co-referential to the query mention in the context text of *Q*. Without resolving the query mention in *Q.text*, it is hard to find the answer in other means. One example is that the query mention is "*The Lions*"', and the answer is "*Singapore national football team*".'

2. answer is from KB infobox or from KB text. For example, query mention "Angela Dorothea Kasner" is the birth name of the answer "*Angela Merkel*" and it is only mentioned in the KB text rather than redirect page.

VSM Model	Micro-average Accuracy
basic	0.621
enhanced	0.712

Table 6: Impact of Enhanced VSM Model

9.3 Impact of Threshold T in Basic VSM Model

Before the KBP2010 entity linking evaluation, we tuned the threshold in the basic VSM model based on the training corpus. Figure 2 shows that at threshold = 0.1, the basic VSM model achieves the best in micro-average accuracy. We then applied the model on the testing corpus using the "best" threshold of 0.1, and produced the first run which coincidently achieved the same score of 0.621 as on the training corpus. After we got the keys for KBP2010 evaluation, we conducted another set of experiments by tuning the threshold on the testing corpus and observed that the model still achieves the best at the threshold of 0.1 (Figure 2).

Figure 3 shows the performance of NIL and non-NIL on the training and testing corpus respectively. For both training and testing, as the threshold increases, the micro-average accuracy of NIL increases while non-NIL drops. At threshold 0.5, the micro-average accuracy of NIL quickly approaches to 1 while the micro-average accuracy of non-NIL approaches to 0. The overall micro-average accuracy at the tail of the curve on training is 55% which is exactly the distribution of NIL queries in the corpus (in training, NIL queries counts 55% of the total queries) while on testing, it is close to 49%.

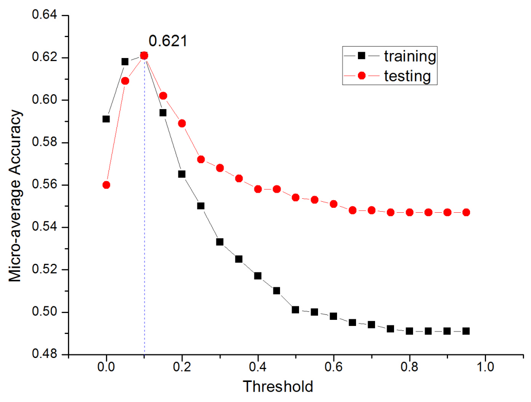

Figure 3: Impact of Threshold in Basic VSM

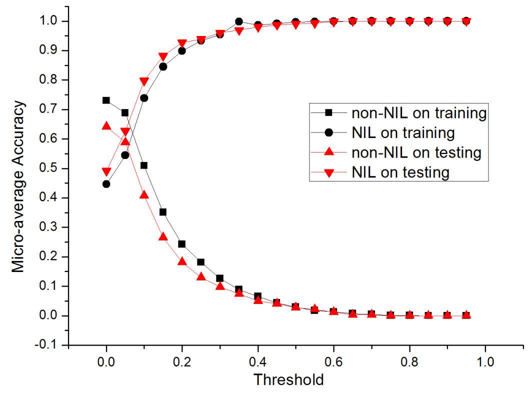

Figure 4: Non-NIL versus NIL

9.4 Impact of Enhanced VSM Model

We re-run the experiments after KBP2010 evaluation since we had made some minor changes in slot filling system which potentially improves the extracted results. Table 6 shows that enhanced model achieves a little higher score than the one submitted to KBP2010.

Since the slot filling system only extracts attributes for PERSON and ORGANIZATION, the gain comes from the candidate disambiguation of persons and organizations.

We observe that some attributes are particularly helpful for candidate disambiguation, for example, "title" and "residence" attributes, partly because our slot filling system can extract such attributes with higher accuracy.

10 Slot Filling Experiments

In this section we present the overall performance of our individual slot filling pipelines and break down the performance to demonstrate the impact of key techniques.

10.1 Data and Scoring Metric

We randomly select 50 queries (25 persons and 25 organizations) and the entire source collection from KBP 2010 training corpora as a development set to evaluate our methods. And we will also measure the impact of some approaches on the KBP2010 evaluation set.

However, as we will demonstrate later in section 10.4, a single human annotator can only achieve lower than 50% recall. Therefore we followed the human assessment procedure as in KBP evaluation to expand the answer key of the development set. We asked Amazon Mechanic Turk and human annotators to manually check all those instances generated by the system but not in human annotation. If an instance is correct, it will be added to form the expanded answer key set. The final answer key for these 50 queries include 711 non-NIL answers.

We follow the KBP 2010 scoring metric to evaluate each pipeline but ignoring document ID comparison. This is a uniform scoring metric based on standard Precision, Recall and F-measure. A unique answer instance {query, slot type, answer} is considered as correct if it matches any instance in the answer key. We added additional answer normalizations to the scorer in order to get more reliable scores and speed up human assessment (normalized 6% instances in the test set). The normalizations are based on a list of 362 country name variants (e.g. *the United States* = *USA*) and a list of 423 person fullname-nickname pairs.

10.2 Top-down vs. Bottom-up

Figure 5 presents the breakdown scores of three individual pipelines for each slot type.

Although different slots obtained quite different comparison results, overall pattern matching achieved the best result. Supervised IE method performs the worst because not all of the slot types have corresponding relation and event types. QA method

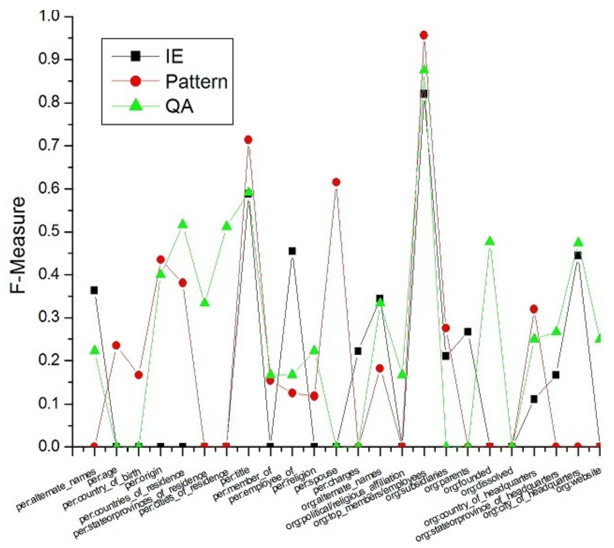

Figure 5: Top-down and Bottom-up Pipeline Comparison on Each Slot Type

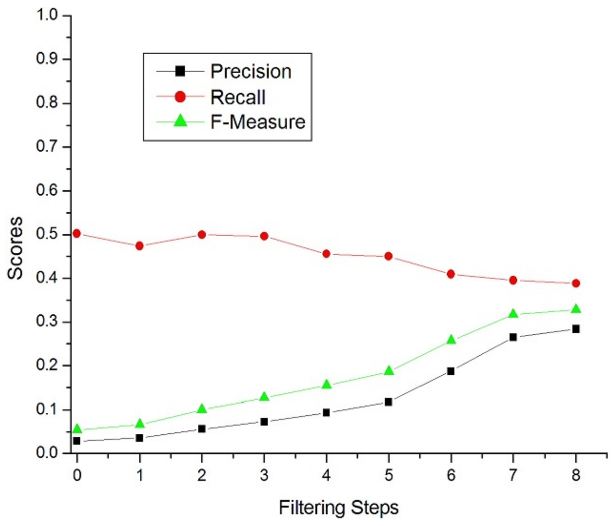

Figure 6: Impact of Answer Filtering on Pattern Matching Pipeline

obtained comparable precision but much lower recall because the candidate answers are restricted by query template design and the annotations (e.g. name tagging) used for answer validation.

10.3 Impact of Answer Filtering

Figure 6 demonstrates how the performance got improved as we added more filtering steps to the pattern matching pipeline. We can see that as we add the filtering steps, we significantly enhanced precision (25.5%) with some loss in recall (11.4%), and eventually obtained 27.4% absolute improvement on f-measure.

10.4 Impact of Reasoning

Experimental results demonstrate the cross-slot and cross-system reasoning approach can enhance the quality of slot filling in two aspects: (1) It can generate new results for the slots which the pipelines failed altogether; (2) It can filter out or correct logically incoherent answers. Table 7 presents the number of unique answers removed or added for the evaluation set. Occasionally this approach removed some correct answers, for example, in the sentence "*She had two daughters with one of the MK'd Westlife singers, Brian McFadden, calling them M olly M arie and Lilly Sue*", our reasoning approach mistakenly removed "*singers*" from the per:title answer for the query "*Brian McFadden*" because "singers" is not in our title list. However, we can see that overall this approach can significantly improve both precision and recall.

Operations	Total	Correct	Incorrect
Removal	277	244	33
Adding	16	16	0

Table 7: Impact of Cross-Slot and Cross-System Reasoning

10.5 Impact of Statistical Re-ranking

We evaluate the impact of re-ranking on . The relative Precision (P), Recall (R) and F-Measure (F) scores are presented in Table 8. We can see that statistical re-ranking significantly improved each pipeline and outperformed a simple combination approach based on pipeline priorities. Supervised Re-Ranking helps to mitigate the impact of errors produced by scoring based on co-occurrence. For example, when applied to our answer set, the answer *Clinton* for the query *Dee Dee Myers*, had a system relevance score for the attribute per:children due to frequent co-occurrence that was reduced and consequently removed by re-ranking. Alternatively, the query *Moro National Liberation Front* and answer *1976* did not have a high co-occurrence in the text collection, but was bumped up by the re-ranker based on the slot type feature org:founded .

Methods	P	R	F
Feature-based IE	24.2	14.2	17.9
Pattern Matching	21.9	37.7	27.7
Question-Answering	26.7	17.3	21.0
Priority-based Combination	30.5	26.6	28.4
Re-ranking	28.0	44.3	34.3

Table 8: Impact of Statistical Re-Ranking on Individual Pipelines (%)

10.6 Impact of Using External Knowledge

We have exploited external knowledge base in a more direct way than distant learning, namely searching and validating the answers in the source collection. Table 9 shows that using Freebase and Wikipedia text mining can achieve slight gains (1.3% F-Measure).

Using External KB	P	R	F
No	27.99	26.02	26.97
Yes	28.74	27.85	28.29

Table 9: Impact of External Knowledge Base (%)

11 Related Work

The entity linking task involves aligning a textual mention of a named-entity to an appropriate entry in a knowledge base, which may or may not contain the entity. There are two major difficulties in this task:

1. Name variation: entities can be referred to by multiple name variants (e.g., aliases, acronyms, misspellings). Entity linking must find an entry despite changes in the mention form.

2. Name ambiguity: different entities can share the same name, i.e., a single mention can match multiple entries in the KB. Entity linking must disambiguate name entities and find the most appropriate entry for the mention.

To alleviate the two difficulties, a typical entity linking system contains two crucial components:

1. Query expansion and candidate generation: The aim of query expansion is to produce an expanded query set including name variations and its ambiguous forms given a query. The aim of candidate selection is to produce a *sufficient* and *manageable* candidate list which means that too restrictive selection may miss a promising candidate but too loose selection may introduce complexity and uncertainty in the second component as described below.

2. Candidate disambiguation: this component is to rank the KB nodes in the candidate set and link the mention to the most probable one. However, if the entity the mention refers to is a new entity not present in the KB, NIL should be returned.

For the first component, up-to-date techniques include:

1. Building a Knowledge Repository based on Wikipedia which provides vast amount of world knowledge of entities (Zhang et al., 2010), e.g., redirect page, disambiguation page, anchor text, “did you mean”.

2. Developing heuristic algorithms to identify the acronym and its expanded form from the context text of the query (Varma et al., 2010).

3. Developing metaphone algorithms to identify spelling variations.

For the second component, the approaches may vary from unsupervised learning to supervised learning. Many unsupervised approaches rank the candidates based on VSM model which computes similarity between mention and candidate KB nodes. However, it is hard to tune the threshold for identifying NIL from non-NIL. Even with some annotated data, the threshold is till questionable since it is hard to know whether the testing queries shares the same data distribution with the training queries. Supervised approaches include SVM classification (Zhang et al., 2010), SVM ranking (Dredze et al., 2010) and ListNet (Zheng et al., 2010), all of which lead to reasonable good performance. However, they must utilize sufficient training data in which each instance is a pair of query and KB node (or NIL).

Answer validation and re-ranking has been crucial to enhance QA performance (e.g. (Magnini et al., 2002); (Peas et al., 2007)). Recent work (Ravichandran et al., 2003) has showed that high performance for QA systems can be achieved using as few as four features in re-ranking. Our results on the QA pipeline support this finding. The same related work (Huang et al., 2009) reports that systems viewed as a re-ranker work clearly outperforms classifier based approaches, suggesting a re-ranking was a bet-ter implementation choice.

(Bikel et al., 2009) designed inference rules to improve the performance of slot filling. We followed their idea but incorporated inference rules into Markov Logic Networks (MLN).

12 Conclusions

In this paper, we described two crucial components in CUNY_BLENDER entity linking system: one component for query expansion and candidate generation, and the other component for candidate disambiguation. We also implemented two approaches for candidate disambiguation. Especially for the enhanced VSM model, we incorporate the attributes extracted from slot filling. Our experiments show that the enhanced VSM model performs better than the BOW model.

We developed three effective pipelines for the slot filling task. We enhanced the performance of slot fillingusing several novel approaches including statistical answer re-ranking and cross-slot reason-ing based on Markov Logic Networks. Furthermore we conducted combinations across human annotators and systems, and proposed an effective approach to enrich an-swer keys effectively and efficiently. Our results showed that adding systems which used new re-sources and achieved good performance can generate more correct answers than simply adding human annotators.

Acknowledgments

This work was supported by the U.S. Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053, the U.S. NSF CAREER Award under Grant IIS-0953149, Google, Inc., DARPA GALE Program, CUNY Research Enhancement Program, PSC-CUNY Research Program, Faculty Publication Program and GRTI Program. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

Dan Bikel, Vittorio Castelli, Radu Florian and Ding-jung Han. 2009. Entity Linking and Slot Filling through Statistical Processing and Inference Rules. *Proc. TAC 2009 Workshop*.

K. Bollacker, R. Cook and P. Tufts. 2008. Freebase: A Shared Database of Structured General Human Knowledge. *Proc. Proc. National Conference on Artificial Intelligence*, (Volume 2).

M. Dredze, P. McNamee, D. Rao, A. Gerber and T. Finin. 2010. Entity Disambiguation for Knowledge Base Population. *Proc. COLING 2010*.

Ralph Grishman, David Westbrook and Adam Meyers. 2005. NYU's English ACE 2005 System Description. *Proc. ACE 2005 Workshop*.

Ralph Grishman and Bonan Min. 2010. New York University KBP 2010 Slot-Filling System. *Proc. TAC 2010 Workshop*.

Z.Huang and M. Thint and A. Celikyilmaz. 2009. Investigation of question classifier in question answering. *Proc. ACL 2009*.

Heng Ji and Ralph Grishman. 2008. Refining Event Extraction Through Cross-document Inference. *Proc. ACL 2008*.

S. Kok, M. Sumner, M. Richardson, P. Singla, H. Poon, D. Lowd, and P. Domingos. 2007. The Alchemy system for statistical relational AI. *Technical report, Department of Computer Science and Engineering, University of Washington.*.

Fangtao Li, Zhicheng Zheng, Fan Bu, Yang Tang, Xiaoyan Zhu and Minlie Huang. 2009. THU QUANTA at TAC 2009 KBP and RTE Track. *Proc. TAC 2009 Workshop*.

Bernardo Magnini, Matteo Negri, Roberto Prevete and Hristo Tanev. 2002. Mining Knowledge from Repeated Co-occurrences: DIOGENE at TREC-2002. *Proc. TREC-2002*.

P. McNamee and H. Dang. 2009. Overview of the TAC 2009 Knowledge Base Population Track. *Proc. NIST TAC 2009 Workshop*.

Anselmo Peas, lvaro Rodrigo and Felisa Verdejo. 2007. Overview of the Answer Validation Exercise 2007. *Working Notes of CLEF 2007*.

D. Ravichandran, E. Hovy and F. J. Och. 2003. Statistical QA -classifier vs. re-ranker: what's the difference?.

Proc. ACL 2003 Workshop on Multilingual Summarization and Question Answering.

Matt Richardson and Pedro Domingos. 2006. Markov Logic Networks. *Machine Learning*, 62:107-136.

Nico Schlaefer, Jeongwoo Ko, Justin Betteridge, Guido Sautter, Manas Pathak, Eric Nyberg. 2007. Semantic Extensions of the Ephyra QA System for TREC 2007. *Proc. TREC 2007*.

V. Varma,V. Bharat,S. Kovelamudi, P. Bysani, S. GSK, K. K. N, K. Reddy, K. Kumar, N. Maganti. 2009. IIIT Hyderabad at TAC 2009. *Proc. NIST TAC 2009 Workshop*.

W. Zhang, J. Su, C. L. Tan and W.T. Wang. 2010. Entity Linking Leveraging Automatically Generated Annotation. *Proc. COLING 2010*.

Z. Zheng, F. Li, M. Huang, X. Zhu. 2010. Learning to Link Entities with Knowledge Base. *Proc. Proc. HLT-NAACL2010*.