# Domino: SAIC's English Entity-Linking System

**Alan Buabuchachart[1,2], Parakh Jain[1,2], Ryan Murphy[1,2], Scott White[1,2], and Leora Morgenstern[1]**

**[1]SAIC**
**4001 North Fairfax Drive**
**Arlington, VA 22203**

**[2]University of Maryland at College Park**
**College Park, MD 20742**
alan2234637@hotmail.com, jainparakh@gmail.com,
murphy.t.ryan@gmail.com, whiteau@umd.edu, leora.morgenstern@saic.com

**Abstract:**
The Domino system was SAIC's student-intern entry to the English Entity-Linking track of the 2012 TAC-KBP competition. This paper describes how Domino was developed using components from the CUNY-BLENDER system and discusses the features and rules that were added to Domino. It analyzes Domino's performance, and suggests ways in which we plan to improve the system in the future.

## 1.Building the Domino Baseline System

### 1.1 Motivation and Constraints

Entity linking is a central task that analysts in the intelligence community (IC) often perform. Analysts must try to determine, for example, whether a person who is referred to in an intercepted email is the same as a person who is reported in some news article to have engaged in some terrorist activity. SAIC, which supports IC analysts in many different ways, is interested in developing methods to help automate the entity-linking process.

There are many similarities between the IC entity-linking task and the TAC-KBP Entity-Linking track, which focuses on linking named entities in news articles or blog posts with Wikipedia articles. We --- a group of students from the University of Maryland who spent the summer of 2012 at SAIC, and our supervisor at SAIC --- therefore decided, in mid-June 2012, to enter the TAC-KBP Entity Linking competition.

Because most of us --- and in particular, the developers among us –were undergraduates with little experience in Natural Language Processing, and because we knew we had only two months to pull together a system, we decided that we would try to use existing resources as much as possible. Our aim was to get an existing entity-linking system and modify it in order to improve output results. We were especially interested in generalizing the entity-linking system so that it would be useful for more than just Wikipedia's domain. SAIC's customers will often be interested in people who keep a low profile and who would be unlikely to have an entry in Wikipedia.

### 1.2 Harnessing Existing Resources

We were fortunate that the researchers who had developed CUNY-BLENDER, the CUNY's entry to multiple TAC-KBP tracks (entity linking and slot filling) in 2010 [Chen et al., 2010] had made CUNY-BLENDER's codebase available to anyone who wanted to use it. We decided to build our system on top of the CUNY-BLENDER pipeline.

Originally, we had envisioned that DOMINO would be a superset of CUNY-BLENDER. We had hoped to quickly get CUNY-BLENDER running, establish a baseline, and then spend most of our time experimenting with new features which would enhance Domino's performance. In fact, we needed to make significant adjustments and modifications to CUNY-BLENDER. As a result, while Domino is

based on CUNY-BLENDER, it is neither a subset nor a superset of it.

### 1.3 Overview of CUNY-BLENDER

The architecture for the CUNY-BLENDER pipeline is shown below.

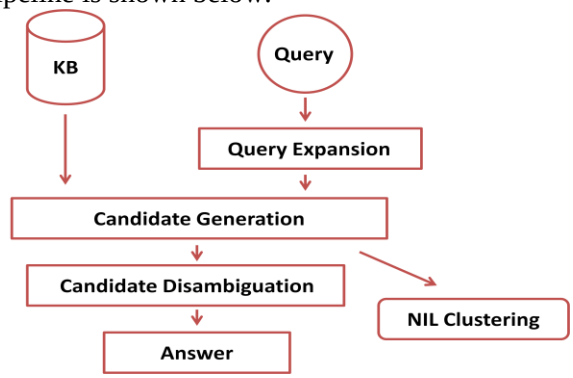


*Figure 1: CUNY-BLENDER'S architecture*

The general procedure for entity linking is as follows:

*Preprocessing:*

1) The source news articles are indexed using Lucene.
2) The Knowledge Base (the information in Wikipedia infoboxes) is indexed using Lucene
3) The Wikipedia dump is imported into a MySQL database

*Run-time steps:*

4) Given a query, perform Query Expansion
5) Given the expanded query, perform Candidate Generation: that is, determine which possible Wikipedia entities might match the queried entity
6) Given a query and its list of generated candidates, perform Candidate Disambiguation: that is, figure out which of these possible candidates is the best match
7) If the Disambiguation process returns NIL as the answer --- that is, none of the candidates can be linked with sufficiently high confidence to the query entity --- move the query to the list of NILs which will be NIL clustered
8) Otherwise, link the query with the answer

### 1.4 Domino's Modifications of BLENDER

To develop the baseline Domino system, we made the following changes to the CUNY-BLENDER system:

- Since we entered only the Entity Linking track and not the Slot Filling task, we made the entity-linking system separate from the slot-filling system. This simplified the code-base.

- Because a new data format was used in this year's conference, an additional step was added to the Preprocessing stage that converted the new format of the data back to the original format. Due to time constraints, it was simpler to convert the data than to change the entire Preprocessing stage to work with the new format. Offsets were a new addition to the queries this year, and we did not have enough samples to take them into account when modifying the system.

- Lucene's spell checker was removed from the Query Expansion stage because it was generating extra false positives, thus decreasing the precision and F-score.

- Unlike the original system, which relied on the collaborative ranking of five different rankers, our system only used SVM. We realized that we probably would be able to incorporate only one of the rankers because of our time limitations. We chose SVM because it has proven to be very effective (Chen and Ji, 2011; Zhang et al, 2010). Moreover, since it is the most commonly used ML algorithm in KBP, it is more reliable compared to other rankers.

- Instead of CUNY's proprietary IE Toolkit, our system used Stanford's NER to classify the types of the query and candidates.

- We used an updated dump of Wikipedia (June 2012) instead of the October 2008 dump that was originally used in CUNY-BLENDER's system.

- In order to improve the run time of the system, the SVM classifiers were saved after the training of the data. This enabled us to easily compare classifiers with different features and training sizes.

### 1.5 Domino's Baseline Performance.

To get a baseline performance for Domino, we trained on 2009 and 2010 data and tested on 2011 data. Our measured baseline performance was:

Precision: .67

Recall: .69

F1 .68

## 2. Improving the baseline score

### 2.1 Strategy

It had taken over a month to get the baseline system working. With just a few short weeks remaining before the Entity-Linking competition, we needed to figure out a way to improve performance as quickly as possible. Our strategy was twofold:

**First**, we examined output data and tried to get a sense for what the largest sources of error were. Could we spot any patterns among the false negatives and false positives? Based on these patterns, could we think of any features that could be added and trained on, or rules that could be added, to potentially improve performance?

**Second**, we were focused primarily on techniques that would work well on data that intelligence analysts would frequently come across. For an example of such a technique, see the discussion of Metaphone below. With limited time, we decided not to focus on features that would be limited to data similar to Wikipedia.

### 2.2 Improving Candidate Generation

**Metaphone:** When examining the output, we noticed that for some queries, the correct candidate would not be generated by baseline Domino because a named entity was misspelled in the source document (news article or blog post). We realized that this phenomenon could occur also when a name with one pronunciation had many spellings. For example, the names Kerry, Cary, Carrie, and Kari have different spellings but identical pronunciations (for most speakers). Especially for foreign names that originate in languages for which there are no standard transliteration schemes, it would be helpful to consider all possible spellings for a particular string of phonemes.

While we realized that this would not yield a large improvement in our F score for the TAC-KBP competition, because there were not that many examples of this phenomenon in the data, we were interested in pursuing this direction because SAIC's customers have told us that intelligence analysts frequently do have to deal with this phenomenon. We chose to use the Metaphone [Philips, 1990] representation to handle this problem, since it would allow the system to consider phonetically-equivalent but differently-spelled words as equivalent.

Metaphone representations were added using the DoubleMetaphone class in Apache Commons to take alternate spellings into account as potential candidates. Metaphones were actually used twice, once in the Query Expansion stage and once in the Candidate Generation stage. Apache Lucene was first used to determine queries that were spelled similarly to the original query. The metaphone representation was then applied on each of the similar queries, and if it matched the metaphone representation of the original query, it was added to the QuerySet.

Candidates were then generated based off the QuerySet. In addition to the Candidate Generation steps that already existed in the original CUNY-BLENDER pipeline, we added an additional step that included candidates that also had an identical metaphone to one or more of the queries in the QuerySet. Furthermore, we manipulated each of the queries in the query set to find similar metaphone representations that may differ due to potential "silent letters". For example, not only do we compare the metaphone representation of "Djokovic" to those of the candidates, we also compare the metaphone representation of "Jokovic" because "D" is a silent character. A list of all such mappings

follows:
{mb → m; bt → t; dj → j; gm → m; gn → n; gh → g; kn → n}

**Removing Periods From Initials:** The last modification made in the Candidate Generation stage was the removal of periods from the queries within the QuerySet that have initials. For example, candidates for "A. K. Antony" would be determined based off the name "A K Antony". The reason for this modification was because names represented in MySQL do not contain periods. This therefore improved the retrieval accuracy.

### 2.3 Improving Candidate Disambiguation

As previously discussed, we used SVM as our ranker. Its implementation using LibSVM, remained the same as for CUNY-BLENDER. We added several different features to SVM, two of which are query type matching and candidate name frequency

**Query Matching:** Query type matching is a straightforward binary feature that given a query and a candidate returns 1 if they exhibit the same type (PER, ORG, or GPE), and 0 otherwise.

**Candidate Name Frequency:** Candidate name frequency is a feature that measures how often each of the candidates' names appears in a given query's text. We disregarded the following noise words: *a, an, and, as, for, I, in, is, it, of, on, that, the, this, to, was, with*. Additionally, we rewarded the first match with the largest point value, and successive matches with increasingly smaller point value. Successive matches were separated into two categories: *same word*, referring to the number of times each word from a candidate's name appear in the query's text, and *different word*, referring to the current number of words that appear at least once in the query's text.

Let *m1* be *same word*, *m2* be *different word*, and *l* be the number of words in the name (counting noise words). Then the base score of a given word can be computed as follows:

$$base = 1 - 0.2(1 - m2/l)$$

Let *cs* be the current score of a given word (*cs* = base). Then the final score of the current word can be computed as follows:

$$for\ 1\ to\ m1\text{: } cs = cs + 0.25(1 - cs)$$

Let *n* be the number of words in the name (not counting noise words). Then the final overall score of a given candidate name can be computed as follows:

$$finalScore = (cs_1 + cs_2 + cs_3 + \ldots + cs_n) / n$$

The value of this feature thus ranges from 0.0 to 1.0.

**Aiding Inference for GPE Queries:** We noticed in the output data that it frequently happened that obvious candidates would be discarded in favor of incorrect candidates. For example, Bakersfield can refer to a city in California, Missouri, Vermont, or Texas. These will typically all be generated by Candidate Generation. However, Domino would often choose the wrong candidate, even if the state was explicitly mentioned in the source document. We therefore added a rule to explicitly check for a candidate GPE's name in the source text. The rule was implemented as follows:

If a given query is a GPE, then check the name of its candidates. If the candidate name appears in the query's text, then pick that candidate as our final choice. Otherwise, if none of the candidate names appear in the query's text, then follow the normal procedure and use SVM to infer the choice.

We also expanded the candidate names by including abbreviations and alternate spellings. For example, CA, Cal., Calif., and California are all recognized as alternate spellings of the same entity.

### 2.4 Results of Enhanced System on Test Data

The combination of the enhancements described above took Domino up from an F score of .68 to an F score of .76 when testing on 2011 data. The following table summarizes how each successive enhancement improved the F score:

| | Precision | Recall | F1 |
|---|---|---|---|
| Baseline | .67 | .69 | .68 |
| B+ query matching | .69 | .70 | .695 |
| B+QM + Metaphone | .70 | .71 | .705 |
| B+QM+MP+ GPE rule | .748 | .76 | .754 |
| B+QM+MP+GPE rule + Candidate Name Frequency | .755 | .767 | .761 |

*Figure 2: Domino's performance: baseline and with several enhancements.*

### 2.5 Features that Didn't Work

There were several features that we tried that did not enhance performance. These included considering hyphenation, the presence of a candidate name (and its expanded name) in a query's text (similar to candidate name frequency, but 0 or 1 instead of real values); features combination (if GPE and tfidf < 0.1 and CNF > 0.8; Dictionary/Spellchecking/"did you mean..." feature of Lucene (during the Query Expansion step)

.

## 3. Results for 2012 Entity Linking Competition

The training data of KBP 2012 for entity linking consisted of 3,904 queries in 2009 Eval-set, 1,500 queries in 2010 Training-set, 2,250 queries in 2010 Eval-set, and 2,250 queries in 2011 Eval-set. Table 2 illustrates the distribution of the training data.

| Genre/Source | Size (entity mentions) | | |
|---|---|---|---|
| | Person | Org. | GPE |
| 2009 Eval | 627 | 2710 | 567 |
| 2010 Training Web data | 500 | 500 | 500 |
| 2010 Eval Newswire | 500 | 500 | 500 |
| 2010 Eval Web data | 250 | 250 | 250 |
| 2011 Eval Newswire | 500 | 491 | 500 |
| 2011 Eval Web data | 250 | 259 | 250 |

*Figure 3: English monolingual entity linking training data*

Domino submitted three results to KBP 2012. Domino1, Domino2, and Domino3 used SVM thresholds of 0.05, 0.10, and 0.50 respectively. The results are reported in Table 3. The scoring metric used in KBP 2012 to evaluate entity linking system is *B-Cubed*$^+$.

| System | Precision | Recall | F1 (in KB) | F1 (All) |
|---|---|---|---|---|
| Domino1 | 0.448 | 0.588 | 0.557 | 0.509 |
| Domion2 | 0.440 | 0.604 | 0.523 | 0.509 |
| Domino3 | 0.416 | 0.633 | 0.499 | 0.433 |
| Highest submission | - | - | 0.687 | 0.730 |
| Median submission | - | - | 0.496 | 0.536 |

*Figure 4: Entity Linking submission scores*

## 4. Preliminary Diagnostics

Although our F1 score was a little below the median, upon closer inspection we see that our F1 score in KB was actually above the median. This implies that the NIL Clustering portion of our system was underperforming. This is an area in which we plan to improve in the future. Due to our limited time, we were not able to write our own NIL clustering class. Instead, we opted to use the one that came with the original system. Specifically, the original system clusters NIL queries by grouping NIL queries with the same names together. This works well as long as the names are not ambiguous, but that assumption may not have been true of 2012 data.

In terms of specific query types, most teams do well in Person queries but suffer from a downgrade in performance in GPE. For Domino, the trend was the opposite. We did better in GPE queries than Person queries, perhaps due to our special-purpose GPE rule. Figure 5 illustrates F1 scores distribution among the three different queries types.

| System | PER | ORG | GPE |
|---|---|---|---|
| Domino1 | 0.467 | 0.507 | 0.560 |
| Domino2 | 0.471 | 0.524 | 0.532 |
| Domino3 | 0.450 | 0.539 | 0.494 |
| Highest | 0.840 | 0.717 | 0.694 |
| Median | 0.646 | 0.486 | 0.447 |

*Figure 5: Distribution of F1 scores among different query types*

## 5. Future Work

We plan to improve Domino in the following ways:

- Improve candidate name frequency by looking at nearby terms rather than through the whole document
- Come up with new and improved ways to handle incorrect and alternate spellings
- Use a gazetteer with superset and region information to disambiguate. For example if you see an article about Paris, and the article also mentions Marseille and Lyons, you'd conclude that the article is about Paris, France rather than Paris, TX since Marseille and Lyons are in the same region as Paris.
- Use slot-filler information to disambiguate.

**Acknowledgements:**
We gratefully acknowledge the support of the Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract number FA8750-09-C-0184.

## References:

Zheng Chen and Heng Ji. 2011. Collaborative Ranking: A Case Study on Entity Linking. *Proc. EMNLP2011*.

Zheng Chen, Suzanne Tamang, Adam Lee, Xiang Li, Wen-Pin Lin, Matthew Snover, Javier Artiles, Marissa Passantino, Heng Ji: CUNY-BLENDER TAC-KBP201: Entity Linking and Slot Filling System Description, *Proceedings TAC-KBP Workshop, 2010*.

Philips, Lawrence: Hanging on the Metaphone, *Computer Language*, 7(12), 1990.

Wei Zhang, Jian Su, Chew Lim Tan, and Wen Ting Wang. 2010. Entity Linking Leveraging Automatically Generated Annotation. *Proc. COLING2010.*