CUNY_BLENDER TAC-KBP2011 Temporal Slot Filling System Description

Javier Artiles, Qi Li, Taylor Cassidy, Suzanne Tamang and Heng Ji
Computer Science Department and Linguistics Department
Queens College and Graduate Center
City University of New York
New York, NY 11367, USA
{javart, hengjicuny}@gmail.com

Abstract

In this paper we describe the participation of the CUNY-BLENDER team in the Temporal Slot Filling (TSF) pilot task organized as part of the TAC-KBP2010 evaluation. Our team submitted results for both the "diagnostic" and "full" TSF subtasks, obtaining the top score in the diagnostic subtask.

We implemented a "structured" and a "flat" approach to the classification of temporal expressions. The structured approach captures long syntactic contexts surrounding the query entity, slot fill and temporal expression using a dependency path kernel tailored to this task. The flat approach exploits information such as the lexical context and shallow dependency features.

In order to provide enough training data for these classifiers we used a distant supervision approach to automatically generate a large amount of training instances from the Web. This data was further refined by applying logistic regression models for instance relabeling and feature selection methods.

1 Introduction

This paper presents the CUNY-BLENDER participation in the KBP2011 Temporal Slot Filling (TSF) pilot task (Ji et al., 2011).

Our approach to the TSF task was to reformulate it as two problems: the classification of temporal expressions and the aggregation of the resulting temporal information. Classification is applied to identify the role of temporal expressions that appear in the context of a particular entity and attribute value. For instance, in "Harry married Sally in 1995" a classifier should determine that "1995" indicates the beginning of the attribute *spouse*. Given the output of this classification the TSF system proceeds to aggregate the available temporal information and provide a final answer. We developed and tested two approaches to the temporal classification problem: a structured approach and a flat approach. The structured approach captures long syntactic contexts surrounding the query entity, slot fill and temporal expression using a dependency path kernel tailored to this task. The flat approach exploits surface lexical context and shallow dependency features. For the aggregation of the temporal information we rely on a simple but effective iterative algorithm.

Given the expensive nature of human-assessed training data for this task we used distant supervision to acquire large amounts of annotated data from the Web without human intervention. We explored the reduction of the feature space to speed up training and eliminate noisy or unnecessary features. Additionally we tested the impact of relabeling training instances based a small set of hand labeled data.

The rest of this paper is structured as follows. Section 2 briefly summarizes the task definition and scoring metric. The different components of our TSF system are described in Section 3. In Section 4 we present the distant supervision approach used to obtain training data for the classification of temporal expressions. In Section 5 we include the results obtained on the KBP2011 TSF test data. Related work is described in Section 7. Finally we provide conclusions and future work plans in Section 8.

2 Task Definition

The TSF task is best characterized as an extension of the existing KBP regular Slot Filling task. Slot Filling aims at, given an entity and a large document collection, extracting values for attributes such as *employee*, *spouse*, *member*, etc. The TSF task focuses on the subset of these attributes whose value may change over time. Systems take as input an entity, slot type and slot value as well as the source document where the slot value was found. In the "diagnostic" subtask correct slot values were provided, while in the "full" subtask participants were required to run their own Slot Filling system. The output expected from the systems is a start/end date for each entity/attribute pair.

The KBP2011 temporal representation model consists of a 4-tuple whose elements are dates (day, month and year), $< t_1, t_2, t_3, t_4 >$. A tuple represents the set of possible beginnings and endings of an event. t_1 and t_3 represent the lower and upper bounds, respectively, for the beginning of the event, while t_2 and t_4 represent the lower and upper bounds for end of the event. This allows the representation of different temporal granularities. For instance one might only know that an event began on a certain year, and in that case t_1 will be set to the first day of that year and t_2 to the last day.

Given an entity name *Jose Padilha*, its slot fill *Film Maker* for the slot type *per:title*, a diagnostic temporal slot filling system may discover a temporal tuple $< -\infty, 2007-12-26, 2007-12-26, +\infty >$ to represent the temporal boundaries.

The official scoring metric $Q(S)$ for the task compares a system's output $S =< t_1, t_2, t_3, t_4 >$ against a gold standard tuple $S_g =< g_1, g_2, g_3, g_4 >$, based on the absolute distances between t_i and g_i:

$$Q(S) = \frac{1}{4} \sum_i \frac{1}{1 + |t_i - g_i|}$$

When there is no constraint on t_1 or t_3 a value of -∞ is assigned; similarly a value of +∞ is assigned to an unconstrained t_3 or t_4.

Let $\{G^1, G^2, ..., G^N\}$ be the set of gold standard tuples, $\{S^1, S^2, ..., S^M\}$ the set of system output tuples. For each unique slot fill i, there is the 4-tuple $G^i := < g_1, g_2, g_3, g_4 >$, and $S^j := < t_1, t_2, t_3, t_4 >$. Then Precision, Recall and F-measure scores are calculated as follows:

$$Precision = \frac{\sum_{S^i \in C(S)} Q(S^i)}{M}$$

$$Recall = \frac{\sum_{S^i \in C(S)} Q(S^i)}{N}$$

$$F_1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

Where $C(S)$ is the set of all instances in system output which have correct slot filling answers, and $Q(S)$ is the quality value of S. In the diagnostic task, precision, recall, and F_1 values are the same since we are provided with correct slot filling values as part of the system input.

3 System Overview

In Figure 1 we summarize our system pipeline. Each relevant source document is fully processed using the NLP Core Stanford toolkit (Finkel et al., 2005) to tokenize, segment sentences, detect named entities, build a coreference chain and analyze the syntactic dependencies within sentences.

Note that in the diagnostic TSF subtask slot values and their corresponding source documents are provided by the organizers and are known to be correct. The full TSF subtask, on the other hand, requires participants to run their own Slot Filling (SF) system to obtain the slot values associated with each entity in the KBP source document collection. In the full task we search the source collection using each query name and its slot value to find documents related to the query in addition to those that support the SF output. This set of related documents is augmented using A Lucene index to search for the top 10 most relevant documents in the KBP source collection containing each entity/slot fill pair found by our SF system (Chen et al., 2010).

The first application of this annotation is to find sentences that mention both the entity and the slot value. String matching only provides very limited coverage and so we use named entity recognition and coreference results to expand this set of relevant sentences. We apply those coreference chains that contain the provided slot value or entity name to select sentences that mention both.

Our next step is to represent each temporal expression in the context of the entity and slot value as a classification instance. For example, the following

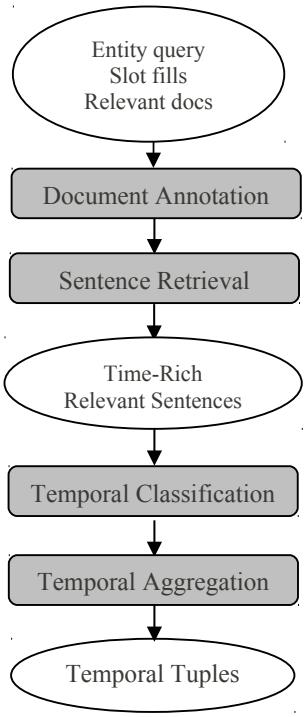

Figure 1: General Temporal Slot Filling System Architecture

sentence contains two temporal expressions, which will result in two different classification instances.

> [Moore$_{entity}$] married [Grant Tinker$_{spouse_attribute}$], an NBC executive, in [1962$_{date}$], and in [1970$_{date}$] they formed the television production company MTM Enterprises, which created and produced the company 's first television series, The Mary Tyler Moore Show .

Sentence are normalized before extracting each token and token ngrams (lengths 2 to 5) as bag-of-words (BoW) instance features. We apply the following rules for sentence normalization:

- Named entities are replaced by a generic string according to their NE type (ORGANIZATION, PERSON, etc)
- A gazetteer of occupation titles (president, CEO, chief engineer, ...) is used to replace matching strings by a generic string "TITLE".
- Mentions of the entity are replaced by "TARGET ENTITY". We consider both direct string matches as well as coreferential mentions. The same applies to the slot value ("TARGET ATTRIBUTE").
- Temporal expressions are substituted by the generic "DATE" string, except for the expression that is considered for classification in the particular instance, which is distinguished by the string "TARGET DATE".

Continuing with our previous example, the normalized sentence focused on the temporal expression '1962' would look as follows:

> TARGET_ENTITY married TARGET_ATTRIBUTE , an ORGANIZATION TITLE , in TARGET_DATE , and in DATE they formed the television production company ORGANIZATION , which created and produced the company 's ORDINAL television series , the MISC .

We can see how this set of features captures short distance patterns (e.g. the trigram "TARGET ENTITY married TARGET ATTRIBUTE").

Our second set of features is intended to cope with longer distance relations within the sentence. These features include all of the syntactic dependencies found in the sentence while keeping the same type of normalization that we applied for bag-of-words (BoW) features.

3.1 Temporal expression classification

Classification is applied to label temporal expressions that appear in the context of a particular entity and the slot value as 'start', 'end', 'holds', 'range' or 'none'. In order to simplify the generation of training data and the classification model itself, we ignored certain cases. For instance, we ignored cases where the temporal expression indicates a time before or after the event (these cases were just considered 'unrelated') such as "In 1996, five years before joining Microsoft, John submitted his first patent".

Suppose our query entity is *Smith*, the slot type is *per:title*, and the slot-fill *Chairman*. Below we provide a description of each class along with its corresponding 4-tuple representation:

START $< t_a, t_b, t_a, \infty >$

The temporal expression describes the beginning of the slot fill.

E.g. *Smith, who was named chairman two years ago*[1]

$<$ 1999-01-01, 1999-01-01, 1999-01-01, $\infty >$

END $< -\infty, t_b, t_a, t_b >$

The temporal expression describes the end of the slot.

E.g. *Smith, who resigned last October*

$< -\infty,$ 2000-10-01, 2000-10-01, 2000-10-31 $>$

HOLDS $< -\infty, t_b, t_a, \infty >$

The temporal expression describes a time at which the slot fill is valid.

E.g. *Chairman Smith*

$< -\infty,$ 2001-01-01, 2001-01-01, $\infty >$

RANGE $< t_a, t_a, t_b, t_b >$

The temporal expression describes a range in which the slot fill is valid.

E.g. *Smith served as chairman for 7 years before leaving in 1991*

$<$ 1984-01-01, 1984-12-31, 1991-01-01, 1991-12-31 $>$

NONE $< -\infty, \infty, -\infty, \infty >$

The temporal expression is unrelated to the slot fill.

E.g. *Last Sunday Smith had a party with his friends*

$< -\infty, \infty, -\infty, \infty >$

The next two subsections describe the two classification approaches we have tested.

3.2 Flat Approach

The flat approach uses two types of features: window features and dependency features. A window feature value for the query entity, slot value, and a target temporal expression is extracted from each example. This value is a set containing all tokens that occur in the normalized sentence within 4 tokens in either direction of any instance of the normalized token in question.

Two dependency feature values for the query entity, slot value, and a target temporal expression are extracted from each example, resulting in two sets of tokens for each normalized token T. One set contains all tokens that any instance of T governs, the other set contains all tokens governed by any instance of T.

Before a feature value set for a normalized token T is created, punctuation marks, duplicate consecutive normalized tokens, and instances of T itself are removed.

Example (1) is from the evaluation set, for the query, attribute = *per:title*, entity = *Makoni*, slot fill = *minister of industry and energy development* .(1') is its normalized version.

(1) In 1981, Makoni was moved to the position of minister of industry and energy development, where he remained until 1983.

(1') In DATE, TE was moved to the position of TA , where he remained until TD.

Table 1 shows the feature values extracted from (1').

Feature	Value
TE Win	be, move, to, in, DATE, position, the
TA Win	of, to, remain, TD, position, the, where, until, he
TD Win	remain, where, until, he
TE Governs	-
TA Governs	-
TD Governs	-
TE Governed by	move
TA Governed by	position
TD Governed by	remain

Table 1: Feature Values for (1)

For two feature values U, V, let K_T be the normalized size of their intersection

$$K_T(U, V) = \frac{|U \cap C|}{\sqrt{|U|^2 + |V|^2}} \quad (1)$$

Let F denote the flat features. Then for any $G \subseteq F$, let K_S be the kernel function for a pair of examples, and $x.i$ the feature value for the i^{th} feature value type for example x:

$$K_S(x, y) = \sum_{i \in G} K_T(x.i, y.i) \quad (2)$$

With these features we trained a classiffier using Support Vector Machines (SVM) (Cortes and Vapnik, 1995; Vapnik, 1998).

[1] All examples here assume the document creation time is January 1st. 2001.

3.3 Structured Temporal Classification

3.3.1 Dependency Path Representation

In the structured approach, we exploit collapsed dependency parsed graphs generated from the Stanford dependency parser (Marneffe et al., 2006) to capture relevant grammatical relations and discover syntactic patterns. Figure 2 shows a part of the dependency graph obtained from the sentence, *"In 1975$_{[time\ expression]}$, after being fired from Columbia amid allegations that he$_{[query\ entity]}$ used company funds to pay for his$_{[query\ entity]}$ son's bar mitzvah, Davis$_{[query\ entity]}$ founded Arista$_{[slot\ fill]}$"* . In this example, *Davis* is the query entity, the slot type is *per:employee_of*, *Arista* is the slot fill, and *1975* is the time expression.

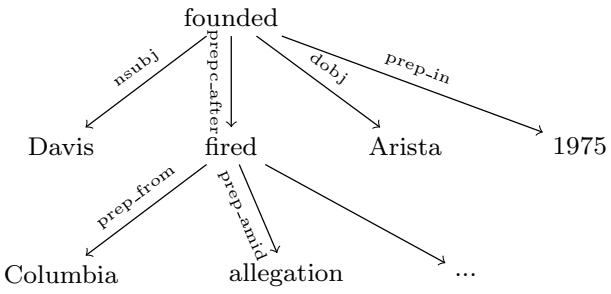

Figure 2: Dependency parsed graph of the sample sentence

We extend the idea of shortest path on a dependency graph (see Section **??**) to include three items: query entity, slot fill and time expression. Each instance is represented by three paths: (i) the path between query entity and temporal expression (P_1), (ii) the path between slot fill and temporal expression (P_2); and (iii) the path between query entity and slot fill (P_3).

Each shortest path P_i is represented as a vector $< t_1, t_2, ..., t_n >$, where t_i can be either a vertex or a typed edge in the dependency graph. Each edge is represented by one attribute, which is formed by combining the corresponding dependency type and direction. More formally, attribute $a \in \mathcal{D} \times \{\leftarrow, \rightarrow\}$, where $\mathcal{D}$ is the set of dependency types, and the arrow is directed from the governor to the dependent word. Vertices, on the other hand, may contain different levels of features, which can be found in Table 2. For example, in the sentence of Figure 2, there exists *prep_in* dependency from *founded* to *1975*. *prep_in* represents prepositional relation between these two words, meaning that the action *founded* happened at *1975*.

Feature	Description
Word	The original word token from the sentence. E.g., "Davis *founded*$_{[founded]}$ Arista"
Stem	Stemmed form of the word token. E.g., "Davis *founded*$_{[found]}$ Arista"
Entity type	Person, Location, Organization. E.g., "fired from Columbia$_{[Organization]}$"
Semantic class of trigger words	Each class contains trigger words of event subtype in Automatic Content Extraction 2005 corpus[1], and some manually collected slot type sensitive key words, E.g. if slot type is *per:spouse*, then the word *marry* belongs to one semantic class while *divorce* belongs to another semantic class.
Part-of-speech	Part-of-speech tag of original word

Table 2: Features of vertices

When we search the shortest path between two nodes, we consider all mentions of the query entity and the slot fill in a sentence. For this reason there could be more than one candidate for each P_i. We define the following simple but effective strategy to choose one path among all candidate paths. If some candidate paths contain predefined trigger words, we choose the shortest path with trigger words. Otherwise, we choose the shortest path among all candidates.

Figure 3 shows three shortest paths that result from the sentence of Figure 2. These paths not only contain lexical features such as words, but also syntactic relations. In the resulting representations, informative patterns are distilled while some irrelevant information, as well as misleading words such as *fire*, are discarded.

The next step in our system is to use a kernel function to generalize these paths and represent them in a high dimensional feature space implicitly.

3.3.2 Kernel Function

Following previous work Lodhi et al. (2002) and Bunescu and Mooney (2005), we present a string kernel function based on dependency paths. The

[1] http://projects.ldc.upenn.edu/ace/

Davis $\xleftarrow{\text{nsubj}}$ founded/found/VBD/Start-Position $\xrightarrow{\text{prep\_in}}$ 1975

Arista $\xleftarrow{\text{dobj}}$ founded/found/VBD/Start-Position $\xrightarrow{\text{prep\_in}}$ 1975

Davis $\xleftarrow{\text{nsubj}}$ founded/found/VBD/Start-Position $\xrightarrow{\text{dobj}}$ Arista

Figure 3: Three shortest paths from Figure 2

main idea is to use the kernel trick to deal with common-substring similarity between dependency paths, and to extract syntax-rich patterns from dependency paths.

Let x, y be two instances. We use $l(P)$ to denote the length of a dependency path P, $P[k]$ to denote the set of all substrings of P which have length k, and a substring $a \in P[k]$ is a substring of P with length k. For example, if P is "*ABC*", then $P[2] = \{$"*AB*", "*BC*"$\}$. The kernel function of x and y is defined as follows:

$$K_s(x, y) = \sum_{i=1}^{3} K_p(x.P_i, y.P_i) \quad (3)$$

$$K_p(P_x, P_y) = \sum_{k=1}^{Min(l(P_x), l(P_y))} \sum_{a \in P_x[k], b \in P_y[k]} \prod_{i=1}^{k} c(a_i, b_i) \quad (4)$$

Where K_p is a kernel function on two dependency paths P_x and P_y which sums the number of common substrings of feature paths in P_x and P_y with length from 1 to the maximum length. In $c(a_i, b_i)$ we calculate the inner product of the attribute vectors of a_i and b_i, where a_i and b_i are elements of two paths respectively. The final kernel function K_s does the summation of the partial results of the three dependency paths (query entity-slot fill, query entity-temporal expression, slot fill-temporal expression).

Consider the following example containing two dependency paths P_x and P_y between an entity (E) and a temporal expression (T) in two different sentences.

E$\xrightarrow{\text{nsubj}}$founded/found/VBD/Start-Position$\xleftarrow{\text{prep_in}}$T

E$\xrightarrow{\text{nsubj}}$joined/join/VBD/Start-Position$\xleftarrow{\text{prep_in}}$T

For instance, if we consider substrings of length 5 we find the following two matches:

E$\xrightarrow{\text{nsubj}}VBD\xleftarrow{\text{prep_in}}$T

E$\xrightarrow{\text{nsubj}}$Start-Position$\xleftarrow{\text{prep_in}}$T

By counting the common substrings for the remaining lengths (1 to 4) we can obtain the final result: $K_p(P_x, P_y) = 26$.

A problem of Equation (4) is that K_p has a bias toward longer dependency paths. To avoid this bias, we normalize K_p as in Lodhi et al. (2002). This normalization scales the feature vector $\phi(P)$ in the kernel space to $\phi'(P) = \frac{\phi(P)}{|\phi(P)|}$:

$$K'_p(P_x, P_y) = \frac{K_p(P_x, P_y)}{\sqrt{K_p(P_x, P_x) \cdot K_p(P_y, P_y)}} \quad (5)$$

A deviation from related work in Lodhi et al. (2002) and Bunescu and Mooney (2005) is that we count common substrings from m to maximum, rather than a fixed length. Furthermore, we only consider contiguous substrings in K_p because each substring feature in the kernel space is treated as a pattern. Non-contiguous substrings with the same length can be safely discarded as different patterns.

Although it's not easy to enumerate all substrings explicitly, like many other kernel functions, K_p can be efficiently computed by using dynamic programming in polynomial time complexity. Here, we applied a variant of the Levenshtein Distance algorithm to calculate K_p. Given the representation and kernel function, SVM (Cortes and Vapnik, 1995; Vapnik, 1998) was applied to train a classifier.

3.4 Temporal Aggregation

In order to produce the final 4-tuple for each entity/slot value pair, we sort the set of the corresponding classified temporal expressions according to the classifier's prediction confidence. We initialize a 4-tuple to $< -\infty, +\infty, -\infty, +\infty >$ and then iterate through that set, aggregating at each point the temporal information as indicated by the predicted label (see Section **??**). Given two four-tuple T and T', we use the following equation for aggregation.

$$T \wedge T' =$$
$$< max(t_1, t'_1), min(t_2, t'_2), max(t_3, t'_3), min(t_4, t'_4) >$$

At each step we modify the tuple only if the result is consistent (i.e. $t_1 \leq t_2$, $t_3 \leq t_4$, and $t_1 \leq t_4$).

The *combination algorithm*, described below in Figure 4, uses 4-tuple representation of temporal information to combine output from the flat classifier,

which uses shallow syntactic features, with that of the structured classifier, which uses deep syntactic features.

Data: Query entity E, slot type, and slot fill S
Result: Four-tuple $T =< t_1, t_2, t_3, t_4 >$

```
begin
    T ←< −∞, ∞, −∞, ∞ >
    T ← Structured Classifier Output
    T′ ← Flat Classifier Output
    for i ← 1...4 do
        if t_i = −∞ or ∞ then
            t_i ← t′_i
        end
    end
end
```

Figure 4: Combination Algorithm

In this algorithm, given an input, we consider the output from the structured classifier T as the default output. If the output equals $< -\infty, \infty, -\infty, \infty >$, then we combine it with output from flat classifier T' as final output.

4 Training data

A main part of our TSF system is the classification of temporal expressions. In order to train this classifier we need contexts where a query entity, slot value and a temporal expression are being mentioned. Additionally we need a label representing the relation of the temporal expression with the entity and the slot value (see Section 3.1).

The development data provided by the KBP2011 organizers consisted of a set of entity/slot fill pairs with their associated temporal information (expressed as a normalized date) along with the associated source document. A total of 1776 individual tuple components were provided in this dataset. Early in our system development we found that this was an insufficient amount of data to train a TSF system that would process the eight different information slots proposed for the evaluation (employee, top_employee, member of, title, spouse, country of residence, state of residence, city of residence). Early attempts to manually annotate a training dataset showed two decisive drawbacks: it is expensive an task that requires a long time and human resources to produce significant amounts of training data; annotators need strong language skills and knowledge of the events or otherwise provide low quality annotations. These drawbacks of human annotation pushed us in the direction of finding an automated process to produce the large amounts of annotated data required to train accurate classification models for the task.

4.1 Distant supervision for TSF

Distant supervision (Mintz et al., 2009) is a learning paradigm that exploits known relations (usually obtained from an existing database) to extract labeled information in context from a large document collection. The general intuition is that whenever two entities that are known to participate in a relation appear in the same context, this context is likely to express the relation in some way. By extracting a large amount of contexts, different ways of expressing the same relation will be captured and machine learning methods can be applied on this data.

The case of TSF is slightly different in that the relation between a pair of entities is already given, either by a database or by a IE component. We are interested in finding out the different ways of expressing the time associated with a relation. Consequently, we have three instead of two elements in the contexts we want to collect and annotate: the two entities involved in the relation, plus a temporal expression.

We use Freebase[2] to gather not only instances of relations, but also the start/end dates of those particular relations. Then we proceed to collect the top 100 web search results for a query containing the two named entities involved in the relation. Then we apply the same type of preprocessing described in Section 3 to collect sentences that mention the entities along with one or more temporal expressions. We follow the usual distant supervision assumption: given a context that mentions the entity pair it is likely that it will express the relation in the database. But our intuition goes beyond the usual distant supervision assumption. We also expect to label temporal expressions occurring in the context of the entity/value pair by comparing it to the start/end temporal information that is stored in our database. For instance, Freebase states that John worked for Nissan between 2001 and 2009. Confronted with the

[2]http://www.freebase.com

sentence, "Nissan chief engineer John Smith gave a lecture at UCLA on the 12 of December 1999.", our distant supervision method compares the temporal expression, "12 of December 1999", to the known temporal information in the database. Since this date falls between the known start and end dates, it assigns the label "HOLDS" to the temporal expression. Our intuitions are: if a temporal expression falls in between known start and end dates it is likely that it is expressing a date where the relation was current; if a temporal expression matches the known start/end dates it is likely that it expresses the start/end of the relation.

Not surprisingly, making this extra assumption leads to the introduction of erroneous annotations. Some common causes of error are:

- Coreference results that match the wrong named entities in a document.
- Temporal expressions that are normalized incorrectly.
- Temporal information with different granularities has to be compared. For instance: Freebase states that John married Mary in 1997, but not the exact day and month. Should we consider a temporal expression such as the 3rd September 1997 as a START ?
- Information offered by Freebase is incorrect or contradictory with information found on the Web documents.

Web search allows us to find more matching sentences but may not represent the same distribution of labels (start, end, holds, etc.) on newswire. Another drawback is that this method does not detect cases when the context does not refer to the event, even though there may be temporal expression matching the start or end of the event. This is particularly problematic when Freebase provides only a year to indicate the start or end of an event.

We obtained a total of 51404 training instances with no human intervention. This number depends on the number of web documents found for each event and the capability of detecting coreferential mentions in the documents.

4.2 Feature Reduction

The basic assumption of distant supervision can introduce noise, especially when the training data is collected from the Web. Previous approaches to distant supervision have used enhancements to reduce the noise in the training data and benefit overall system performance.

We reduced the feature space to speed up training and eliminate noisy or unnecessary features. To explicitly reduce the number of candidate features, we use univariate multi-class logistic regression to eliminate all features with a p ¿ 0.1. Second, to perform classification of the temporal fragments, we use L1, or 'lasso' regression, with the final feature set which provides additional feature selection methods. A theoretical justification for regularization is that it attempts to limit the expression of extraneous information on the solution. The lasso minimizes the residual sum of squares with the constraint that the absolute value of the regression coefficients must be less than a constant, π, that functions as a tuning parameter and is used for shrinkage. When π is large enough, there is no effect on the solution, but when it shrinks it has the effect of reducing some model coefficients close or equal to zero. We used cross-validation to determine the best values for π in our experiments.

4.3 Training Instances Re-labeling

To provide a more concise feature representation and mitigate computational issues presented by sparse matrices, we use the reduced feature set, which averaged 0.89% feature reduction for the evaluation. This reduced feature space was filtered using univariate logistic regression, to build a classifier for self-training. Using regularized regression for self-training provides the benefit of embedded feature selection. The technique has been effectively incorporated in the distant supervision framework (Surdeanu et al., 2010; Mintz et al., 2009), and we extend its use to semi-supervised learning methods for improving the labeling quality of the training set.

To relabel the training data for the evaluation, an initial classifier was build using a small set of human labeled 'seed' data. For the evaluation, we used 0.022% of the distant supervision data and initial seeds. After the data has been divided into folders,

we label a portion of the remaining unlabeled temporal instances. A criteria is used to select instances to retrain the classifier for a new unlabeled portion; and those examples that were not considered reliable were held out. To assess the reliability of a label for instance, we looked for agreement between labels assigned by the same classification model, but with different thresholds of π . As the new data portion is annotated, those retained for retraining are instances for which there is classifier agreement. When all the data has been annotated, or another stopping criteria has been reached, the procedure terminates.

5 Results

The CUNY-BLENDER team submitted three runs to each of the two TSF subtasks ('diagnostic' and 'full'):

- CUNY-BLENDER-1 uses SVM and a linear kernel to classify the temporal expressions. The classifier was trained with the distant supervision results without further modifications.
- CUNY-BLENDER-2 includes the unsupervised feature selection and self-training of distance supervision described in Sections 4.2 and 4.3.
- CUNY-BLENDER-3 uses the two SVM-based systems (one that uses shortest dependency paths based kernel, another that uses surface features such as window features and pattern features) described in Sections 3.3 and 3.2. Finally we combine the outputs of these two systems and the CUNY-BLENDER-1 system.

Table 3 and Table 4 present our results for the diagnostic task and the full task. CUNY-BLENDER-3 system achieved the highest score in the full task, especially for "spouse" slot.

6 Discussions

6.1 What Works

6.1.1 Enhance Distant Supervision through Rich Annotations

We have introduced rich annotations including name tagging, entity coreference resolution, time expression identification and normalization, dependency parsing into the distant supervision process. In this way, the projection is done on top of annotated/normalized context sentences so that more positive samples can be located more accurately. In addition, another limitation of using Web data for distant supervision is its large size of training instances and feature space. We used a logistic regression model and a self-training based instance re-labeling method to reduce the feature space and dramatically speed up model learning (about 100 times) and improve temporal classification accuracy at the same time.

6.1.2 Multi-level Reference Time Extraction

In most news and web blog documents, we can use the document creation time as the reference date to normalize time expressions. However, some events may appear in a list of parallel items in which each item has its own reference date. In the following example, we should use "Aug. 6, 2007" as the reference date for the slot fill "Tom LaSorda" as a 'top_employee" of "Daimler Chrysler AG":

> "Aug. 3, 2007: **Daimler Chrysler AG** finalizes the sale of Chrysler to Cerberus. **Aug. 6, 2007**: Bob Nardelli appointed Chrysler chairman and CEO. **Tom LaSorda** becomes vice chairman and president. Sources: Chrysler, Bloomberg, Detroit News Top Chrysler execs Bob Nardelli is the 19th man to lead Chrysler since the company's founding in 1925. Tom LaSorda, president and CEO, Sept. 2005-Aug. 2007 Dieter Zetsche, president and CEO, Nov. 2000- Sept. 2005 James P. Holden, president and CEO, Oct. 1999-Nov. 2000 Thomas T. Stallkamp, president, Jan. 1998-Dec. 1999 Robert A. Lutz, president"

We segmented such documents according to temporal blocks so that each block has its own local reference date.

6.2 How Much to Compress?

For many NLP tasks including this new TSF task, one main challenge lies in capturing long contexts. Semantic analysis such as dependency parsing can

System	Overall	Employee	City	State	Country	Member	Title	Top_members	Spouse
CUNY-BLENDER-1	0.633	**0.623**	0.850	0.652	0.720	0.637	0.600	0.653	0.614
CUNY-BLENDER-2	0.624	0.617	0.850	0.621	0.727	0.627	0.589	0.637	**0.620**
CUNY-BLENDER-3	**0.640**	0.620	0.850	**0.683**	**0.726**	**0.651**	**0.610**	**0.665**	0.609

Table 3: Results on the diagnostic TSF task (harmonic mean of Precision and Recall)

System	Overall	Employee	City	State	Country	Member	Title	Top_members	Spouse
CUNY-BLENDER-1	0.226	**0.250**	0.050	**0.202**	**0.380**	0.181	0.246	**0.122**	0.413
CUNY-BLENDER-2	**0.228**	0.249	0.072	0.120	0.339	**0.187**	**0.248**	0.119	**0.484**
CUNY-BLENDER-3	0.215	0.239	**0.079**	0.124	0.364	0.164	0.231	0.105	0.458

Table 4: Results on the full TSF task (harmonic mean of Precision and Recall)

make unstructured data more structured by *compressing* long contexts and thus reduce ambiguities. However, current core NLP annotation tools such as dependency parsing and coreference resolution are not yet ideal for real applications. The deeper the representation is, the more risk we have to introduce annotation errors. Furthermore, for certain types of slots such as "title", since the contexts are relatively short between the query and its slot fill (e.g. "Today[Time] President[Title] Obama [Query]..."), structured representation is not appropriate. Therefore we pursued a more conservative approach combining benefits from both flat approach (local context, short dependency path, etc.) and structured approach (e.g. dependency path kernel). We reported that the structured approach outperforms the flat approach for all slot types except "per:title" which usually involves shorter contexts. Furthermore, combining them through cross-document temporal aggregation can achieve higher performance than each approach alone.

For example, there is a long context between the query "Mugabe", the time expression "1980" and its slot fill "ZANU-PF" in the following sentence "ZANU, which was renamed **ZANU-PF** after taking over ZAPU, has been the country's ruling party and led by **Mugabe** since **1980**." The structured approach successfully identified "1980" as the starting date based on the short dependency paths among "ZANU-PF", "Mugabe" and "Mugabe". It's particularly effective to capture long contexts when the query has multiple slot fills in one sentence. For example, in the sentence "**Trong** became secretary of the Hanoi Party Committee in January 2000, chairman of the Central Theoretical Council in 2001, member of the CPVCC in April 2001, and member of the **Political Bureau** in **April 2006**", a structured approach can compress the long contexts among the query "Nguyen Phu Trong", slot fill "Political Bureau" and time expression "April 2006" into "[Query] member of [Slot Fill] in [Time]".

On the other hand, dependency parsing can produce errors. For example, it failed to capture the dependency relation between "September 2005" and "the Brookings Institute" in the following sentence "In **September 2005**, **Dichter** left office and became a research fellow at **the Brookings Institute** in Washington , D.C.". In contrast the flat approach can easily identify "September 2005" as the starting date for the query "Avi Dichter" to be a member of "the Brookings Institute" based on lexical features such as "became".

We also found that the gains by the structured approach are highly correlated with the compression rate, which is defined by (1 - the lengths of dependency paths among [query, slot fill, time expression] divided by the number of context words). For example, using structured approach they achieved much higher gains on residence and spouse slots (about 0.78 compression rate) than title (about 0.68 compression rate).

6.3 Remaining Challenges

6.3.1 Implicit and Wide Contexts

We found that some temporal information can be found from explicit contexts. For example, "[Query] moved to [Slot Fill] in [Time]" indicates a Residence relation has the value START, and, "[Query], TITLE of [Slot Fill]" indicates the Title and Employment relations have the value HOLD during the reference

time. However, In many cases the temporal information is implicitly represented. In particular, temporal employment/membership information can be represented by many forms. Some examples are as follows.

- *Profit*: "**Daimler Chrysler** reports 2004 profits of $3.3 billion; **Chrysler earns** $1.9 billion" indicates "Schrempp Chrysler" is an employee of "Daimler Chrysler" WITHIN the reference time;

- *Contest*: "**Sutil**, a trained pianist, **tested** for **Midland** in 2006 and raced for Spyker in 2007 where he scored one point in the Japanese Grand Prix" indicates "Sutil" was a member of "Midland" WITHIN 2006;

- *Speech*: ""**Daimler Chrysler** is not yet where we want it to be, but we are headed precisely in the right direction, " **Schrempp** says" indicates "Schrempp" is an employee of "Daimler Chrysler AG" WITHIN the reference date.

6.3.2 Coreference Resolution

As in other IE tasks, coreference resolution presents a bottleneck in this TSF task. These errors can be categorized into the following three types:

(1) Name Coreference Errors:

These errors normally appear between entity mentions that could refer to different entity types. For example, a slot fill "R" means "Republican Party", but coreference systems without using world knowledge mistakenly linked it to other names including the letter "R"; it's also difficult to identify the coreference link between a slot fill "Brooklyn Dodgers" with "Brooklyn" in the following sentence "**He** forever will be remembered for helping **Brooklyn** finally win a World Series title in **1955**." without knowing "Brooklyn" is a team name in this context.

(2) Nominal Coreference Errors:

Nominal coreference resolution remains very challenging, especially when multiple antecedents appear as candidates in different sentences from the nominal mention. For example, coreference resolution systems failed to identify the link between the slot fill "Giuliani Partners" and "the firm" in the following sentences:

> "Almost overnight, he became fabulously rich, with a $3-million book deal, a $100,000 speech making fee, and a lucrative multifaceted consulting business, **Giuliani Partners**. As a celebrity rainmaker and lawyer, his income last year exceeded $17 million. His consulting partners included seven of those who were with him on 9/11, and in 2002 Alan Placa, his boyhood pal, went to work at **the firm**.".

Even in the same sentence, coreference resolution systems need to exploit certain world knowledge to resolve some nominal mentions. For example, in the following sentence, the coreferential link between the slot fill "Toyota" and "the Japanese company" was missed so the TSF systems failed to identify "2006" as a WITHIN time for the membership between the query "Frank Perera" and "Toyota":

> "After successful karting career in Europe, Perera became part of the **Toyota** F1 Young Drivers Development Program and was a Formula One test driver for **the Japanese company** in **2006**."

(3) Pronoun Coreference Errors:

Most of the coreference errors propagated into TSF are about pronoun resolution, especially for female pronouns. For example, in the following sentences, the coreferential link between "Kelly Cass" and "She" was missed so the TSF systems were not able to identify "January 2000" as a WITHIN time for the membership between "Kelly Cass" and "The Weather Channel":

> "Meteorologist **Kelly Cass** is an On-Camera Meteorologist at The Weather Channel, **She** first appeared on air at The Weather Channel in **January 2000**."

Some newswire documents include centroid entities to which most pronouns should be linked. For example, in a document about "3rd Ld-Writethrou: Nguyen Phu Trong re-elected Vietnamese top legislator", the query "Nguyen Phu Trong" is the centroid entity and so many paragraphs only use pronouns to refer to this entity:

"<P> **He** pursued a master degree in political economics at the Nguyen Ai Quoc High- ranking Party Institute from September 1973 to April 1976. **He** worked as an editor at the magazine's Party Construction Department between May 1976 and August 1981, studied in the former Soviet Union from September 1981 to July 1983 and gained associate professor title there, became vice head and then head of the Party Construction Department between August 1983 and February 1989. </P>".

In the above paragraph, it's not difficult to identify the slot fills and time spans if the coreference resolution component successfully links these pronoun mentions to the query.

6.3.3 Temporal Reasoning

As in regular slot filling, inferences are required to extract temporal information for the remaining difficult cases. We can roughly categorize them into the following types:

Cross-entity Aggregation: The TSF task in KBP2011 was designed as a top-down question answering task, by sending one entity query and one slot fill each time. However, various entities (both queries and non-queries) and their attributes are often inter-dependent and thus their temporal boundaries can be used to infer from each other and ensure consistency.

When there are multiple entities associated with the same slot fill, temporal analysis should extend beyond individual words to common semantic patterns. For example, in the following sentence, we need to apply the pattern, "[Slot Fill1] and [Slot Fill2] respectively led by [Query1] and [Query2]" to identify slot fills for the two corresponding query entities:

"With the Lancaster House Agreement signed on Dec. 21, 1979, the Patriotic Front , consisting of ZAPU -LRB- Zimbabwe African Peoples Union -RRB- and ZANU, then respectively led by Joshua Nkomo and Mugabe, and the Zimbabwe Rhodesia government agreed on a new constitution for a new Republic of Zimbabwe with elections in February 1980."

Cross-slot Reasoning:

Some slot types are inter-dependent, such as title and employment slots, so a TSF system can conduct cross-slot reasoning in order to enhance temporal information extraction.

Sometimes an entity has different titles within the same organization, which, for example, would require us to choose between multiple candidate values for the slot type *top members/employees*. For instance, in the following sentence, even if "Nguyen Phu Trong" is given as the "top_employee/member" of "Communist Magazine" in the diagnostic task, a TSF system should identify that only the title "editor-in-chief" indicates a top-level position, and classify "August 1991" as WITHIN for this relation:

"Trong became member of the magazine's editing board between March 1989 and April 1990, deputy editor-in-chief from May 1990 to August 1991, and editor-in-chief in August 1991."

In many other cases, the temporal information for one slot type only exists in other types of slots. For example, the BEGINNING/ENDING time of Residence is often implied by the entity's birth/death events. Some examples are as follows.

Our system failed to infer the ENDING date for the residence relation between "Paul Newman" and "Westport, Conn.":

"**Paul Newman**, one of the last of the great 20th-century movie stars, died **Friday** at his home in **Westport, Conn.**."

We can infer the BEGINNING date for the employment relation between "Mugabe" and "southern African country" from the "founding" event of the country:

"The incumbent President **Robert Mugabe**, who has been the head of state since the **southern African country** gained independence in **1980**, is seeking another term in office."

Such reasoning can be extended from sentence-level to discourse-level, for example, from the following sentence we can infer "Richard Widmark" died on "March 24, 2008" and so the ENDING date of his residence relation with "Connecticut" is also "March 24, 2008":

> "Milestones: prominent deaths of **2008**
> ...
> American Hollywood actor **Richard Widmark**, aged 93 at his home in **Connecticut**. (**March 24**)."

6.3.4 Distant Supervision Challenges

Distant supervision methods have achieved some reasonable success for this new TSF task, but some significant challenges remain.

In the following we listed some assumptions made by current distant supervision methods:

(1) One sense per query:

One basic assumption of distant supervision is that an unstructured context containing elements matching a structured KB entry expresses the same relation as the KB entry. This assumption is often invalid especially when an element of the entry is ambiguous. For example, one KB entry indicates "RAUL CASTRO" is a "General" during the document creation time; but distant supervision approach mistakenly identified the following context sentence as a positive training sample but "general" is not a title here:

> "Monday, **Raul Castro** set the date for local (city and town) **general** elections as **October 21** with a second round **October 28**."

Similar ambiguities can occur when the query and slot fill share some names. For example, for a query "Giovanni Ferrero" and a slot fill "Ferrero SpA", we will get the following context sentence:

> "Since 1997, his sons, **Giovanni Ferrero** and Pietro Ferrero have led **Ferrero SpA**."

Simple name string matching cannot distinguish the query and slot fill clearly in this sentence.

(2) One query per context:

A context sentence that mentions the tuple of <query, slot fill, time expression> is likely to express the relations among these three elements. But because of the high variability of time expressions, they are often linked to other entities, relations or events in the context. Therefore, we should consider other facts in the context sentence as competitors for any given query and slot fill in order to connect with the candidate time expressions. Then we can compare their confidence values to filter out some wrong temporal answers. In the following example that is identified as a context sentence for <Chris Kronner, Slow Club, December>:

> Slow Club's Chris Kronner faced similar challenges taking on his second executive chef position at Serpentine, which opened in December.

(3) One sentence per query: Almost all systems that used distant supervision did so on the sentence-level, and assume that three elements in the tuple should exist in the same sentence after entity coreference resolution. This is invalid when a document is typically talking about a centroid entity (e.g. the employment history of a person or an organization). In such a document, a distant supervision approach should locate the centroid query or slot fill and search for context sentences that include the other elements of a KB entry. For example, in the second example in section 8.2.3, a system needs to recognize "Chrysler LLC" as the employer slot fill for all of the person query entities.

6.3.5 "Long Tail" Problem

We extracted indicative contexts from distant supervision and summarized the number of instances that match each pattern in Figure 5, 6 and 7.

The final challenge lies in the long-tailed distribution of temporal context patterns. A few patterns match many instances, while a high percentage of patterns match only a few instances. Dependency parsing can filter out some irrelevant contexts but deeper understanding will be required to adequately generalize from diverse lexical contexts. For example, the starting date of an employment relation can be expressed by many long-tail patterns such as, "would join", "would be appointed", "will start at",

"went to work", "was transferred to", "was recruited by", "took over as", "succeeded PERSON", "began to teach piano", etc.

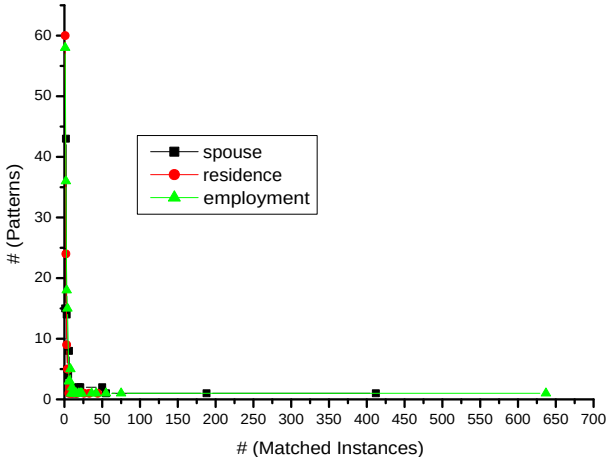

Figure 5: BEGINNING Pattern Distribution

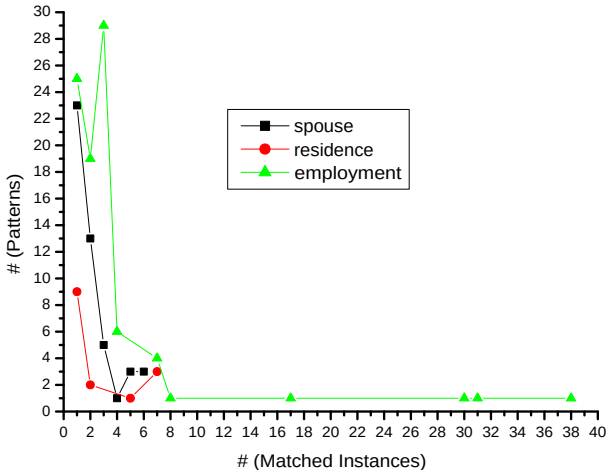

Figure 6: WITHIN Pattern Distribution

7 Related work

While the extraction of temporal arguments for relations and events has recently received the attention of the TempEval community (Verhagen et al., 2007; Pustejovsky and Verhagen, 2010), it is focused on extracting temporal relations from individual documents. In this "information explosion era", the amount of available data continues to grow exponentially. Adequate management of temporal information cannot be achieved by local extraction alone; instead, facts must be aggregated across multiple documents and reasoning must be conducted over these facts in order to generate compact and accurate knowledge bases.

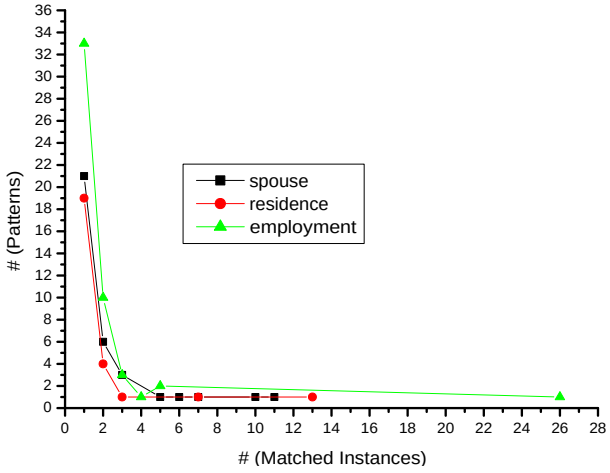

Figure 7: ENDING Pattern Distribution

Various approaches have been developed for this task, which can be roughly categorized into flat or structured approaches:

Flat approaches: Chambers et al. (2007) built a supervised learning model based on syntactic and semantic features at the lexical level, making use of tense and aspect, to classify a pair of event triggers in a sentence . Yoshikawa et al. (2009) used similar features, using a Markov Logic based joint inference framework for temporal relations. Ling and Weld (2010) also exploited cross-event joint inference, but they used shallow dependency features to build local formulas without considering the deeper syntactic structure.

Structured approaches: Bethard and Martin (2007) designed syntactic and semantic features based on syntactic treelets and verbal government for temporal relation classification. (Puşcaşu, 2007) used sentence level syntactic trees to propagate temporal relations between syntactic constituents. (Cheng et al., 2007) introduced a type of feature called *tree position* that classifies nodes on a syntactic dependency tree based on their position in the tree relative to that of a target node.

The need for structural representations is acknowledged in many Natural Language Processing fields. For example, statistical machine translation has recently moved from flat-structured models such

as word-based (Brown et al., 1993) and phrase-based models (Yamada and Knight, 2001; Zens and Ney, 2004) to more tree-structured models (Shen et al., 2010; kiu Lo and Wu, 2011). The shortest path between two vertices in a dependency parsed graph has been used to capture the syntactic and semantic relation between two words (Snow et al., 2005; Bunescu and Mooney, 2005; Wu and Weld, 2010).

8 Conclusions

In this paper we have presented our approaches for KBP2011 temporal slot filling task, and shared the lessons we have learned from this pilot study. Experimental results show that the individual flat and structured approaches both outperform bag-of-words based classifier and that the hybrid approach and re-labeling approach lead to statistically significant improvements. The query selection and document selection in the diagnostic task are not representative of normal temporal information distribution in newswire data, which has made it difficult to achieve much higher performance than some simple baselines (e.g. label any time expression in the sentence with the query and slot fill as HOLD). However we believe our approach is promising when the task setting becomes more reasonable. In the future we are particularly interested in conducting cross-query and cross-slot temporal reasoning to enhance the performance.

Acknowledgments

This work was supported by the U.S. Army Research Laboratory under Cooperative Agreement No. W911NF- 09-2-0053 (NS-CTA), the U.S. NSF CAREER Award under Grant IIS-0953149, the U.S. NSF EAGER Award under Grant No. IIS-1144111, the U.S. DARPA Broad Operational Language Translations program, PSC-CUNY Research Program and CUNY Junior Faculty Award. 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 U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

Steven Bethard and James H. Martin. 2007. Cu-tmp: Temporal relation classification using syntactic and semantic features. In *In SemEval-2007: 4th International Workshop on Semantic Evaluations*.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Dellla Pietra, and Robert L. Mercer. 1993. The Mathematics of Statistical Machine Translation. *Computational Linguistics*, 19:263–311.

Razvan C. Bunescu and Raymond J. Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proc. of the HLT and EMNLP*, pages 724–731.

Nathanael Chambers, Shan Wang, and Dan Jurafsky. 2007. Classifying temporal relations between events. In *In Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 173–176.

Z. Chen, S. Tamang, A. Lee, X. Li, W.P. Lin, M. Snover, J. Artiles, M. Passantino, and H. Ji. 2010. Cuny-blender tac-kbp2010 entity linking and slot filling system description. In *Proceedings of the 2010 Text Analysis Conference*.

Yuchang Cheng, Masayuki Asahara, and Yuji Matsumoto. 2007. Naist.japan: Temporal relation identification using dependency parsed tree. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 245–248.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. In *Machine Learning*, pages 273–297.

Jenny Rose Finkel, Trond Grenager, and Christopher D. Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL*.

Heng Ji, Ralph Grishman, Hoa Trang Dang, Xuansong Li, Kira Griffitt, and Joe Ellis. 2011. An Overview of the TAC2011 Knowledge Base Population Track. In *Proceedings ot the Third Text Analytics Conference (TAC2011)*.

Chi kiu Lo and Dekai Wu. 2011. Structured vs. Flat Semantic Role Representations for Machine Translation Evaluation. In *Fifth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-5)*.

Xiao Ling and Daniel S. Weld. 2010. Temporal information extraction. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*.

H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. 2002. Text classification using string kernels. *The Journal of Machine Learning Research*, 2:419–444.

Marie-Catherine De Marneffe, Bill Maccartney, and Christopher D. Manning. 2006. Generating Typed Dependency Parses from Phrase Structure Parses. In *In LREC 2006*.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACL/AFNLP*, pages 1003–1011.

Georgiana Puşcaşu. 2007. Wvali: Temporal relation identification by syntactico-semantic analysis. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 484–487.

James Pustejovsky and Marc Verhagen. 2010. Semeval-2010 task 13: Evaluating events, time expressions, and temporal relations (tempeval-2).

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2010. String-to-Dependency Statistical Machine Translation. *Computational Linguistics*, 36:649–671.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2005. Learning syntactic patterns for automatic hypernym discovery. In *Advances in Neural Information Processing Systems 17*, pages 1297–1304.

Mihai Surdeanu, David McClosky, Julie Tibshirani, John Bauer, Angel X. Chang, Valentin I. Spitkovsky, and Christopher D. Manning. 2010. A simple distant supervision approach for the tac-kbp slot filling task. In *Proceedings of the Third Text Analysis Conference (TAC 2010)*, Gaithersburg, Maryland, USA, November.

Vladimir Vapnik. 1998. *Statistical Learning Theory*. John Wiley.

Marc Verhagen, Robert Gaizauskas, Frank Schilder, Graham Katz, and James Pustejovsky. 2007. Semeval2007 task 15: Tempeval temporal relation identification. In *In SemEval-2007: 4th International Workshop on Semantic Evaluations*.

Fei Wu and Daniel S. Weld. 2010. Open Information Extraction using Wikipedia. In *Proc. of the 48th Annual Meeting of the Association for Computational Linguistics*.

Kenji Yamada and Kevin Knight. 2001. A Syntax-based Statistical Translation Model. In *Proc. ACL2001*.

Katsumasa Yoshikawa, Sebastian Riedel, Masayuki Asahara, and Yuji Matsumoto. 2009. Jointly identifying temporal relations with markov logic. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 405–413.

Richard Zens and Hermann Ney. 2004. Improvements in Phrase-Based Statistical Machine Translation. In *Proc. HLT/NAACL 2004*.