Improved Temporal Relation Classification using Dependency Parses and Selective Crowdsourced Annotations

Jun − Ping NG Min − Yen KAN
National University of Singapore, 13 Computing Drive, Singapore 117417
{junping,kanmy}@comp.nus.edu.sg

ABSTRACT
We study the problem of classifying the temporal relationship between events and time expressions in text. In contrast to previous methods that require extensive feature engineering, our approach is simple, relying only on a measure of parse tree similarity. Our method generates such tree similarity values using dependency parses as input to a convolution kernel. The resulting system outperforms the current state-of-the-art.

To further improve classifier performance, we can obtain more annotated data. Rather than rely on expert annotation, we assess the feasibility of acquiring annotations through crowdsourcing. We show that quality temporal relationship annotation can be crowdsourced from novices. By leveraging the problem structure of temporal relation classification, we can selectively acquire annotations on problem instances that we assess as more difficult. Employing this annotation strategy allows us to achieve a classification accuracy of 73.2%, a statistically significant improvement of 8.6% over the previous state-of-the-art, while trimming annotation efforts by up to 37%.

Finally, as we believe that access to sufficient training data is a significant barrier to current temporal relationship classification, we plan to share our collected data with the research community to promote benchmarking and comparative studies.

KEYWORDS: temporal relations, information extraction, crowdsourcing, convolution kernels, clause structure, dependency parsing.

1 Introduction

We study the problem of temporal information extraction; specifically, we seek to establish when an event has taken place relative to a time expression. We refer to this task as *event-temporal* relationship classification. We adopt the same definitions for event and time expressions used in TimeML[1]. Consider the sentence[2]:

> Two top aides to Netanyahu , political adviser Uzi Arad and Cabinet Secretary Danny Naveh, left for Europe on *Sunday* , apparently to investigate the Syrian issue , the newspaper said. (1)

In Sentence 1 "*Sunday*" is a time expression, while the various underlined words such as "said" are events. A pictorial summary of the timeline of the events as they are described in the sentence is shown in Figure 1. The time expression "*Sunday*" is marked on the timeline to represent an instance in time. The spans of the arrows representing each of the events denote the temporal relation between the three events and the time expression. In this case "left" happens in the same time span denoted by "*Sunday*". The other two events (i.e. "investigate" and "said") do not coincide with the "*Sunday*" marker and take place after the time span denoted by "*Sunday*".

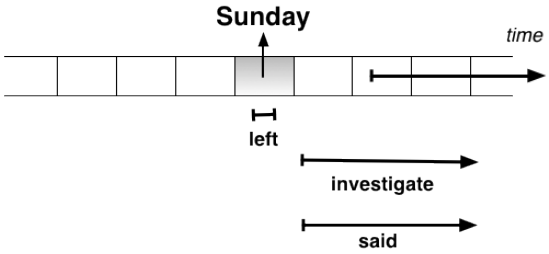

Figure 1: Temporal relationships between events and a time expression in Sentence 1.

Knowing such temporal relations are useful for downstream natural language processing applications. For example, Harabagiu and Bejan (2005) attempt to identify relationships between entities and time expressions to improve their question answering system. They combine knowledge of the relations between answer candidates and temporal expressions with semantic inference to derive exact answers to temporal questions. Temporal relations are also a recent focus in information extraction. In the temporal slot-filling task of the Knowledge Population Track featured in the Text Analysis Conference (Ji et al., 2011), the shared task required systems to add temporal information to extracted information. Such information would enable inference on such knowledge bases to be extended to event ordering and perhaps help to identify temporal contradictions.

Recognizing the value of temporal relations, the community has organized shared tasks to spur research efforts in this area. In our work, we adopt the community-standard TempEval-2 task definition and dataset (Verhagen et al., 2010). This dataset consists of 959 training instances for the *event-temporal* relationship classification task, and a testing set of 138 instances. A summary of the various *event-temporal* relationships annotated in the dataset is shown in Table 1.

[1] `http://www.timeml.org`.

[2] Extracted from document APW19980301.0720 of the TempEval-2 dataset.

Event-Temporal Relationship	Relative temporal ordering
OVERLAP	Event happens *within* time expression
BEFORE	Event happens *before* time expression
AFTER	Event happens *after* time expression
BEFORE-OR-OVERLAP	Event happens either *before* or *within* time expression
OVERLAP-OR-AFTER	Event happens either *within* or *after* time expression
VAGUE	Unable to assign a relation

Table 1: Event-temporal relationships in the TempEval-2 dataset.

State-of-the-art approaches to this problem make extensive use of careful feature engineering to model both surface lexical cues as well as information about the syntactic relations between parts of the sentences, focusing on the words that comprise the events and time expressions. Both feature types manifest a wide array of values, making data sparsity a concern. Current state-of-the-art systems achieve only about 65% accuracy, where the low performance has indeed been partially attributed to data sparsity (Yoshikawa et al., 2009).

There are several techniques to address sparse data. We adopt two in this paper, both of which lead to improved performance. First, we simplify the input features and classification methodology. Specifically, we use a support vector machine provisioned with only two features: dependency parse similarities of the time expression and the path from the event to time expressions. By limiting our input features, we obtain denser judgments which enhance the quality of the decision boundary. Second, we can enlarge our annotated dataset. To this end, we exploit annotations obtained from novices via crowdsourcing. We show that augmenting the TempEval-2 dataset with crowdsourced information improves classification performance slightly. However, we can further boost performance by selectively acquiring annotations for difficult and under-represented categories of instances. Pairing both techniques together allow us to achieve an 8.6% improvement in accuracy over the current state-of-the-art, which is statistically significant.

After first reviewing the related work, the remainder of this paper sequentially details these two key contributions of our temporal classification work. In the first half, we detail our approach for temporal relation classification. In the second half, we discuss our approach to selectively annotate important problem instances via crowdsourcing.

2 Related Work

There has been a good amount of research on temporal relationship classification, culminating in recent shared tasks in TempEval-1 (Verhagen et al., 2007) and TempEval-2 (Verhagen et al., 2010). In fact, the *event-temporal* relationship classification problem we focus on is exactly Task C in TempEval-2. Other temporal information extraction tasks include *event-event* relationship classification, which involves determining which of two events took place earlier. Top performing teams attempting Task C made use of supervised machine-learning, including conditional random fields (Kolya et al., 2010), Markov logic (UzZaman and Allen, 2010; Ha et al., 2010), and maximum entropy classification (Derczynski and Gaizauskas, 2010). Enlarging the problem, Yoshikawa et al. (2009) proposed building an inference model which jointly predicts *event-temporal*, *temporal-document creation time* and *event-event* temporal

relationships within a given sentence. While the system performed comparably to other systems in TempEval-1, such a joint model greatly enlarges the problem complexity as the search space is a Cartesian product of the three separate problems.

All of the above systems employed similar features which we categorize into three feature types: 1) lexical cues, such as signal words and part-of-speech tags, 2) context, including the attributes of the event and time expressions and 3) the grammatical structure of the sentences, obtained by processing automatic parses. Possibly since they use similar features, the top performing systems all perform similarly, achieving around 65% accuracy in their judgments. We note that the feature types employed can take on many different values and thus represent a large potential feature space. Given that the TempEval-2 dataset comprises of only 959 instances, we believe the learned models suffer from sparse data, and can improve from the incorporation of additional data.

Many researchers have looked at ways to reduce the annotation effort required to build a sizable temporal dataset. Setzer et al. (2003) proposed making use of a set of inference rules to identify useful unlabeled instances for annotation. If the equivalence of an unlabeled instance to an already labeled instance can be shown, the unlabeled instance is discarded. Similarly, we are proposing a systematic way in which unlabeled instances can be left out of the data acquisition process. But instead of making use of pre-identified inference rules, our proposal is fully automatic and exploits the dependency parse structure of unlabeled instances without prior curation of hand-written rules.

In a bid to balance the cost of data acquisition with the need for more annotated data, the organizers of TempEval-3 (UzZaman et al., 2012), to be held at SemEval-2013 propose augmenting a manually annotated dataset of 100K word tokens with around 500K semi-automatically annotated instances. At the time of this writing, the semi-automatically annotated instances have yet to be released.

In recent times, crowdsourcing has been a popular method employed to collect linguistic annotations. Munro et al. (2010), for example, lists some linguistic projects which have benefitted from crowdsourcing. To the best of our knowledge however, there is no existing literature on the adaptation of crowdsourcing for temporal datasets.

3 Convolution Kernels using Dependency Parse Trees

What features and model should we adopt to create a robust temporal relation classifier that takes full advantage of the small amount of annotated training data?

Given the sparsity of values in the features used by previous work, our approach is to reduce the dimensionality of the input feature space, to make the feature space more dense. While we concur with the prior work that the relation between the events and time expression are important, we feel that the current approaches to model this information can be simplified. Such structural relations are perhaps best captured by parse tree paths. Intuitively, if the parse tree of a testing instance is similar to one from a training instance, we may guess that the training instance's type of temporal relation would also hold for the previously unseen testing instance.

To compute the similarity between two parse trees, a typical approach is to engineer flat representations of the parse structure through feature engineering. This process is time consuming and often requires good knowledge of the problem structure to decide which

features are more discriminative than others. Convolution kernels (Collins and Duffy, 2001) on the other hand model this form of structure similarity well. Convolution tree kernels take as input tree structures and calculate a degree of similarity between the two trees. In its simplest form, similarity is computed by recursively counting the number of identical subtrees that appear in both input instances. With this structural similarity measure, we can do away with the need to "flatten" the structure with hand-devised representations. For these reasons, we use a support vector machine (SVM) as our supervised classification model, adopting a convolution kernel as its kernel function (Moschitti, 2006).

A second issue is then to decide the type of parse tree to employ. Most previous work centered around the use of constituent grammar parses. In fact, the only other work that also adopts a convolution kernel approach to temporal relation classification (Mirroshandel et al., 2011) also uses constituent grammar parses. However, as constituent parses create internal phrasal nodes for every semantic constituent, such parse trees are often deep and overly detailed. Paths in such trees are fine-grained, capturing nuances (*e.g.*, intervening finite verb phrase nodes), and as such, may not generalize well when used to compute tree or path similarities.

To remedy this, we employ dependency parses instead of constituent parses. Dependency parses are generally more compact than constituent grammar parses because they have no immediate phrasal nodes. This translates into a more compact feature space. Bearing in mind that one of the problems we are trying to overcome is that of data sparsity, we find that dependency parses are generally more useful than constituent parses for this task.

We compute two features based on dependency parses:

1. ***Dependency path from event to time expression.*** Starting from a full dependency parse of a sentence, we identify the vertices representing the event and time expressions. We locate the shortest path between these two vertices and use this sub-tree as a feature.
2. ***Dependency parse of the time expression.*** Time expressions can range from single word tokens to multi-word phrases, with vastly different semantics. For example, *Friday*, *last Friday* and *next Friday* convey very different meanings. To capture this, we extract a sub-tree from the full dependency parse consisting of all the vertices and edges related to the time expression and use this as a feature.

3.1 Evaluation

How does our simplified, two input feature SVM compare against the prior work on the TempEval-2 dataset?

Table 2 gives the performance of our classifier which we will refer to as `SVMConvoDep` vis-a-vis the top performing systems in TempEval-2. We adopt the same *accuracy* metric used in the TempEval-2 task, which is the number of correct answers divided by the number of answers. We also list macro-averaged precision, recall, and F_1 measures. We find macro-averaged measures more appropriate than micro-averaged ones for this task. As will be explained later, the dataset has a skewed distribution of labels, with `OVERLAP` forming the majority label. Micro-averaged measures give more weight to the most common label in such skewed datasets, but we feel that it is important for systems to be able to perform well across all the temporal labels.

To the best of our knowledge, our classifier outperforms all previous temporal relation classifiers. While this performance gain is probably not statistically significant (as we have no access to the participating systems' individual judgments, it is impossible to check for statistical significance),

we feel that these are impressive results as our classification input is decidedly simple (just two subtrees derived from dependency parses as features).

System	Accuracy (%)	Precision	Recall	F_1
SVMConvoDep	67.4	0.828	0.512	0.523
TRIOS	65.0	Not Available		
JU_CSE	63.0			
NCSU-indi	63.0			
NCSU-joint	63.0			
TRIPS	63.0			
USFD2	63.0			

Table 2: Performance on TempEval-2 testing set. Results for TempEval-2 systems are cited from Verhagen et al. (2010).

4 Enlarging the Dataset through Crowdsourcing

Our next proposal to solve the data sparsity problem is to enlarge the dataset available. Developing a large, suitably annotated dataset is expensive – both in terms of time and monetary cost. Recent work in natural language processing suggests that crowdsourcing annotations from the untrained public can provide annotated data at similar annotation quality as expert annotators, but for a fraction of the cost (Sheng et al., 2008; Hsueh et al., 2009). However we are unsure if these results are applicable to a temporal dataset which is inherently complex (Setzer and Gaizauskas, 2000) and require a better understanding of the target language.

To find out, we set up a crowdsourcing task on CrowdFlower. We chose CrowdFlower as our crowdsourcing platform, as it has access to a large user base (it uses Amazon Mechanical Turk to find workers), but adds an extra validation layer to attempt to address quality concerns, as this has been an issue in many applications of crowdsourcing in natural language annotation tasks (Mason and Watts, 2009; Callison-Burch and Dredze, 2010).

4.1 Task Setup

Each annotation instance consists of a single sentence. We pre-process the sentence to highlight one event expression and one time expression found within it. Annotators were tasked to choose from five (`OVERLAP`, `AFTER`, `BEFORE`, `NOT-RELATED`, `BAD-SENTENCE`) possible temporal relationships between the marked event and time expression. An additional choice for `BAD-SENTENCE` was included to allow annotators to indicate if there had been problems with our automatic pre-processing. Such instances (67 in our study) were discarded.

We required that at least 3 judgments be made for each annotation, and majority voting was then used to decide on a final label for each annotation. To ensure the quality of the judgments we had obtained, we made use of a validation facility provided by CrowdFlower. Pre-annotated gold instances were mixed together with unlabeled instances. These gold instances were used to validate the annotations made by each annotator. Annotators were not informed which were the gold instances. During the annotation process, annotators who failed to label these gold instances correctly were stopped from proceeding with the task, and the annotations they made were discarded.

Raw Data. To ready a set of unlabeled instances for annotation, we crawled news articles on several news web sites, including *Wall Street Journal*, *New York Times*, *CNN*, and *Channel News*

Asia from 2 June to 8 July 2012. We made use of the sentence splitting module from the Apache OpenNLP[3] library to obtain individual sentences from these news articles. We built a CRF-based event and time expression extractor (`CRFEventTimexExt`) which is able to automatically identify event and time expressions in each of the collected sentences. For event extraction, we made use of part-of-speech (POS) tags as a feature. For time expression extraction, we made use of a compiled lexicon of the days of a week and months of a year on top of POS tags. The performance of our extractor is compared against top participating systems in TempEval-2 in Table 3. We include only the results of systems which are able extract both event and time expressions as it is not required for systems to be able to do both in TempEval-2.

System	Event			Time		
	Precision	Recall	F_1	Precision	Recall	F_1
`CRFEventTimexExt`	0.81	0.82	0.82	0.76	0.62	0.68
Edinburgh	0.75	0.85	0.80	0.85	0.82	0.84
TIPSem	0.81	0.86	0.83	0.92	0.80	0.85
TRIPS	0.55	0.88	0.68	0.85	0.85	0.85
TRIOS	0.80	0.74	0.77	0.85	0.85	0.85

Table 3: Comparison of event and time expression identification with TempEval-2 systems. Performance of TempEval-2 systems are cited from Verhagen et al. (2010).

We are able to turn in competitive F_1 scores for event expression extraction, but less so for time expression extraction due mainly to poor recall. We find the precision scores of 0.81 and 0.76 for event and time expression identification respectively sufficiently high as this is part of pre-processing. Incorrectly identified event and expressions can be labeled as so by annotators, and these precision scores will minimize the occurrences of such mistakes. To address the problems with recall, we can always collect more sentences to increase the number of time expressions we have for annotation.

4.2 Experiments

We first want to test how a similarly sized crowdsourced annotation compares with the manual, expert annotations from TempEval-2. For this reason, we extracted an initial batch of 1,000 tuples (close in size to the TempEval-2's training data of 959 instances) of event and time expressions from the sentences we had crawled. We collected annotations for each of the tuples on CrowdFlower. We refer to this dataset as `d-1000`. We compared the distribution of the labels within this collected set with the TempEval-2 training and testing sets in Table 4.

The skewed distribution in the TempEval-2 datasets are similarly observed in the collected dataset, with the `OVERLAP` label taking up more than 50% of the whole dataset.

We trained a new classifier using the same features as `SVMConvoDep` with the crowdsourced annotations `d-1000`. The performance of this trained classifier, `CF-1000`, on the TempEval-2 testing set is shown in Table 5. In the table we have also included macro-averaged precision, recall and F_1 measures.

`CF-1000` did not do as well as `SVMConvoDep` in terms of accuracy. While its F_1 score is slightly higher, the difference is not statistically significant. This is not surprising for two reasons. First

[3] `http://opennlp.apache.org/`.

Dataset	Size	Distribution of label (%)			
		OVERLAP	BEFORE	AFTER	Others
TempEval-2 training set	959	53.8	18.0	23.0	5.2
TempEval-2 testing set	138	55.1	14.5	19.6	10.9
d-1000	1000	70.0	12.2	17.8	0.0

Table 4: Distribution of labels in different datasets.

the annotators recruited for the annotation task are not domain experts, and we can expect mistakes in the annotations obtained. Second, SVMConvoDep is trained on the TempEval-2 training set. This dataset is prepared in similar fashion together with the TempEval-2 testing set we are evaluating on. We will expect the two datasets to share more similar attributes and characteristics than the d-1000 set that we had collected. The content of the TempEval-2 datasets and ours span a different time period, are sourced from different sources, and possibly drawn from different domains and categories.

We tried to combine the TempEval-2 training set with d-1000. With this combined dataset, we trained another classifier CF-1000+TE. The performance of this classifier is also reported in Table 5. We get improved results in both accuracy and F_1 scores. This improvement is significant, with $p < 0.05$ when tested with the paired Student's t-test.

System	Accuracy (%)	Precision	Recall	F_1
SVMConvoDep	67.4	0.828	0.512	0.523
CF-1000	65.2	0.578	0.535	0.525
CF-1000+TE	71.7	0.726	0.598	0.615

Table 5: Classifier performance on TempEval-2 testing set.

There are two important conclusions that we can draw from these results. First, the difference in performance between SVMConvoDep and CF-1000 is slight. So while the the novices recruited via crowdsourcing may not have been domain experts, they are able to generate a dataset that is comparable to an expert-curated one.

Then, the improvement to the performance of CF-1000+TE shows us that despite the possible differences in time span, source, and categorization of d-1000 from the TempEval-2 dataset as we have suggested, there is value to the crowdsourced dataset. Doubling the amount of training data available by putting d-1000 and the TempEval-2 training set together rewards performance.

5 A Smarter Way

With crowdsourcing, the costs associated with generating temporal datasets are lowered. Yet we want to do this efficiently with minimal wastage. In this section, we explain how we can selectively acquire annotations to reduce the costs involved without affecting the efficacy of the data collected.

Motivation. We recognize that it is often not easy to decide on the relationship between an event and time expression. For example let us go back and take a look at Sentence 1.

Within the sentence there are several event expressions. For the sake of this discussion we focus on just two of them: 1) "left", and 2) "said". One immediate observation is that time

expressions are commonly adjuncts (in this case, a prepositional phrase (PP)) that attach to a verb phrase (VP). This implies that time expressions often directly modify only the head it is attached to. In such cases, it is usually easy to identify the temporal relationship between the event and time expression. Looking at "left" and "*Sunday*", it is quite straightforward to determine that they take place within the same time span.

However, it gets significantly more difficult when we look at "said" and "*Sunday*". We have to read through more of the sentence to build up an understanding of the relation between "left", "investigate", and "said" before we can conclude that "said" takes place after "*Sunday*".

Our key insight to more efficient data acquisition is to leverage this observation. We postulate that it is more useful if we can gather annotations for complex instances instead of easier ones.

Definitions. We define a few terms that we will use in the rest of this paper. For the *event-temporal* relationship classification problem, the input is a collection $\mathbb{S}$ of sentences. Each sentence s is composed of one or more word tokens, i.e. $s = w_1 w_2 \ldots$. Let s^* be the set of all possible subsequences of s. For each s, we can define a set of unigram word tokens $\epsilon_s = \{w, w \in s\}$ that are event expressions. We can also define a set $\Theta_s = \{\theta, \theta \in s^*\}$ which includes all the time expressions within s. The problem then is to define some function $f : s, e, t \rightarrow R, s \in \mathbb{S}, e \in \epsilon_s, t \in \Theta_s$ where R is the temporal relationship between the event expression e and time expression t.

Building on our observation, we want to be able to partition a set of unlabeled instances so that the more complex instances are separated from the easier ones. We propose building such a partitioning scheme around the ordering of the elements in ϵ_s given a time expression $t \in \Theta_s$.

To do this, we first build a dependency parse[4] of the input sentence s, where each $w_i \in s$ forms a vertex within the dependency parse tree. A portion of the dependency parse tree for Sentence 1 is shown in Figure 2.

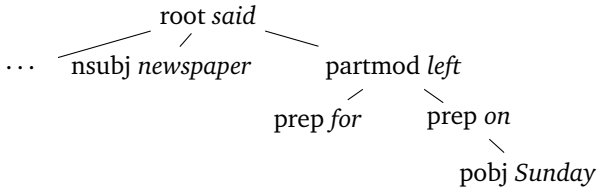

Figure 2: Excerpt of the dependency parse for Sentence 1.

In the figure, the governing dependency relation of a word is shown as a vertex. The associated word is also shown in *italics* to illustrate which part of the sentence this parse is about.

For a given dependency parse and a target time expression t, we can define a total order O_t on ϵ_s. O_t is defined by arranging each $w \in \epsilon_s$ in ascending order of their respective distances from the time expression vertex. Distance in this case is defined as the number of edges that needs to be traversed to reach the time expression vertex.

Imposing the total order O_t on ϵ_s gives us a totally ordered set, i.e. $(O_t, \epsilon_s) = \{e_0, e_1, \ldots\}$. From this, for every input sentence s and its associated event (e) and time expressions (t), we can place the tuple $< s, e, t >$ into a partition $\mathscr{P}_i$, where i is the index of e within (O_t, ϵ_s).

[4]We make use of Stanford dependencies (Klein and Manning, 2003).

Referring to Sentence 1 as an illustration, we thus place <"left", "*Sunday*"> in $\mathscr{P}_0$ because "left" is the nearest event expression to "*Sunday*" in the dependency parse in Figure 2. <"said","*Sunday*"> will be placed in $\mathscr{P}_1$ as "said" is the next nearest event expression. For convenience, we will also be referring to $\mathscr{P}_i$ as the set of Level-*i* instances.

This partitioning scheme is premised on the intuition that it requires more effort to understand the temporal relationship between event and time expressions which are both structurally and semantically further away from each other. Higher level instances thus should be more complex than their lower level peers.

Following this scheme, we separated the TempEval-2 testing set into different partitions based on our prescribed methodology. A breakdown of the performance of `SVMConvoDep` on each of the partitions $\mathscr{P}_i$ is shown in Table 6. Accuracy drops steadily from Level-0 instances to Level-2 instances. This provides support for our intuition that higher level instances are harder to classify accurately.

Given that there are only 10 Level-3 instances and 1 Level-4 instance, the variance in measurements can potentially be very wide. There are too few Level-3 and Level-4 instances for their results to be analyzed reliably.

System	Accuracy (%)				
	Level-0 (59)	Level-1 (47)	Level-2 (21)	Level-3 (10)	Level-4 (1)
`SVMConvoDep`	84.5	66.0	42.9	30.0	100.0

Table 6: Breakdown of performance on partitions of TempEval-2 testing data. The number of instances for each partition is indicated in parentheses.

Given the high prediction accuracy on Level-0 instances, we argue that it is not necessary to obtain more annotations for them. Instead we should focus on the higher level instances. By opting not to annotate additional Level-0 instances, substantial cost savings can be made, as seen from Table 7. In the table we present a breakdown of the relative size of each partition to its entire dataset. `d-full` is a set of 8,851 tuples of event and time expressions that we have extracted from the same set of sentences we had crawled earlier. As seen from the table, Level-0 instances consistently form a large part of the various datasets. Not annotating Level-0 instances will directly lead to a cost savings of at least 37%.

Dataset	Relative size of partition (%)			
	Level-0	Level-1	Level-2	Others
TempEval-2 Training Set	40.9	35.2	15.1	8.8
TempEval-2 Testing Set	41.4	34.3	15.7	8.6
`d-full`	37.0	34.3	17.5	11.2

Table 7: Breakdown of partition sizes of different datasets.

5.1 Experiments

To study the effectiveness of our recommendations, we collected annotations for all the tuples in `d-full` via CrowdFlower in a similar way to what we had done earlier. From this `d-full` collection of 8,851 annotations, we removed all Level-0 instances to create a subset of 5,576 annotations which we will call `d-nolevel0`.

Using d-full and d-nolevel0, we trained two new classifiers CF-Full and CF-NoLevel0 respectively. Table 8 shows the performance of these two new classifiers with that of SVMConvoDep when tested with the TempEval-2 testing set.

From the results, we note that CF-NoLevel0 is able to deliver a significant performance gain of about 8.6% over SVMConvoDep ($p < 0.05$) even though it only made use of part of the full dataset we had collected. Further, it is able to match the performance of CF-Full which was trained over all collected instances (d-full). The performances of CF-NoLevel0 and CF-Full are not significantly different.

System	Accuracy (%)	Precision	Recall	F_1
SVMConvoDep	67.4	0.828	0.512	0.523
CF-NoLevel0	73.2	0.659	0.643	0.639
CF-Full	73.2	0.660	0.647	0.641

Table 8: Accuracy on TempEval-2 testing set.

These results are illuminating. We see that our proposed partitioning scheme is able to reliably identify unlabeled instances that will not be able to contribute to better classifier performance. By focusing our annotation effort only on more complex instances, we bring down data acquisition costs by a large amount (37%) while retaining classifier performance.

6 Analysis and Discussion

6.1 Selective Data Acquisition

We try to probe deeper into the performance of both CF-NoLevel0 and CF-Full. We want insight into why the former works as well as the latter, despite the large reduction in training data.

Table 9 shows a more concise breakdown of the precision, recall and F_1 scores of both classifiers when tested with the TempEval-2 testing set. We see that both classifiers turn in similar performance across all three labels. Performance for the OVERLAP label is better, possibly because there are more training instances for it in the training dataset.

Classifier	OVERLAP			BEFORE			AFTER		
	P	R	F_1	P	R	F_1	P	R	F_1
CF-NoLevel0	0.72	0.96	0.82	0.56	0.45	0.5	0.70	0.52	0.60
CF-Full	0.72	0.95	0.81	0.57	0.40	0.47	0.70	0.60	0.64

Table 9: Performance measures on TempEval-2 testing set broken down by individual labels.

Table 10 illustrates the distribution of the AFTER and BEFORE labels in the first three partitions of the test data. The labels make up a larger part of the annotations at Level-1 and Level-2 than at Level-0.

Putting the pieces of the puzzle together, it is likely that the training instances in Level-0 are more useful for the OVERLAP label than the other two. CF-NoLevel0 is already able to classify these instances quite well, so not having access to Level-0 training instances does not affect it adversely.

We break down the performance of the classifiers on each partition of the test data in Figure 3. As expected, without access to Level-0 training instances, CF-NoLevel0 is slightly outperformed

Label	Distribution of labels (%)		
	Level-0	Level-1	Level-2
AFTER	10.1	21.2	23.6
BEFORE	5.1	13.7	16.1

Table 10: Distribution of labels in each partition.

(but not statistically so) by CF-Full for Level-0 test instances. Interestingly, the performance of CF-NoLevel0 for Level-2 and Level-3 instances are also behind that for CF-Full. This suggests that having additional Level-0 training instances can have an effect on classifier performance for Level-2 and Level-3 test instances. In future work we will like to investigate why this is the case.

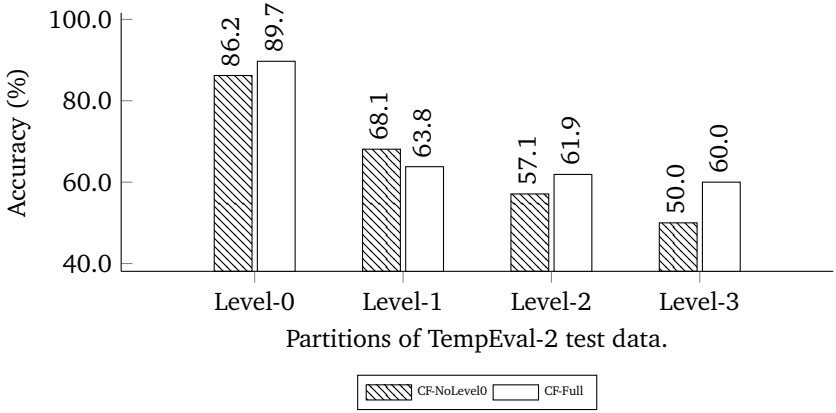

Figure 3: Breakdown of performance across different partitions.

6.2 Common Errors

We are also interested to investigate what more can be done to push system performance towards its upper-bound. Typically, such a theoretic upper-bound is derived from the inter-annotator agreement for the training set. The inter-annotator agreement for the annotations we have collected for d-full is 78.8%. Considering that the participants of the crowdsourcing exercise are not domain experts, the actual upper-bound could be higher. However this value is a good starting point for our analysis.

The best performing classifier we have presented in this work has an accuracy of 73.2%. There is still room for improvement towards our pessimistic upper-bound estimate. We study the misclassifications made, building a confusion matrix for CF-NoLevel0. Studying the matrix in Table 11, we observe two large clusters of errors.

Skewed dataset. First, BEFORE and AFTER instances are often mis-classified as OVERLAP. Reflecting on this, we believe the skewed training dataset where OVERLAP instances form a majority is one of the reasons why.

Considering the lower recall scores we get for BEFORE and AFTER labels, the mis-classifications are likely a direct result of a lack of suitable training instances within the training dataset to better identify BEFORE and AFTER instances.

Actual Label	Predicted label		
	OVERLAP	BEFORE	AFTER
OVERLAP	78	2	1
BEFORE	7	9	4
AFTER	13	0	14

Table 11: Confusion matrix for `CF-NoLevel0`.

Looking closer, we see that the performance on `BEFORE` instances is slightly lower than the performance for `AFTER` instances. We relate this to the smaller number of training instances available for the `BEFORE` label as seen from Table 10.

In future work it will be useful to verify if increasing the number of training instances available can help to improve the recall of the classifier for instances of these two labels, thereby eradicating this cluster of errors.

Feature design. The next major cause of mistakes is the misclassification of `BEFORE` instances as `AFTER`. Studying the mis-labeled instances, we realized this is because the features that we have used neglected to consider modal or copula modifications to the event expressions. For example, take a look at sentence 2[5]. The relationship between "added" and "*early November*" should have been `BEFORE` but was incorrectly classified as `AFTER`. The relevant portions of the dependency parse extracted as a feature for this instance is shown in Figure 4.

> He added that final guidelines to be published in *early November* will determine whether the bank is in compliance . (2)

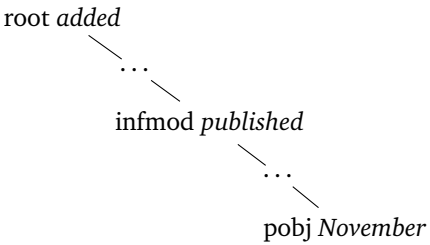

Figure 4: Dependency parse of Sentence 2.

The dependency parse feature that we extract is the shortest path between the vertices for "added" and "*November*". The key to interpreting the temporal relationship in this example however is the copula modifier "*to be*" in front of "published".

Without capturing these modifiers, the sentence will appear to read as "He added that final guidelines published in *early November* will determine . . . ". In this reading, "added" takes place `AFTER` the time span indicated by "*early November*", which explains the mis-classification by our classifier.

It will be useful following this analysis to study if including the auxiliary modifiers of event expressions into our parse features can help improve classifier performance.

[5]Extracted from document wsj_0527 of the TempEval-2 dataset.

Conclusion

In our study, we target the the main problem which we believe is hampering better performance for *event-temporal* relationship classification — lack of sufficient training data. We adopt a two-prong approach to tackle this problem: 1) by simplifying the feature space with the use of dependency parses as features to a convolution kernel support vector machine, and 2) by leveraging on crowdsourcing to get more data to support supervised machine learners.

We show that the classifier design we adopted is competitive when pitted against classifiers which make use of far more complex mechanics and features. With this as a starting point, we went on to expand the training data that is available for use via crowdsourcing. Despite the complexity of the annotation task, novice annotators are able to generate a dataset that helps improve classifier performance significantly.

Building on our insight of the clausal structure inherent to event and time expressions, we suggest an effective way to selectively acquire annotations. Our proposal reduces the amount of data to be annotated by up to 37% without sacrificing classifier performance. We achieved a classification accuracy of 73.2%, which represents a 8.6% improvement over our very competitive baseline.

Acknowledgments

This work is funded by the NExT Search Centre, which is supported by the Singapore National Research Foundation and Interactive Digital Media R&D Program Office, MDA under research grant (R-252-300-001-490).

References

Callison-Burch, C. and Dredze, M. (2010). Creating Speech and Language Data with Amazon's Mechanical Turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 1–12.

Collins, M. and Duffy, N. (2001). Convolution Kernels for Natural Language. In *Proceedings of NIPS*, volume 14, pages 625–632.

Derczynski, L. and Gaizauskas, R. (2010). USFD2: Annotating Temporal Expressions and TLINKs for TempEval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 337–340.

Ha, E. Y., Baikadi, A., Licata, C., and Lester, J. C. (2010). NCSU: Modeling Temporal Relations with Markov Logic and Lexical Ontology. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 341–344.

Harabagiu, S. and Bejan, C. A. (2005). Question Answering Based on Temporal Inference. In *Proceedings of the AAAI-2005 Workshop on Inference for Textual Question Answering*, pages 27–34.

Hsueh, P., Melville, P., and Sindhwani, V. (2009). Data Quality from Crowdsourcing: A study of Annotation Selection Criteria. In *Proceedings of the NAACL HLT 2009 Workshop on Active Learning for Natural Language Processing*, pages 27–35.

Ji, H., Grishman, R., and Dang, H. T. (2011). Overview of the TAC2011 Knowledge Base Population Track. In *Proceedings of the Text Analysis Conference (TAC)*.

Klein, D. and Manning, C. (2003). Accurate Unlexicalized Parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, volume 1, pages 423–430.

Kolya, A. K., Ekbal, A., and Bandyopadhyay, S. (2010). JU_CSE_TEMP: A First Step towards Evaluating Events, Time Expressions and Temporal Relations. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 345–350.

Mason, W. and Watts, D. (2009). Financial Incentives and the Performance of Crowds. In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, pages 77–85.

Mirroshandel, S. A., Ghassem-Sani, G., and Khayyamian, M. (2011). Using Syntactic-Based Kernels for Classifying Temporal Relations. *Journal of Computer Science and Technology*, pages 68–80.

Moschitti, A. (2006). Making Tree Kernels Practical for Natural Language Learning. In *Proceedings of European Chapter of the Association for Computational Linguistics*, volume 6, pages 113–120.

Munro, R., Bethard, S., Kuperman, V., Lai, V., Melnick, R., Potts, C., Schnoebelen, T., and Tily, H. (2010). Crowdsourcing and Language Studies: The New Generation of Linguistic Data. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 122–130.

Setzer, A. and Gaizauskas, R. (2000). Building a Temporally Annotated Corpus for Information Extraction. In *Proceedings of the LREC-2000 Workshop on Information Extraction Meets Corpus Linguistics*.

Setzer, A., Gaizauskas, R., and Hepple, M. (2003). Using Semantic Inferences for Temporal Annotation Comparison. In *Proceedings of the 4th International Workshop on Inference in Computational Semantics (ICoS-4)*.

Sheng, V., Provost, F., and Ipeirotis, P. (2008). Get another label? Improving data quality and data mining using multiple, noisy labelers. In *Proceeding of the 14th ACM International Conference on Knowledge Discovery and Data Mining*, pages 614–622.

UzZaman, N. and Allen, J. F. (2010). TRIPS and TRIOS System for TempEval-2: Extracting Temporal Information. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 276–283.

UzZaman, N., Llorens, H., Allen, J., Derczynski, L., Verhagen, M., and Pustejovsky, J. (2012). TempEval-3: Evaluating Events, Time Expressions, and Temporal Relations. *ArXiv e-prints*.

Verhagen, M., Gaizauskas, R., Schilder, F., Hepple, M., Katz, G., and Pustejovsky, J. (2007). SemEval-2007 Task 15: TempEval Temporal Relation Identification. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 75–80.

Verhagen, M., Sauri, R., Caselli, T., and Pustejovsky, J. (2010). SemEval-2010 Task 13: TempEval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 57–62.

Yoshikawa, K., Riedel, S., Asahara, M., and Matsumoto, Y. (2009). Jointly Identifying Temporal Relations with Markov Logic. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference On Natural Language Processing of the AFNLP*, pages 405–413.