

# Efficient Knowledge Acquisition for Extracting Temporal Relations

Son Bao Pham and Achim Hoffmann<sup>1</sup>

**Abstract.** Machine learning approaches in natural language processing often require a large annotated corpus. We present a complementary approach that utilizes expert knowledge to overcome the scarceness of annotated data. In our framework KAFTIE, the expert could easily create a large number of rules in a systematic manner without the need of a knowledge engineer. Using KAFTIE, a knowledge base was built based on a small data set that outperforms machine learning algorithms trained on a much bigger data set for the task of recognizing temporal relations. Furthermore, our knowledge acquisition approach could be used in synergy with machine learning algorithms to both increase the performance of the machine learning algorithms and to reduce the expert’s knowledge acquisition effort.

## 1 Introduction

Recent years have seen a growing interest in Natural Language Processing applications. Machine learning became the method of choice for building domain specific text analysis systems. Statistical NLP proved very popular as shallow approaches to language understanding appear more promising than previously thought. However, limitations of what those techniques can achieve are on the horizon: machine learning approaches are limited by the available tagged data, in particular, where supervised learning is involved.

There has been an increasing demand for temporal analysis. A number of approaches have addressed temporal processing: identification and normalization of time expressions [6], time stamping of event clauses [4], temporally ordering of events [5], recognizing time-event relations in TimeML [1]. At a higher level, these temporal expressions and their relations are essential for the task of reasoning about time, for example, to find contradictory information [3]. In this emerging domain, there is a clear lack of a large annotated corpus to build machine learning classifiers for detecting temporal relations.

We present our incremental knowledge acquisition approach that utilizes expert’s knowledge (almost any competent speaker of the relevant language) to overcome the scarceness of annotated data. We demonstrate the effectiveness of our approach as we have quickly developed a large knowledge base (KB) based on a small data set that performs better than machine learning (ML) approaches trained on a much bigger data set on the task of recognizing temporal relations.

We further show how ML algorithms can be used in synergy with our approach when annotated data is available to both increase the performance of the ML algorithms and reduce the knowledge acquisition effort. This work is based on [7, 8].

Our knowledge acquisition framework builds on the idea of Ripple Down Rules (RDR) [2] which allows one to add rules to the given KB

incrementally while automatically ensuring consistency with previously seen cases i.e the KB’s performance is continuously improved. Let us consider simple examples in the task of assigning a positive or negative class to sentences. Given the following sentence:

*Our approach has a good performance.*

the user could assign the positive class to the sentence due to the word *good* by creating the following rule:

**if the text contains “good” then assign positive class** (1)

The following sentence:

*The approach’s performance is not good.*

will be assigned with the positive class by rule (1). Suppose the user disagrees as the presence of the word *not* negates the effect of having the word *good*. He or she can in turn create an exception to rule (1):

**if the text contains “not” then assign negative class** (2)

This rule is potentially applicable only in the context where rule (1)’s condition is satisfied i.e. it is an exception of rule (1). Effectively, the above two rules are equivalent to:

**if the text contains “good” then assign positive class  
except if the text contains “not” then assign negative class**

For the sentence:

*The work does not follow traditional approaches.*

rule (1)’s condition is not satisfied. Rule (2) will therefore not be considered either. However, the sentence

*This approach is good and does not suffer from existing problems.*

will be labeled incorrectly with the negative class by rule (2). This is probably not the intention of the user when creating the exception rule (2) as the word *not* does not modify the meaning of the word *good*. The user could add another exception rule to refine rule (2):

**if “not” appears after “good” then assign positive class** (3)

The above examples use a very simple pseudo rule language to illustrate how the knowledge acquisition process works. The actual rule language description used is presented in detail in section 3.1.

In section 2 we present a short overview of RDR. This is followed by a description of KAFTIE in section 3. Section 4 reports experimental results on recognizing temporal relations which demonstrates the success of our approach. Section 5 discusses how ML could be utilized in KAFTIE. The conclusions follow in section 6.

<sup>1</sup> School of Computer Science and Engineering, University of New South Wales, Australia, emails: {sonp,achim}@cse.unsw.edu.au

## 2 Knowledge Acquisition Methodology

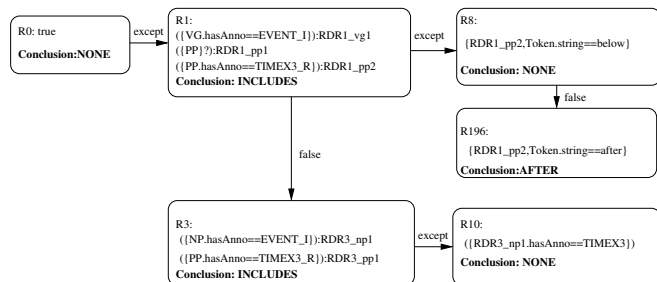
In this section we present the basic idea of Ripple-Down Rules (RDR) [2] which inspired our approach. RDR allows one to add rules to a knowledge base incrementally without the need of a knowledge engineer. A new rule is only created when the KB performs unsatisfactorily on a given case. The rule represents an explanation for why the conclusion should be different from the KB’s conclusion on the case at hand.

A *Single Classification Ripple Down Rules* (SCRDR) tree is a binary tree with two distinct types of edges. These edges are typically called *except* and *if-not* edges. See figure 1. Associated with each node in a tree is a *rule*. A rule has the form: *if*  $\alpha$  *then*  $\beta$  where  $\alpha$  is called the *condition* and  $\beta$  the *conclusion*.

Cases in SCRDR are evaluated by passing a case (an object to be classified) to the root of the tree. At any node in the tree, if the condition of a node  $N$ ’s rule is satisfied by the case, the case is passed on to the exception child of  $N$  using the *except* link if it exists. Otherwise, the case is passed on to the  $N$ ’s *if-not* child. The conclusion given by this process is the conclusion from the last node in the RDR tree which *fired* (satisfied by the case). To ensure that a conclusion is always given, the root node typically contains a trivial condition which is always satisfied. This node is called the *default* node. We will show how a more complex *default* rule can improve the knowledge acquisition process in section 5.

A new node is added to an SCRDR tree when the evaluation process returns the wrong conclusion. The new node is attached to the last node in the evaluation path of the given case with the *except* link if the last node is the *fired* rule. Otherwise, it is attached with the *if-not* link.

With the structure of RDR tree, it is possible that redundancy is present as one rule could appear in multiple places. Reorganizing RDR structure was reported in [11]. However, this is not crucial as the structure of the KB is transparent to the users.



**Figure 1.** An extract of a KB for recognizing TLINK between an EVENT and a TIMEX3. Note that SCRDR is different from a decision tree: rules in internal nodes can be used to give conclusions to input cases.

## 3 Our KAFITIE framework

We use SCRDR for building knowledge bases in the KAFITIE (Knowledge Acquisition Framework for Text classification and Information Extraction) framework. Some levels of abstraction in the rule’s condition is desirable to make the rule expressive enough in generalizing to unseen cases. To realize this, we use the idea of annotations where phrases that have similar roles (belong to the same concept) are deemed to belong to the same annotation type. Annotations contain the annotation type, the character locations of the beginning and ending position of the annotated text in the document, and a list of feature value pairs.

## 3.1 Rule description

A rule is composed of a condition part and a conclusion part. A condition is a regular expression pattern over annotations. It can also post new annotations over matched phrases of the pattern’s sub-components. The following is an example of a pattern which posts an annotation over the matched phrase:

$(\{Noun\}\{VG.type==FVG\}\{Noun\}):MATCH$

This pattern would match phrases starting with a Noun annotation followed by a VG, which must have feature *type* equal to FVG, followed by another Noun annotation. When applying this pattern on a piece of text, MATCH annotations would be posted over phrases that match this pattern. As annotations have feature value pairs, we can impose constraints on annotations in the pattern by requiring that a feature of an annotation must have a particular value.

The rule’s conclusion contains the classification of the input text. In the task of recognizing temporal relations between a pair of temporal expressions (event or time), the conclusion is either the relation type or NONE.

## 3.2 Annotations and Features

**Built-in annotations:** As our rules use patterns over annotations, the decision on what annotations and their corresponding features should be are important for the expressiveness of rules. The following annotations and features make patterns expressive enough to capture all rules we want to specify for various tasks.

We have Token annotations that cover every token with *string* feature holding the actual string, *category* feature holding the POS and *lemma* feature holding the token’s lemma form.

As a result of the Shallow Parser module, which is a cascade of finite state transducers, we have several forms of noun phrase annotations ranging from simple to complex noun phrases, e.g. NP (simple noun phrase), NPList (list of NPs) etc. There are also VG (verb groups) annotations with *type*, *voice*, *headVerbPos*, *headVerbString* etc. features and other annotations e.g. PP (prepositional phrase), SUB (subject), OBJ (object). The Shallow Parser used in this work is not geared towards the task.

Every rule that has a non-empty pattern would post at least one annotation covering the entire matched phrase. Because rules in our knowledge base are stored in an exception structure, we want to be able to identify which annotations are posted by which rule. To facilitate that, we number every rule and enforce that all annotations posted by rule number  $x$  have the prefix RDRx\_. Therefore, if a rule is an exception of rule number  $x$ , it could use all annotations with the prefix RDRx\_ in its condition pattern.

Apart from the requirement that an annotation’s feature must have a particular value, we define extra annotation constraints: *hasAnno*, *hasString*, *underAnno*, *endsWithAnno*. For example, *hasAnno* requires that the text covered by the annotation must contain the specified annotation:

$NP.hasAnno == TIMEX3$

only matches NP annotations that have a TIMEX3 annotation covering their substring. This is used, for example, to differentiate a TIMEX3 in a noun group from a TIMEX3 in a verb group.

**Custom annotations:** Users could form new named lexicons during the knowledge acquisition process. The system would then post a corresponding annotation over every word in those lexicons. Doing this makes the effort of generalizing the rule quite easy and keeps the KB compact.

### 3.3 The Knowledge Acquisition Process in KAFTIE

The knowledge acquisition process goes through a number of iterations. The user gives a text segment (e.g. a sentence) as input to the KB. The conclusion (e.g. classification) is suggested by the KB together with the *fired* rule  $R$  that gives the conclusion. If it is not the default rule, annotations posted by the rule  $R$  are also shown (see section 4.3) to help the user decide whether the conclusion is satisfactory.

If the user does not agree with the KB's performance, there are two options: adding an exception rule to rule  $R$  or modifying rule  $R$  if possible. In either case, user's decision will be checked for consistency with the current KB before it gets committed to the KB. To create a new exception rule, the user only has to consider why the current case should be given a different conclusion from rule  $R$ . This effort does not depend on the knowledge base size. The level of generality of the new rule is not crucial as this process can be supported by the work reported in [9].

## 4 Experiments

We build a knowledge base using KAFTIE to recognize TLINK relations between an EVENT and a TIMEX3 in the TimeBank corpus. According to the specification of TimeML [10], TIMEX3 marks up explicit temporal expressions such as times, dates, durations etc. EVENT covers elements in a text describing situations that occur or happen while TLINK is a temporal link representing a relation between an event and a time or between two events.

### 4.1 The TimeBank corpus

The TimeBank corpus is marked up for temporal expressions, events and basic temporal relations based on the specification of TimeML. Currently, the TimeBank corpus has 186 documents.

Excluding TIMEX3s in document's meta-data (doc creation time), the majority of TLINKs are between EVENTS and TIMEX3s within the same sentence. Hence, in all experiments, we focus on recognizing intra-sentential temporal relations.

The TimeBank annotation guidelines suggest distinctions among TLINK types between two EVENTS but do not explicitly specify how those types are different when it comes to relations between an EVENT and a TIMEX3. In fact, some of the TLINKs types between an EVENT and a TIMEX3 are hard to distinguish and a number of cases of inconsistency are observed in the corpus. We group similar types together: BEFORE and IBEFORE are merged, AFTER and IAFTER are merged and the rest is grouped into INCLUDES.

### 4.2 KAFTIE for extracting Temporal Relations

For the task of extracting EVENT-TIMEX3 temporal relations, we consider all pairs between an EVENT and a TIMEX3 in the same sentence and build a knowledge base to recognize their relations. The sentence containing the pair is used as the input to the knowledge base. As there could be more than one EVENT or TIMEX3 in the same sentence, we change the EVENT and the TIMEX3 annotations in focus to EVENT\_I (instance) and TIMEX3\_R (related.to) annotations respectively. This enables our rule's pattern to uniquely refer to the pair's arguments. For each pair, the KB's conclusion is the type of its temporal relation if there exists a relation between the pair's arguments or NONE otherwise.

Apart from this, there was no prior design required to adapt KAFTIE to the task of recognizing temporal relations.

### 4.3 How to build a Knowledge Base in KAFTIE

The following examples are taken from the actual knowledge base discussed in section 4.4 and shown in figure 1. Suppose we start with an empty KB for recognizing temporal relations between an EVENT and a TIMEX3 within the same sentence. I.e. we would start with only a default rule that always produces a NONE conclusion. When the following sentence is encountered:

```
Imports of the types of watches [VG [EVENT_I totaled EVENT_I] VG]
[PP about $37.3 million PP] [PP in [NP [TIMEX3_R 1988 TIMEX3_R]
NP] PP], ...
```

our empty KB would use the default rule to suggest the relation between EVENT\_I and TIMEX3\_R is NONE i.e. no relation exists. This can be corrected by adding the following rule to the KB:

```
Rule 1:
(({VG.hasAnno==EVENT_I}):RDR1_vg1
({PP }?):RDR1_pp1
({PP.hasAnno == TIMEX3_R}):RDR1_pp2
):RDR1_
Conclusion: INCLUDES
```

Each of the component in the rule's pattern is automatically assigned a tag which will effectively post a new annotation over the matched token strings of the component if the pattern matches a text. New tags of rule  $i$  always start with  $RDRi_$ . This rule would match phrases starting with a VG annotation which covers the EVENT\_I annotation, followed by an optional PP annotation followed by a PP annotation covering the TIMEX3\_R annotation. When the sentence containing the pair EVENT\_I-TIMEX3\_R is matched by this rule, the pair is deemed to be of INCLUDES relation type. Once matched, new annotations RDR1\_vg1, RDR1\_pp1 and RDR1\_pp2 will be posted over the first VG, the first PP and the last PP in the pattern respectively. This is to enable exception rules to refer to the results of previously matched rules. Notice here that the first PP component is specified optional. It could be that the expert already *anticipates* future cases and makes the current rule more general. Alternatively, experts always have a choice of modifying existing rule to cover new cases.<sup>2</sup> When we encounter this pair:

```
The company's shares [RDR1_vg1 [VG are [EVENT_I] wallowing
EVENT_I] far VG] [RDR1_vg1] [RDR1_pp2] [PP below [NP their
[TIMEX3_R 52-week TIMEX3_R] NP] PP] [RDR1_pp2] high....
```

Rule **R1** fires suggesting that the pair has INCLUDES relation with the newly posted annotation highlighted in bold face. This is incorrect as there is no relation. The following exception rule is added to fix the misclassification:

```
Rule 8:3
({RDR1_pp2.Token.string==below }):RDR8_
Conclusion: NONE
```

This rule says that if the second PP matched by Rule 1 (RDR1\_pp2) starts with a token string *below* then there is no relation between the pair. Notice that the sentence could have different PP annotations. As each rule posts unique annotations over the matched phrases, we can unambiguously refer to relevant annotations.

<sup>2</sup> Automatic recommendation on which existing rules and how to modify to cover new cases is reported in [9].

<sup>3</sup> We only select some rules to show as examples, hence indices of rules are not consecutive

## 4.4 Experimental results

Out of 186 documents in the TimeBank corpus with approximately 1350 intra-sentential TLINK instances, we randomly took half of that as training data and keep the remaining half for testing. Using our KAFTIE framework, we built a knowledge base of 229 rules to recognize relations between an EVENT and a TIMEX3 in the same sentence using the training data.<sup>4</sup> The knowledge base uses NONE as its default conclusion. In other words, by default and initially when the KB is empty, all EVENT-TIMEX3 pairs are deemed not to have any TLINK relations.

The overall results are shown in table 1 for two settings: 3 types (BEFORE, INCLUDES and AFTER see section 4) and *without typing* - collapsing all TLINK types into one type, effectively detecting if the pair has a TLINK relation regardless of the type. While the accuracy reflects performance on the test data across all types including NONE, the F-measure is based on only TLINKs types, i.e. excluding NONE.<sup>5</sup> On the *without typing* setting, the built knowledge base achieved an F-measure of more than 75% and an accuracy of 86.7%.

Methods	3 types		w/o typing	
	F-m	Acc.	F-m	Acc.
KAFTIE	<b>71.3%</b>	<b>86.1%</b>	<b>75.4%</b>	<b>86.7%</b>
J48 (5-folds)	62.3%	78.7%	66.4%	81.2%
SMO (5-folds)	61.4%	77.4%	63.1%	78.7%

**Table 1.** Results on recognizing TLINKs for 3 types and *without typing* settings.

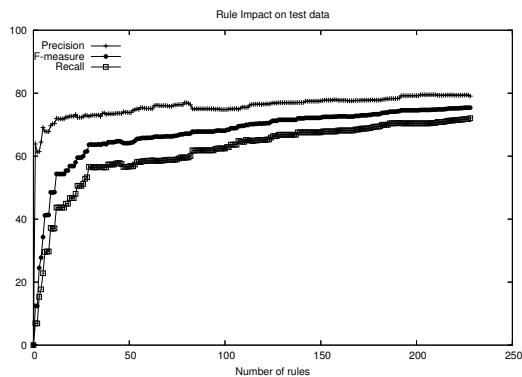
For comparison with standard machine learning approaches, we use Weka’s J48 and SMO [12] as implementations for C4.5 and support vector machine algorithms respectively. To adapt machine learning algorithms to the task of extracting relations, we define the following feature representation which is capable of capturing the relation arguments (EVENTs and TIMEX3s) and the surrounding context. We break up the sentence containing the EVENT-TIMEX3 pair into five segments namely: spans of the two arguments (EVENT and TIMEX3), the span between the two arguments and spans to the left/right of the left/right arguments. From each segment, we use token strings, token lemmas, parts-of-speeches, bigrams of parts-of-speeches and all annotations from the Shallow Parser as features.

J48 and SMO are run using 5-fold cross validation. As the result could vary depending on the seed used for the cross validation, we report results averaged over 100 runs with different random seeds. As can be seen in table 1, the knowledge base built using our framework significantly outperforms standard J48 and SMO. In fact, our knowledge base with the initial 60 rules (as the result of seeing roughly 60 misclassified TLINK pairs) already outperforms J48 and SMO (see figure 2).

Figure 2 shows the performance of our knowledge base on the test data as rules are incrementally added. Given the upwards trend of the graph as more rules are added, it is plausible that our KB would get to even higher performance had we used 80% of the available data for building the KB.

<sup>4</sup> To evaluate the TLINK recognition task alone, we use the EVENT and TIMEX3 annotations in the TimeBank corpus.

<sup>5</sup> The NONE type occurs approximately 2.5 times more often than the TLINK types.



**Figure 2.** Impact of incrementally adding rules to the KB.

## 5 Combining ML with KA

In this section, we investigate how machine learning could be used in synergy with our knowledge acquisition approach to both improve the machine learning algorithms’ performance and reduce the required knowledge acquisition effort.

A knowledge acquisition (KA) session corresponds to the creation of a new rule as a result of the KB performing incorrectly on a particular case. Notice that the KB’s default rule is quite simple. It always returns the default conclusion which is *NONE* for the task of recognizing temporal relations. We conjecture that if we start with a better default rule then the number of knowledge acquisition sessions can be reduced. One way of doing it is to take some training data to train a classifier as the default rule.

We carry out the following experiment: We focus on the task of recognizing temporal relation in the *without typing* setting and use the exact training and test data from the previous section to build and test a KB respectively. The difference is that we now split the training data into two parts i.e. *ML data* and *KB data*. The *ML data* is used to train a classifier as the default rule in a KB while the *KB data* is used to add exception rules to the KB.

Instead of having real experts involved in the process of creating exception rules, we simulate it by consulting the KB built from the previous section, called *oracleKB*.<sup>6</sup> For each example in the *KB data* that is misclassified by the current KB, we use the *fired* rule for the example from the *oracleKB* i.e. the rule that the *oracleKB* would use on the example.

Table 2 shows the results of using J48 and SMO which are trained on varying portions of the training data. Specifically, it shows the f-measure of the classifier alone, the KB with the classifier as the default rule on the test data as well as the number of rules in the KB. The number of rules in the KB reflects the number of KA sessions required for building the KB. All figures are averaged over 20 different runs using different seeds on splitting the training data into *ML data* and *KB data*.

In all situations, machine learning algorithms can always be augmented with more rules using our knowledge acquisition approach to achieve a significantly better performance, see figure 3.

As we increase the percentage of *ML data*, the number of KA sessions required gets smaller. Ideally, we want to minimize the number of KA sessions while maximizing the f-measure. When the *ML data* is empty (0% of the training data), we effectively rebuild a KB in

<sup>6</sup> Every rule in this KB is made independent of other rules by incorporating all of its context e.g. the rule it is an exception of.

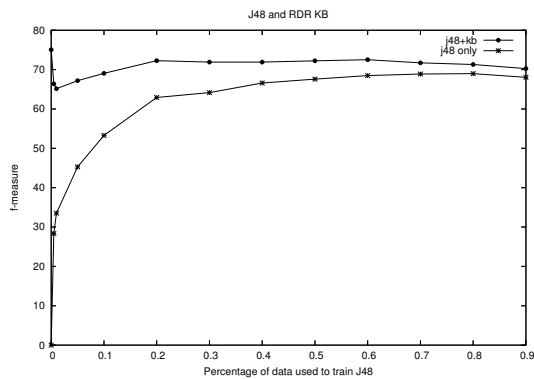


Figure 3. F-measures of J48 and J48 with KB.

the same fashion as in the previous section with different orders of examples. The f-measure of 75.1% suggests that the performance of our approach is independent of the order of examples presented to the experts.

As can be seen from table 2, the f-measures of the KBs with SMO or J48 as the default rule follow the same trend. It first degrades as we start giving some data to the classifier and improves as more data is used to train the classifier. After certain thresholds, the f-measures start degrading again as we get less data for experts to add exception rules while the classifiers do not improve their performance.

Depending on the task and the classifier used, we can choose an appropriate amount of data to train a classifier as the default rule in order to substantially reduce the number of KA sessions involved while still achieve a reasonable performance. For example with J48 the best performance is at 72.5% when we use 60% of the training data for training J48 and the rest to add exceptions rules. This reflects a 4% improvement in f-measure to the raw machine learning classifier. Compared to a fully manual approach (using 0% data for the classifier), we achieve a 60% reduction in the number KA sessions.

As the *oracleKB* is built with the assumption that the default rule always returns *NONE*, all of the exception rules at level 1 (exception rules of the default rule) are rules covering *INCLUDES* instances. Even though rules at level 2 cover *NONE* instances, they have limited scopes because they were created as exceptions to level 1 rules. When the default rule gives an incorrect *INCLUDES* conclusion, it is likely that we would not be able to consult the *oracleKB* for an exception rule to fix this error.<sup>7</sup> It therefore suggests that if we use real experts, we could achieve a better result.

## 6 Discussion and Conclusions

We have shown that with our unconventional Knowledge Acquisition approach using KAFTIE, we could quickly build a knowledge base that performs better than the classifiers built by existing machine learning approaches while requiring much less data.

It took 7 minutes on average to create one rule, independent on the current size of the KB. This includes the time needed to read the sentence to understand why there is a certain relation between the pair of an EVENT and a TIMEX3 as well as the time required to test the new rule before committing it to the KB. If the users have to classify the pair's relation from scratch, when we do not have annotated data,

<sup>7</sup> A better simulation is to build another *oracleKB* with the default rule always returning the *INCLUDES* conclusion.

%	j48	j48 +kb	#rules	smo	smo +kb	#rules
0%	0	<b>75.1</b>	173	0	<b>75.1</b>	173
0.5%	28.4	<b>66.4</b>	146	27	<b>68.6</b>	151
1%	33.5	<b>65.2</b>	143	27.3	<b>71.6</b>	158
5%	45.3	<b>67.2</b>	138	54.3	<b>71.9</b>	142
10%	53.3	<b>69</b>	131	61.7	<b>72.4</b>	133
30%	64.2	<b>71.9</b>	103	65.6	<b>72.7</b>	108
50%	67.6	<b>72.2</b>	78	66.6	<b>71.8</b>	84
60%	68.5	<b>72.5</b>	65	67	<b>71.7</b>	70
90%	68	<b>70.3</b>	22	66	<b>68.5</b>	25

Table 2. Results of using j48/smo as the default rule for KBs averaged over 20 random seeds.

then the actual time spent on creating a rule would be much less, as understanding the sentence takes most of the time.

We further demonstrated how ML algorithms could be used together with our approach to reduce the knowledge acquisition effort. This also results in performance improvement to the ML algorithms.

In future work, we will investigate the application of our framework KAFTIE to tasks in a different language other than English.

## REFERENCES

- [1] Branimir Boguraev and Rie Kubota Ando, 'TimeML-compliant text analysis for temporal reasoning', in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 997–1003, UK, (2005).
- [2] Paul Compton and R. Jansen, 'A philosophical basis for knowledge acquisition', *Knowledge Acquisition*, 2, 241–257, (1990).
- [3] Richard Fikes, Jessica Jenkins, and Gleb Frank, 'JTP: A system architecture and component library for hybrid reasoning.', in *Proceedings of the 7th World Multiconference on Systemics, Cybernetics and Informatics*, Orlando Florida, USA, (2003).
- [4] Elena Filatova and Eduard Hovy, 'Assigning time-stamps to event-clauses', in *Proceedings of the 10th Conference of the EACL*, Toulouse, France, (2001).
- [5] Inderjeet Mani, Barry Schiffmann, and Jianping Zhang, 'Inferring temporal ordering of events in news', in *Proceedings of the NAACL*, Edmonton, Canada, (2003).
- [6] Inderjeet Mani and George Wilson, 'Robust temporal processing of news', in *Proceedings of the 38th annual meetings of the ACL*, Hong Kong, (2000).
- [7] Son Bao Pham and Achim Hoffmann, 'Incremental knowledge acquisition for building sophisticated information extraction systems with KAFTIE', in *5th International Conference on Practical Aspects of Knowledge Management*, pp. 292–306, Vienna, Austria, (2004). Springer-Verlag.
- [8] Son Bao Pham and Achim Hoffmann, 'Incremental knowledge acquisition for extracting temporal relation.', in *Proceedings of 2nd IEEE International Conference on Natural Language Processing and Knowledge Engineering (NLP-KE05)*, pp. 354–359, Wuhan, China, (2005). IEEE.
- [9] Son Bao Pham and Achim Hoffmann, 'Intelligent support for building knowledge bases for natural language processing (in print).', in *Perspectives of Intelligent Systems' Assistance*, ed., Roland Kaschek, Idea Group Inc., (2006).
- [10] J. Pustejovsky, J. Castano, R. Ingria, R. Sauri, R. Gaizauskas, A. Setzer, G. Katz, and D. Radev, 'TimeML: Robust specification of event and temporal expressions in text.', in *AAAI Spring Symposium on New Directions in Question Answering.*, Stanford, CA, (2003).
- [11] Hendra Suryanto, *Learning and Discovery in Incremental Knowledge Acquisition.*, Ph.D. dissertation, University of New South Wales, Australia, 2005.
- [12] Ian H. Witten and Eibe Frank, *Data Mining: Practical machine learning tools with Java implementations*, Morgan Kaufmann, 2000.