Feature Selection Using a Semantic Hierarchy for Event Recognition and Type Classification

Yoonjae Jeong and Sung-Hyon Myaeng
Korea Advanced Institute of Science and Technology (KAIST)
291 Daehak-ro (373-1 Guseong-dong), Yuseong-gu, Daejeon 305-701,
Republic of Korea
{hybris, myaeng}@kaist.ac.kr

Abstract

Event recognition and event type classification are among the important areas in text mining. A state-of-the-art approach utilizing deep-level lexical semantics and syntactic dependencies suffers from a limitation of requiring too large feature space. In this paper, we propose a novel feature selection method using a semantic hierarchy of features based on WordNet relations and syntactic dependencies. Compared to the well-known feature selection methods, our proposed method reduces the feature space significantly while keeping the same level of effectiveness. For noun events, it improves effectiveness as well as efficiency. Moreover, we expect the proposed feature selection can be applied to the other types of text classification using hierarchically organized semantic resources such as WordNet.

1 Introduction

Feature selection is an important issue in text-based classification because features can be generated in a number of different ways from text. Selecting features affects not only efficiency when the space is big but also classification effectiveness by eliminating noise features (Manning, Raghavan, & Schütze, 2008). In this paper, we propose a new feature selection method that utilizes semantic aspects of word features and discuss its relative merits compared to other well-known feature selection methods.

Among many text-based classification problems, this research focuses on event recognition (a kind of binary classification) and type classification that have been studied extensively to improve performance of applications such as automatic summarization (Daniel, Radev, & Allison, 2003) and question answering (Pustejovsky, 2002). For event recognition and type classification, TimeML has served as a representative annotation scheme of events (Pustejovsky, Castaño, et al., 2003), which are defined as situations that happen or occur and expressed by verbs, nominalizations, adjectives, predicative clauses or prepositional phrases. TimeML defines seven types of events, REPORTING, PERCEPTION, ASPECTUAL, I_ACTION, I_STATE, STATE, and OCCURRENCE (Pustejovsky, Knippen, Littman, & Saurí, 2007), to which a recognized event text is classified for event type classification.

Different approaches to recognize and classify TimeML events have been proposed, ranging from rule-based approaches (Saurí, Knippen, Verhagen, & Pustejovsky, 2005) to supervised machine learning techniques based on lexical semantic classes and morpho-syntactic information around events (Bethard & Martin, 2006; Boguraev & Ando, 2007; Jeong & Myaeng, 2013; Llorens, Saquete, & Navarro-Colorado, 2010). Jeong & Myaeng (2013) recently showed that using the deeper-level of semantics increased the performance. They obtained the best performance in their classification experiments when lexical semantic features using hypernyms at the maximum depth of eight in WordNet were used for the event candidates and the words having syntactic dependency. While the approach showed a meaningful improvement, it has a problem of generating too many features.

Semantic features that can be mapped to a structure like WordNet have hierarchical relationships. In this situation, when two features

International Joint Conference on Natural Language Processing, pages 136–144,
Nagoya, Japan, 14-18 October 2013.

have a hypernym-hyponym relationship, the higher-level feature encompasses the lower-level one (see Figure 1-(a)). If a conventional feature selection method were used, therefore, the selected features would include both overly specific, low-level features and more general ancestors that cover the characteristics of the children (see Figure 1-(b)). When the general features are accurate and specific enough to represent the class, their descendants are unnecessary and redundant. When redundant features of similar kind are used, they cause not only efficiency problems but also potential overfitting of the model because the resulting model may become biased towards the semantics covered by the sub-tree containing the features.

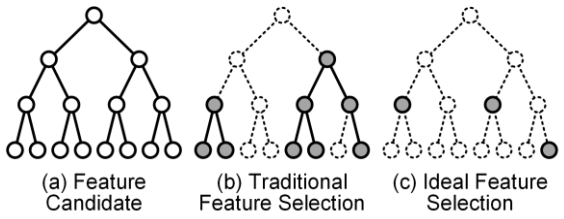


Figure 1. Feature Selection in Hierarchical Feature Space

It is important to select the features that are sufficiently general to encompass more specific features found in the training data but specific enough to utilize deep-level semantics available in the hierarchy (see Figure 1-(c)). The leftmost feature in (c) covers the semantics of the two features under it without having to keep them. Choosing the feature in the center and the rightmost feature has a similar effect and at the same time avoids using the overly general feature that encompasses both as well as the sibling of the rightmost one, which is not an appropriate one. In other words, we should select as general a feature as possible as long as none of them are considered irrelevant for the class, thereby it can cover the semantics of the features underneath it, without which we can achieve better efficiency.

In short, we propose a method for solving the problem of using features that are semantically redundant. Assuming that all the features can be organized in the form of a hierarchy, the method attempts to select the features that are as specific as possible as long as there are no semantically redundant features.

2 Event Recognition and Type Classification Task

We first describe the task for recognition and type classification of TimeML events. For word-based event recognition and type classification, we converted the phrase-based annotations into a form with BIO-tags. For each word in a document, we assign a label indicating whether it is inside or out-side of an event (i.e., BIO2[1] label) as well as its type. For type classification, in addition, each word must be classified into one of the known event classes. Figure 2 illustrates an example of chunking and labeling components of an event in a sentence.

Word	Event Label	Event Type Label
All	O	O
75	O	O
people	O	O
on	B-EVENT	B-STATE
board	I-EVENT	I-STATE
the	O	O
Aeroflot	O	O
Airbus	O	O
died	B-EVENT	B-OCCURRENCE
.	O	O

Figure 2. Event chunking for a sentence, "*All 75 people on board the Aeroflot Airbus died.*" B-EVENT, I-EVENT and O refer to the beginning, inside and outside of an event.

Our method consists of three parts: preprocessing, feature extraction and selection, and classification. The preprocessing part analyzes raw text for tokenization, PoS tagging, and syntactic parsing (dependency parsing). It is done by the Stanford CoreNLP package[2], which is a suite of natural language processing tools. Then, the feature extraction part converts the preprocessed data into the feature space, followed by feature selection. Finally, the classification part determines whether the given word is an event or not and its type using a maximum entropy (ME) classifier.

3 Feature Candidate Generation

Because the goal of the proposed method is to automatically select the most valuable features, we generate feature sets based on the same criteria of Jeong & Myaeng's work (2013), which showed better performance for TimeML event than the state-of-the-art approach. The details are below:

[1] IOB2 format: (B)egin, (I)nside, and (O)utside
[2] Stanford CoreNLP, http://nlp.stanford.edu/software/corenlp.shtml

Lexical Semantic Features (LSF). The set of target words' lemmas and their all-depth WordNet semantic classes (i.e., hypernyms). For example, a noun "*drop*" that is mapped to such a WordNet class is always an event regardless of its context in a sentence in the TimeBank corpus (Pustejovsky, Hanks, et al., 2003).

Windows Features (WF). The lemma, hypernyms, and PoS of the context defined by a five-word window [-2, +2] around a target word.

Dependency-based Features (DF). They are similar with WF, but the context is defined by syntactic dependencies. This feature type differs from WF because the context may go beyond the fixed size window and the features are not just words. Increasing the window size for WF instead of using this feature type is not an option because it would end up including some noise by including too big a context. Four dependencies we consider are: subject (SUBJ), object (OBJ), complement (COMP), and modifier (MOD).

- **SUBJ type.** A feature is formed with the governor or dependent word and its hypernyms that has the SUBJect relation (*nsubj* and *nsubjpass*) with the target word.
- **OBJ type.** It is the governor or dependent word and its hypernyms, which has the OBJect relation (*dobj*, *iobj*, and *pobj*) with the target word. In "*... delayed the game ...*", for instance, the verb "*delay*" can describe the temporal state of its object noun, "*game*".
- **COMP type.** It indicates the governor or dependent word and its hypernyms, which has the COMPlement relation (*acomp* and *xcomp*) with the target word. In "*... called President Bush a liar ...*", for example, the verb "*called*" makes the state of its object ("*Bush*") into the complement noun, "*liar*". In this case, the word "*liar*" becomes a STATE event.
- **MOD type.** It refers to the dependent words and their hypernyms in MODifier relation (*amod*, *advmod*, *partmod*, *tmod* and so on). This feature type is based on the intuition that some modifiers such as temporal expression reveal the word it modifies has a temporal state and therefore is likely to be an event.

Combined Features (CF). They are a combination of LSF and DF (or WF). A certain DF may not be an absolute clue for an event by itself but only when it co-occurs with a certain lexical or semantic aspect of the target word.

4 Feature Selection Based on Semantic Hierarchy

Since a large number of features are generated with the aforementioned feature generation method, it is necessary to filter out those whose roles in classification are minimal. We first remove the feature candidates whose frequency in the training data is less than two. If a target word containing the feature candidate is determined not to be an event more than 50% in the training data, it is also eliminated. The remaining feature candidates are then organized into a meaning hierarchy so that we can apply the tree-based feature selection method.

An entailment relationship between two features, $f_i >> f_j$, is established by a hypernym/hyponym relationship, syntactic dependency, or occurrence sequence as in Table 1. A and D represent an ancestor and a descendent in a feature hierarchy tree with A >> D. We call the LSF and DF (or WF) features in CF as target and context elements, respectively. LSF can be an ancestor of CF because LSF does not consider the surrounding context of a target word whereas CF includes the context. CF_{LD} and CF_{LW} mean CF of LSF and DF and CF of LSF and WF, respectively.

A	D	Condition
LSF	LSF	A is hypernym of D.
LSF	CF_{LD} or CF_{LW}	A is synset/hypernym of target in D. e.g.) $process_{LSF}$ >> ($report_{DF}$, $\underline{process}_{LSF}$)
DF	DF	Same dependency type. A is the hypernym of D.
DF	CF_{LD}	Same dependency with the target. A is synset/hypernym of surrounding in D. e.g.) $report_{DF}$ >> ($report_{DF}$, $process_{LSF}$)
WF	WF	Same position from target. A is the hypernym of D.
WF	CF_{LW}	Same position from target. A is the synset/hypernym of the context in D. e.g.) $before_{WF}$ >> ($before_{WF}$, $launch_{LSF}$)
CF_{LD}	CF_{LD}	Same dependency with target. The target and the context of A

A	D	Condition
		are the synset/hypernym of those of D, respectively.
CF_{LW}	CF_{LW}	Same position from target. The target and the context of A are the synset or hypernym of those of D, respectively.

A: Ancestor, **D:** Descendant (A >> D)

Table 1. Entailment Relation of Features

4.1 Feature Tree Generation

Given that the entailment relationship >> can be established between two features, we can construct a feature tree that becomes a basis for tree-based feature selection. We begin with a tree that only has a root node R, a meta-feature that is the ancestor of all features. R entails and keeps adding new features to the tree until all the features are added to the tree. We define *a*, *d*, and *c* for ancestor, descendent, and child features with the relationships $a >> d$ and $a > c$ where > means *c* is a child of *a*, restricting that there is no node between *a* and *c* with $a >> c$. Figure 3 illustrates the detail algorithm of feature tree generation.

When a new feature f is added to the (sub-)tree whose root is *a* and $a >> f$, *f* either becomes a child of *a* or is added to one of the sub-trees of *a* (line 9~28). If there is *c* such that $c >> f$, *f* is added to a subtree whose root is *c* (line 14~17). On the other hand, if $f >> c$, *f* replaces *c*, and *c* is entered to the sub-tree whose root is *f* (line 19~25). Finally if *f* has no entailment relation with any of the children nodes of *a*, *f* is added as a child of *a* (line 26~27).

```
 1 program GenerateTree;
 2 F := feature candidates set;
 3 r := root of feature tree;
 4 begin
 5   for f in F do
 6     add_feature(r, f);
 7 end;
 8
 9 procedure add_feature (a, f)
10 a : ascendant feature;
11 f : new feature;
12 begin
13   for c of a's children
14     if f is descendant of c then
15       begin
15         add_feature (c, f);
16         break;
17       end
18     else
19       if c is descendant of f then
20         begin
21           remove c from a's chil-
dren;
22           add f to a's children;
23           add_feature (f, c);
24           break;
25         end;
26   if no child of a is ancestor or
descendant of f then
27     a's children <- f;
28 end.
```

Figure 3. Feature Tree Generation Algorithm

4.2 Tree-Based Feature Selection

The key idea of the selection algorithm we devised is to evaluate each of the paths in the tree and select the appropriate node (i.e. feature). A path is defined to be the list of nodes between the root and a leaf node. In essence, the problem of selecting nodes or features from a tree is converted into smaller problems of selecting a node from individual paths. The process is illustrated with Figure 4 where each node of the tree except the root represents a feature. The tree has *n* paths corresponding to the number of leaf nodes. The algorithm selects the most representative node on a path, which is marked with a black node in Figure 4.

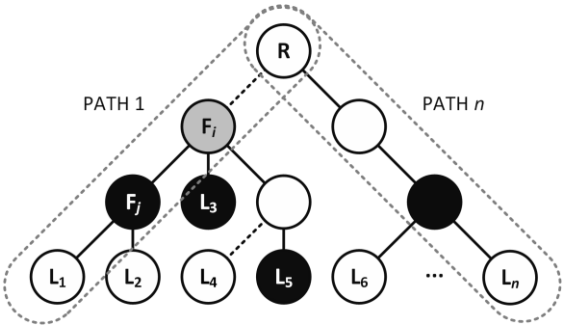


Figure 4. Paths between the root and the leaf nodes in a feature tree

To select the most representative feature on a path, we employed the notion of *lift*, which has been used widely in the area of association rule mining to compute the degree to which two items are associated (Tufféry, 2011). More specifically, it is defined as Equation (1) where $P(f)$ indicate the probability of a feature *f* in training data set. $P(E \mid f)$ is the conditional probability of events occurring given that *f* occurs.

$$lift(f) = \frac{P(E|f)}{P(f)} \quad (1)$$

While general feature selection methods such as χ^2 are based on the degree of belief, our selection method considers the reliability and applicability (or generality) of a feature. In other words, a feature we choose should have a high *lift* value (i.e., high reliability) and lie closest to the root on a path so that we can broaden its applicability.

These criteria would be particularly true when the amount of training data is not sufficient.

However, selecting the feature at the highest level in the tree may not be the best choice. In Figure 4, for example, even if the node F_i in grey is determined to be the most representative one for the path 1, it may not be the best one. In this case, F_j may be a better one because it happens to be the representative node for the path between F_i and L_1. However, there is a chance that the sub-tree of F_i may have important features (i.e., L_3, L_5) that end up elevating F_i's weight unfairly. Instead of F_i, using F_j would be a better choice.

In order to handle this problem, we developed an algorithm where the key idea works as in Figure 5. We first collect all the representative features from the paths based on the reliability and generality criteria mentioned above (line 29~45). For each representative node, we check if any of the descendant nodes have been selected as a representative node of other paths (line 21). If the condition is met, the node is no longer considered as a representative node (line 23). The same process is applied to the sub-tree whose root is the node just deleted from the set of representative nodes (line 25). Up to now, this process does not require manually checking the performance for the selected features.

```
 1 program SelectFeatures;
 2 T := feature tree
 3 F := selected feature set
 4 begin
 5   F ← ∅;
 6   select_features(T);
 7 end;
 8
 9 procedure select_from_tree
10 t := subtree of T
11 begin
12   r ← root of t
13   L ← leaf features of t
14   for l of L
15     p ← path from r to l;
16     f ← select_from_path (p);
17     add f to F;
18   end;
19   for f of F
20     D ← descendants of f;
21     if {d | d ∈ D and d ∈ F} ≠ ∅
   then
22       begin
23         remove f from F;
24         t_f ← subtree of t whose root
   is f
25         select_from_tree (t_f);
26       end;
27 end.
28
29 procedure select_from_path
30 p := feature path
31 begin
32   cur ← front of p
33   While
34     next ← next of cur
35     if next is null then
36       begin
37         add cur to F;
38         return;
39       end;
40     if lift(cur) ≥ lift(next) then
41       begin
42         add cur to F;
43         return;
44       end;
45 end.
```

Figure 5. Tree-Based Feature Selection Algorithm

We select the final features among those obtained through the above process by employing a widely used feature selection method (in our case, χ^2). It is because the most representative feature in a path might not be effective one in the entire feature space.

5 Experiment

5.1 Experimental Setup

The main goal of the experiment is to examine the efficacy of the proposed tree-based feature selection method in the context of event recognition and event type classification. For test collection, we use the TimeBank 1.2 corpus (Pustejovsky, Hanks, et al., 2003), which is the most recent version of TimeBank, annotated with the TimeML 1.2.1 specification. It contains 183 news articles and more than 61,000 non-punctuation tokens, among which 7,935 represent events.

We analyzed the corpus to investigate on the distribution of PoS (Part of Speech) for the tokens annotated as events. Most events are expressed in verbs and nouns. Sum of the two PoS types covers about 93% of all the event tokens, which is split into about 65% and 28% for verb and nouns, respectively.

The experiment is designed to see the effect of the selection method by using the feature candidates generated by the work of Jeong & Myaeng (2013), which showed the best performance in TimeML event recognition and classification in the literature. It generates feature sets based on the same criteria of the proposed method using syntactic dependencies and WordNet hypernyms. To find the concept (i.e., synset) of a target word, we applied the word sense disambiguation module of BabelNet (Ponzetto & Navigli, 2010). We also used Stanford Parser (Klein & Manning,

2003) to get the syntactic dependency based features.

A maximum entropy (ME) classifier was used because it showed the best performance for the tasks at hand, according to the literature. We also considered SVM, another popular machine learning algorithm in natural language processing. The evaluation was done by 5-fold cross validation, and the data of each fold was randomly selected. For the classifier, we used the Mallet machine learning package (McCallum, 2002) and Weka (Witten, Frank, & Hall, 2011).

5.2 Evaluation

We first evaluated the proposed tree-based feature selection in comparison with two widely accepted feature selection methods: information gain (IG) and χ^2. For each feature selection method, we chose the number of features that gave the best performance in F1. In Table 2, TSEL means the pure tree-based feature selection without the reselection process using χ^2 whereas TSEL+χ^2 means the proposed method followed by χ^2.

Compared to χ^2[3], TSEL dramatically reduced the feature space significantly by 73.93% and 54.42% for event recognition and type classification, respectively, but the decrease of effectiveness was insignificant for the both tasks. The decrease was compensated by the reselection process (hence the TSEL+χ^2 case) to the point of 1.26% improvement over the χ^2 case. For type classification, only 40.68% of the features required by χ^2 were enough to achieve the same level of effectiveness achieved χ^2. Due to the decrease of feature space, the running times of classification tasks (except preprocessing) were also quite reduced. The time-savings by TSEL were about 40% and 45% of χ^2 in the recognition and the type classification.

Event Recognition (ME)				
	IG	**χ^2**	**TSEL**	**TSEL +χ2**
# features	202,495	255,371	66,578 (-73.93%)	64,041 (-74.92%)
P	0.8878	0.8720	0.8664	0.8779
R	0.8413	0.8531	0.8571	0.8687
F1	0.8639	0.8624	0.8617 (-0.08%)	0.8733 (+1.26%)
T	4.12 s	4.14 s	2.52 s (-39.13%)	2.51 s (-39.37%)

Type Classification (ME)				
	IG	**χ^2**	**TSEL**	**TSEL +χ2**
# features	291,408	267,226	121,793 (-54.42%)	108,705 (-59.32%)
P	0.8050	0.8117	0.7847	0.8411
R	0.6340	0.6199	0.6334	0.6026
F1	0.7094	0.7029	0.7010 (-0.28%)	0.7021 (-0.11%)
T	9.73 s	9.69 s	5.31 s (-45.20%)	5.28 s (-45.51%)

P: Precision, **R:** Recall
T: Running Time of Classification (at PC with 3.0 GHz Core 2 Duo CPU and 8 GB memory)

Table 2. Comparisons in time and effectiveness for event recognition and type classification

Event Recognition (SVM)				
	IG	**χ^2**	**TSEL**	**TSEL +χ2**
# features	202,495	255,371	66,578 (-73.93%)	64,041 (-74.92%)
P	0.8277	0.8048	0.7338	0.8128
R	0.8406	0.8592	0.8806	0.8576
F1	0.8341	0.8311	0.8005	0.8346

Type Classification (SVM)				
	IG	**χ^2**	**TSEL**	**TSEL +χ2**
# features	291,408	267,226	121,793 (-54.42%)	108,705 (-59.32%)
P	0.6189	0.6179	0.6633	0.6833
R	0.6931	0.6531	0.6790	0.6700
F1	0.6539	0.6350	0.6711	0.6766

Table 3. Comparisons in effectiveness for event recognition and type classification using SVM classifier

Looking at the performance of different PoS types, we found that the performance of noun events was more meaningfully improved with a significantly reduced feature set. With the feature set reduction ratios of 81.66% and 81.50% for recognition and type classification, respectively, we achieved 6.85% and 3.94% of increase in F1[4]. For verbs, the numbers of features used for class recognition were also reduced significantly, but the F1 scores were slightly decreased. Our analysis shows that the increase in effectiveness for nouns is mainly attributed to the fact that the synsets of most nouns are located at a deep level of WordNet hierarchy. On the contrary, the hierarchy for verbs is not as deep as that of nouns. Note that the tree-based selection method is most helpful when heavy redundancy of features with a deep hierarchy causes a problem.

[3] We use χ^2 for discussion instead of IG because it showed the better performance than IG for the verb and noun event classification, which is the main focus of the research.

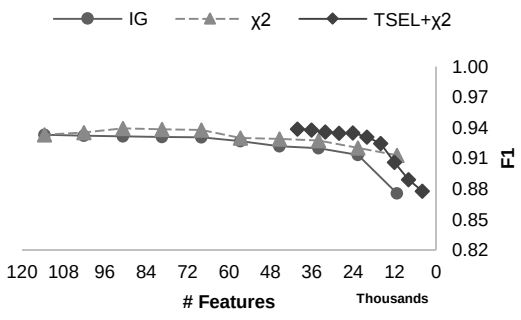
[4] The results are statistically significant with $p < 0.05$.

Recognition (ME)				
	Verb		Noun	
	# features	F1	# features	F1
χ^2	90,792	0.9393	123,480	0.7138
TSEL	40,189 (-55.74%)	0.9385 (-0.09%)	25,169 (-79.62%)	0.7273 (+1.89%)
TSEL + χ^2	40,180 (-55.74%)	0.9386 (-0.07%)	22,644 (-81.66%)	0.7627 (+6.85%)

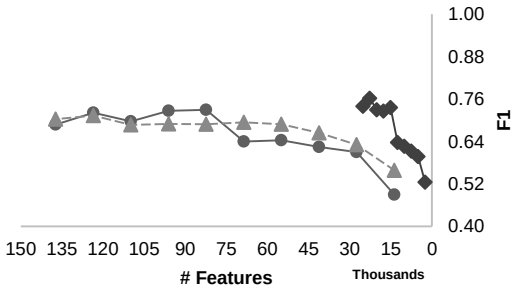
Classification (ME)				
	Verb		Noun	
	# features	F1	# features	F1
χ^2	217,287	0.7406	47,080	0.6288
TSEL	99,100 (-54.39%)	0.7220 (-2.51%)	21,722 (-53.86%)	0.6149 (-2.21%)
TSEL +χ^2	49,550 (-77.20%)	0.7223 (-2.47%)	8,708 (-81.50%)	0.6536 (+3.94%)

Table 4. Feature space sizes and effectiveness values for noun and verb events in event recognition and type classification

Figure 6 and 7 show the performance changes incurred by reducing the feature sets for different feature selection methods. The lines start from the point where all the selected features were used in each method and continue with a decrement of 10% of the feature set all the way to the minimum of 10% of the originally selected feature set. The starting points of TSEL+χ^2 indicate the results of pure TSEL. Despite the elimination of many features, the pure TSEL does not much harm the F1 compared to the best cases of IG and X2. It clearly shows that reducing the size of feature sets is less detrimental with the proposed method in almost all the cases than the other selection methods. TSEL also shows the possibility to select valuable features without manual check of performance for the feature space size. For event type classification, the manual selection process (TSEL+χ^2) is still needed in order to find the best features but it guarantees the more effectiveness.

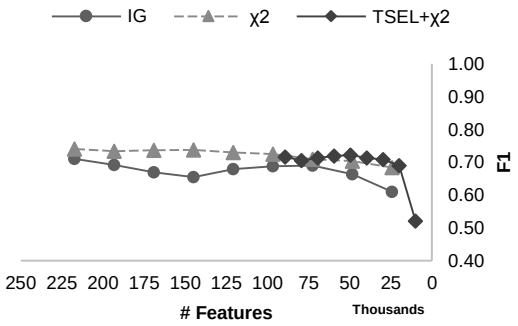


(a) Verb Event Recognition

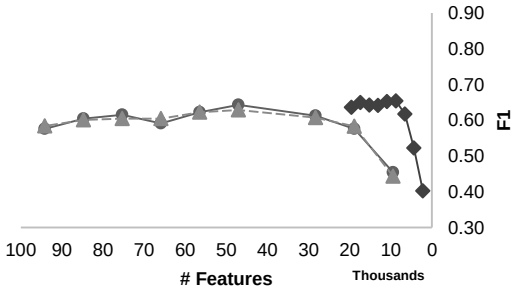


(b) Noun Event Recognition

Figure 6. Performance change with feature set reduction in event recognition in each of the feature selection methods



(a) Verb Event Type Classification



(b) Noun Event Type Classification

Figure 7. Performance change with feature set reduction in event type classification in each of the feature selection methods

For the type classification task, Table 5 shows detailed scores for all the event types separately. An improvement is observed for most of the event types except for OCCURRENCE. Our analysis shows that this is related to the size of the training data. Since the ratio of OCCURRENCE events is about 53% of all the events in the TimeBank corpus, the training data for the OCCURRENCE type is much bigger than the others. It indicates that the feature redundancy is problematic when the training data is relatively small and that careful selection of features is particularly important to avoid overfitting.

	χ^2	TSEL+χ^2
REPORTING	0.9111	0.9201 (+0.99%)
PERCEPTION	0.6186	0.6292 (+1.71%)
ASPECTUAL	0.6444	0.6771 (+5.07%)
I_ACTION	0.6173	0.6346 (+2.80%)
I_STATE	0.6251	0.6866 (+9.84%)*
OCCURRENCE	0.7219	0.6980 (-3.31%)*
STATE	0.5246	0.5534 (+5.49%)*

Table 5. Performance for different event types (unit: F1). * indicates that the percent increase or decrease is statistically significant with $p < 0.05$.

6 Related Work

EVITA (Saurí et al., 2005) is the first event recognition tool for TimeML specification. It recognizes events by using both linguistic and statistical techniques. It uses manually encoded rules based on linguistic information as main features to recognize events. It also uses WordNet classes to those rules for nominal event recognition, and checks whether the head word of noun phrase is included in the WordNet event classes. For sense disambiguation of nouns, it utilizes a Bayesian classifier trained on the SemCor corpus.

Boguraev & Ando (2007) analyzed the TimeBank corpus and presented a machine-learning based approach for automatic TimeML events annotation. They set out the task as a classification problem, and used a robust risk minimization (RRM) classifier to solve it. They used lexical and morphological attributes and syntactic chunk types in bi- and tri-gram windows as features.

Bethard & Martin (2006) developed a system, STEP, for TimeML event recognition and type classification. They adopted syntactic and semantic features, and formulated the event recognition task as classification in the word-chunking paradigm. They used a rich set of features: textual, morphological, syntactic dependency and some selected WordNet classes. They implemented a Support Vector Machine (SVM) model based on those features.

Llorens et al. (2010) presented an evaluation on event recognition and type classification. They added semantic roles to features, and built the Conditional Random Field (CRF) model to recognize events. They conducted experiments about the contribution of semantic roles and CRF and reported that the CRF model improved the performance but the effects of semantic role features were not significant.

Jeong & Myaeng (2013) argued and demonstrated that unit feature dependency information and deep-level WordNet hypernyms are useful for event recognition and type classification. Their proposed method utilizes various features including lexical se-mantic and dependency-based combined features. In the TimeBank 1.2 corpus, the approach achieved 0.8601 and 0.7058 in F1 in event recognition and type classification, respectively.

7 Conclusion

In this paper, we proposed a novel feature selection method for event recognition and event type classification, which utilizes a semantic hierarchy of features. While our current work is based on the WordNet hierarchy and syntactic dpendencies, the proposed method can be applied as long as it is possible to utilize a feature hierarchy, and shows the possibility to select valuable features without manual check of performance for the feature space size.

Our experimental results show that the proposed method is significantly effective in reducing the feature space compared to the well-known feature selection methods, and yet the overall effectiveness is similar to or sometimes better than a state-of-the-art approach depending on the PoS of the events. In particular, the effectiveness for noun events was improved quite meaningfully when the feature space was reduced significantly.

Although the proposed method showed the encouraging results, it still has some limitations. One issue is on the depth of the features in hierarchy. For verb, most features are located at shallow levels so the feature space reduction ratio is lower than those of noun. It implies that we need other approaches for verbs. Another one is on the recall. The proposed method showed high precision but relative lower recall. We conjecture that one reason is the lack of lexical information due to small size of TimeBank corpus.

Not only to improve recall but also for extensibility of the proposed method, we need to utilize other larger-scale resources for this tasks and even apply the proposed method for other types of text classification.

Acknowledgments

This work was supported by a Microsoft Research Asia (MSRA) Faculty-Specific Project and by the research project of Korean Agency for Defense Development (ADD) [UD120064ED, Research on Extracting Contextual Factors and their Relations from Natural Language Factors].

Reference

Bethard, S., & Martin, J. H. (2006). Identification of event mentions and their semantic class. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing* (pp. 146–154). Association for Computational Linguistics.

Boguraev, B., & Ando, R. (2007). Effective Use of TimeBank for TimeML Analysis. In F. Schilder, G. Katz, & J. Pustejovsky (Eds.), *Annotating, Extracting and Reasoning about Time and Events* (Vol. 4795, pp. 41–58). Springer Berlin Heidelberg.

Daniel, N., Radev, D., & Allison, T. (2003). Sub-event based multi-document summarization. In *Proceedings of the HLT-NAACL 03 on Text summarization workshop* (Vol. 5, pp. 9–16). Association for Computational Linguistics.

Jeong, Y., & Myaeng, S.-H. (2013). Using WordNet Hypernyms and Dependency Features for Phrasal-level Event Recognition and Type Classification. In P. Serdyukov, P. Braslavski, S. Kuznetsov, J. Kamps, S. Rüger, E. Agichtein, … E. Yilmaz (Eds.), *Advances in Information Retrieval* (Vol. 7814, pp. 267–278). Springer Berlin Heidelberg.

Klein, D., & Manning, C. D. (2003). Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics* (pp. 423–430). Association for Computational Linguistics.

Llorens, H., Saquete, E., & Navarro-Colorado, B. (2010). TimeML events recognition and classification: learning CRF models with semantic roles. In *Proceedings of the 23rd International Conference on Computational Linguistics* (pp. 725–733). Association for Computational Linguistics.

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

McCallum, A. K. (2002). MALLET: A Machine Learning for Language Toolkit.

Ponzetto, S. P., & Navigli, R. (2010). Knowledge-rich Word Sense Disambiguation rivaling supervised systems. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics* (pp. 1522–1531). Association for Computational Linguistics.

Pustejovsky, J. (2002). TERQAS: Time and Event Recognition for Question Answering Systems. In *Proceedings of ARDA Workshop*.

Pustejovsky, J., Castaño, J., Ingria, R., Saurí, R., Gaizauskas, R., Setzer, A., & Katz, G. (2003). TimeML: Robust Specification of Event and Temporal Expressions in Text. In *Proceedings of the 5th International Workshop on Computational Semantics*.

Pustejovsky, J., Hanks, P., Saurí, R., See, A., Gaizauskas, R., Setzer, A., … Lazo, M. (2003). The TIMEBANK Corpus. In *Proceedings of the Corpus Linguistics 2003 conference* (pp. 647–656).

Pustejovsky, J., Knippen, R., Littman, J., & Saurí, R. (2007). Temporal and Event Information in Natural Language Text. In H. Bunt, R. Muskens, L. Matthewson, Y. Sharvit, & T. E. Zimmerman (Eds.), *Computing Meaning* (Vol. 83, pp. 301–346). Springer Netherlands.

Saurí, R., Knippen, R., Verhagen, M., & Pustejovsky, J. (2005). Evita: a robust event recognizer for QA systems. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing* (pp. 700–707). Association for Computational Linguistics.

Tufféry, S. (2011). *Data Mining and Statistics for Decision Making* (2nd ed.). John Wiley & Sons.

Witten, I. H., Frank, E., & Hall, M. A. (2011). *Data Mining: Practical Machine Learning Tools and Techniques* (3rd ed.). San Francisco, CA, USA: Morgan Kaufmann.