Free Model of Sentence Classifier for Automatic Extraction of Topic Sentences

M.L. Khodra[1], D.H. Widyantoro[1], E.A. Aziz[2] & B.R. Trilaksono[1]

[1]School of Electrical Engineering and Informatics,
Bandung Institute of Technology, Indonesia
[2]Faculty of Language and Arts Education,
Indonesia University of Education, Indonesia
Email: masayu@informatika.org

Abstract. This research employs free model that uses only sentential features without paragraph context to extract topic sentences of a paragraph. For finding optimal combination of features, corpus-based classification is used for constructing a sentence classifier as the model. The sentence classifier is trained by using Support Vector Machine (SVM). The experiment shows that position and meta-discourse features are more important than syntactic features to extract topic sentence, and the best performer (80.68%) is SVM classifier with all features.

Keywords: *automatic extraction; sentence classifier; SVM; topic sentence.*

1 Introduction

In recent years, the number of scientific articles increases explosively. Jinha [1] estimates this number to be 50 millions. Since researchers use these articles as the primary source of current leading-edge information in their fields, keeping updated on all relevant papers has become a problem. Therefore, multi-paper summarization is proposed to help researchers handle this problem.

Summarization transforms reductively source text to summary by selection and/or generalisation on important content in the source [2]. In multi-paper summarization, the source text is a collection of papers. Each paper is represented as Rhetorical Document Profile (RDP) [3] that consists of rhetorical slots. RDP slot fillers are important sentences in the paper so that RDP can be used as summary of a paper.

Since topic sentences give essential contents of a document and 99% of 1018 paragraphs in scientific papers have topic sentences [4], this research explores a sentence classifier to extract topic sentences of paragraphs that will be selected as RDP slot fillers. Topic sentences increase reading comprehension and help readers to better understand a document.

Received December 13th, 2010, Revised March 8th, 2011, Accepted for publication March 10th, 2011.

A computational model for identifying topic sentences has been developed by conducting discriminant analysis [5]. The contribution of the present paper over existing research in this area (particularly [5]) is to study application of Support Vector Machines classifier for identifying topic sentences of paragraphs of scientific articles. Three types of features, which are position, syntactic features, and meta-discourse features, are combined to be investigated.

The rest of the paper is organized as follows. Section 2 provides an overview of prior works related to topic sentence. The free model for automatic topic sentences extraction system is then described in section 3. Experiment results are discussed in Section 4, followed by some concluding remarks on Section 5.

2 Related Works

Position of topic sentences in a paragraph has been studied and revisited by many researchers. Kaplan [6] defined that topic sentence appears both at the beginning and at the end of paragraphs. Baxendale [7] investigated 200 paragraphs, and found that topic sentence appears at the beginning in 85% of paragraphs and at the final in 7% of paragraphs. Smith [8] revisited Braddock's previous research and concluded that the percentage of expository paragraph that begins with a topic sentence is 66%, not only 13% as at Braddock's conclusion. Our previous research also found that positions of topic sentences are 78.54% at the beginning of paragraphs, 11.27% at the end, and the rest in other positions [4].

There are two computational models for identifying topic sentences: derived model that uses paragraph context, and free model that uses only sentential features [5]. Since human identifies topic sentence by evaluating sentence relationship in a paragraph, it is clear that derived model is more promising than the free model. On the contrary, McCarthy, et al. [5] compared the two models and found that free model is more promising than the derived model. By conducting discriminant analysis for free model, the accuracy ranged from 73% to 78% were comparable with those of expert raters recorded 78% [5].

A topic sentence can also be identified using discourse analysis of paragraph. For automatic discourse analysis of paragraphs, Theijssen [9] used Rhetorical Structure Theory and explored twenty features of five different feature types to predict the presence of rhetorical relations between sentences within paragraphs. Unfortunately, the performance was disappointing, and the experiment was conducted using only a small dataset [9].

Toward to RDP generation, Teufel [3] employed metadiscourse features in her experiments. Hyland [10] defined that metadiscourse is more generally seen as

the author's linguistic and rhetorical manifestation in the text in order to bracket the discourse organization and the expressive implications of what is being said. These features were used to capture profile of document structure "who-does-what" with some syntactic variations. For this purpose, Teufel [3] used three metadiscourse features: formulaic expression, type of agent, and type of action.

Kupiec, et al. [11] proposed extract selection as classification problem, and used probability classifier to select some important sentences as the extracts. In sentence classification, a corpus is analyzed to construct a model, which is known as a sentence classifier, to predict the class of sentence whose class label is unknown. Since each sentence is represented as a feature vector, this corpus-based approach offers a direct method for finding an optimal combination of extraction features. Teufel [12] replicated Kupiec's experiment with different data, and showed the usefulness of the methodology for different types of data and evaluation strategies. In identifying topic sentence, McCharty, et al. [5] also used classification approach by conducting discriminant analysis with sentence type as predefined classes.

3 Sentence Classifier as Free Model

Since free model is more promising than derived model [5], this research employs free model that uses only sentential features without paragraph context. For finding optimal combination of features in free model, corpus-based classification is also used in constructing the model for extracting the topic sentence of a paragraph.

Kupiec, et al. [11] used corpus-based classification to select important sentences for document summarization. The classifier computed probabilities for each sentence based on its features. All sentence probabilities were ranked, and n-top sentences were selected and arranged as a document summary. Since free model does not use paragraph context, topic sentence selection in corpus-based classification is without ranking process. The assignment of a sentence as the topic sentence is performed by comparing its probabilities for each label.

The development of a corpus-based classification system requires two phases: model development, and extraction of topic sentences using the model. Figure 1 shows that model development returns the classifier that will be used to extract topic sentences of a paragraph.

In the model development phase, the paragraph corpus is preprocessed into a collection of sentences, in which each sentence is represented as a parse-tree. For each parse-tree, all syntactic and metadiscourse features are extracted and

represented as a feature vector. The resulting feature vectors are used as training set for the learning process to get a classifier model.

In the extraction phase, unseen paragraph is preprocessed using the same process that is used in the development phase to get feature vectors. For each feature vector, the system calculates the probability of topic sentence assignment.

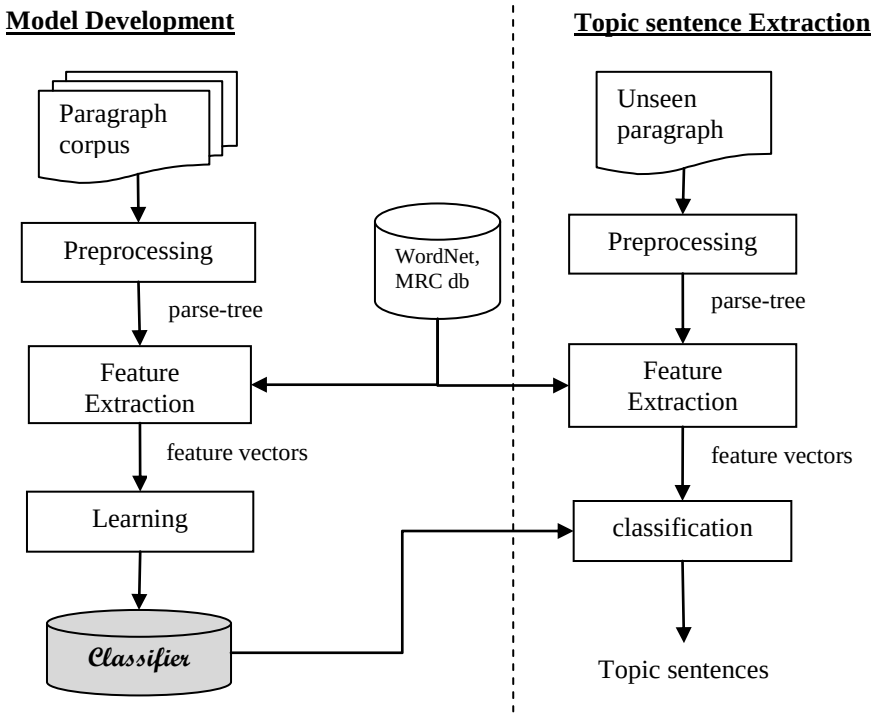

Figure 1 Corpus-based classification system in two phases: model development, topic sentence extraction.

Each next sub-section provides description for each system component that is illustrated in Figure 1: paragraph corpus, preprocessing, feature extraction, learning, and classification.

3.1 Paragraph Corpus

A paragraph corpus is a collection of paragraphs equipped with information about the topic sentences of each paragraph. This corpus developed in previous research were taken from ACL Anthology Reference Corpus (ACL-ARC) [13],

a corpus of scholarly publications about Computational Linguistics [14]. The topic sentences of each paragraph are assigned in the annotation process by manual selection [4].

The resulting corpus consists of 1018 paragraphs of 4349 sentences, of which 1108 are topic sentences. Figure 2 shows an example of paragraphs in XML format. Each paragraph corresponds to a list of topic sentences that may be empty. The number of topic sentences in a paragraph is in the range of zero to three sentences. However, 89.69% of the paragraphs have only one topic sentence and 0.98% paragraphs have no topic sentences.

```
<corpus>
- <topic_paragraph>
    <paragraph file="A00-2011_edited.pdf.raw.xml" id_paragraph="4" section="1. Introduction">Many corpus-based MT systems
    require parallel corpora (Brown et_al, 1990; Brown et_al, 1991; Gale and Church, 1991; Resnik, 1999). Kikui (1999) used a
    word sense disambiguation algorithm and a non-parallel bilingual corpus to resolve translation ambiguity.</paragraph>
  - <topic_sentences num="1">
      <topic_sentence>2</topic_sentence>
    </topic_sentences>
  </topic_paragraph>
- <topic_paragraph>
    <paragraph file="A00-2011_edited.pdf.raw.xml" id_paragraph="55" section="5. Group Similarity">Fortunately, state of the art
    graph partitioning algorithms can approximate these bisections in polynomial time (Goehring and Saad, 1994; Karypis and
    Kumar, 1999; Kernighan and Lin, 1970). We used the same approximation methods as in (Karypis et_al, 1999).</paragraph>
  - <topic_sentences num="1">
      <topic_sentence>2</topic_sentence>
    </topic_sentences>
  </topic_paragraph>
- <topic_paragraph>
    <paragraph file="A00-2011_edited.pdf.raw.xml" id_paragraph="60" section="6. Experimental Results">In Section 3, we
    presented an algorithm for identifying the contextually similar words of a word in a context using a corpus-based thesaurus
    and a collocation database. Each of the six nouns has similar words in the corpus-based thesaurus.</paragraph>
  - <topic_sentences num="1">
      <topic_sentence>1</topic_sentence>
    </topic_sentences>
  </topic_paragraph>
```

Figure 2 Example of paragraphs and their corresponding topic sentences.

3.2 Preprocessing

The preprocessing component accepts a paragraph corpus, and returns the corresponding collection of parse-trees. There are two processes in preprocessing: splitting each paragraph to a sentence collection, and parsing each sentence to get a parse-tree.

For the splitting of sentences in a paragraph, it is assumed that each sentence is delimited by period (“.”). However, there are some exceptions to this rule, including the point in decimal numbers, and acronyms (e.g., U.S.A, et al., i.e.). To account for the use of point (“.”) in decimal numbers, space is added to the delimiter. Acronyms are handled by replacing all periods in acronym with underline, for example, i.e. is replaced by i_e.

A parse-tree or syntax tree represents syntactic structure of a text. In this research, Stanford Parser [15] is used to parse sentences, with the output is a parse-tree of the corresponding sentences. This parse-tree used Penn Treebank tag set [16]. Figure 3 shows the parse-tree generated from an example sentence "*There were totally 202 hypotheses generated*.".

```
(ROOT
  (S [62.698]
    (NP [4.519] (EX [0.433] there))
    (VP [52.009] (VBD [1.098] were)
      (VP [47.348]
        (ADVP [35.344] (RB [7.717] totally)
          (NP [23.705] (CD [10.798] 202) (NNS [11.443] hypotheses)))
        (VBN [6.948] generated)))))
```

Figure 3 Parse-tree of sentence "*There were totally 202 hypotheses generated*.";

EX: existential there; VBD: verb in past tense; RB: adverb; CD: cardinal number; NNS: noun in plural; VBN: past participle.

3.3 Feature Extraction

This process extracts all potential features. First, position is the most important feature to identify topic sentence in a paragraph because 78.54% of topic sentences appear at the beginning of paragraphs [4]. Second, all of McCarthy et al.'s syntactic features [5] are also considered here except the domain and expository features because the system is limited to process only texts from the domain of computational linguistics, and all paragraphs are assumed as expository paragraphs. Last, Teufel's metadiscourse features [3] are used as structure markers based on syntactic pattern.

Table 1 shows the list of features to be extracted and the extraction process of each. All features can be grouped in four categories except the three first features. So, there are seven different processes for feature extraction as follows.

The first is calculating the length of a sentence, which is number of words in the sentence. For example, "There were totally 202 hypotheses generated." is a sentence with six words. Each word in the sentence is underlined. Thus, length of the sentence is six.

The second process is classifying position values based on the position of sentence and the number of sentences in paragraph. This process is used to determine the value of position feature. If a sentence ***s*** has position ***id*** in a paragraph that has ***N*** sentences, position value for sentence ***s*** is determined by formula 1.

$$position(s) = \begin{cases} 1 & , id = 1 \\ 0.5 & , id = N \\ 0 & , id \notin [1, N] \end{cases} \quad (1)$$

Table 1 Feature pool and global description how to be extracted.

Feature	Extraction
Position sentence length number of words before a main verb	Extracting directly based on its definition
Word Incidence Features adjective incidence existential there incidence incidence of 3rd person singular grammatical form anaphora incidence coordinators incidence cardinal number incidence incidence of past tense endings	Counting the incidence of tag in a sentence
Word Taxonomy Features Hypernymy Polysemy	Explored word taxonomy for all words in a sentence
Psycholinguistics Feature concreteness index	Get concreteness index of each words
Metadiscourse Features formulaic type agent type action type	Matching all syntax pattern defined by Teufel [3]

The next process is counting the incidence of a part of speech (POS) tags. This process is used to extract values of seven features: incidence of adjective, existential there, anaphora, coordinators, cardinal number, past tense endings, and 3rd person singular grammatical form. Formula 2 shows general formula for calculating the incidence feature value. Table 2 shows the corresponding POS tags for the incidence features extracted. The term "anaphora incidence" is used here for pronoun incidence. The term "coordinators incidence" is used for conjunction, connectives, or clarification incidence.

$$incidence(tag, s) = \frac{1000 * frequency(tag, s)}{length(s)} \quad (2)$$

The fourth process is counting the number of words before a main verb. The purpose of this process is to extract the number of words before a main verb. For the sentence in Figure 3, *were* is the main verb, and the number of words preceding the main verb is one.

Table 2 POS tags in accordance with the features extracted.

Feature	Penn Treebank Tags
Adjective incidence	JJ, JJR, JJS
existential there incidence	EX
incidence of 3rd person singular grammatical form	VBZ
anaphora incidence	PRP, PRP$, WP, WP$, WDT, WH
Coordinator incidence	CC + connective phrases [17]
cardinal number incidence	CD
incidence of past tense endings	VBD

The fifth is accessing word taxonomy to calculate average of each word level and synonym sets. This process is used to extract values of hypernymy and polysemy. WordNet [18] is used as word taxonomy, and JWI [19] is used as the library for accessing the WordNet. Only words recognized by WordNet (adjective/JJ*, verb/VB*, MD, noun/N*, adverb/RB*) have hypernym and polysemy values. If a sentence ***s*** has ***nb_wn***, which is number of words in sentence ***s*** that are recognized by WordNet, the hypernymy and polysemy values of sentence ***s***, are calculated using formula 3 and 4.

$$hypernymy(s) = \frac{\sum_{1}^{nb\_wn} hypernymy(w_i)}{nb\_wn} \tag{3}$$

$$polysemy(s) = \frac{\sum_{1}^{nb\_wn} polysemy(w_i)}{nb\_wn} \tag{4}$$

The next process is accessing psycholinguistics database to calculate the average of word concreteness. This process is used to extract value of concreteness index. The MRC psycholinguistics database [20] is used here as the data source, and jMRC [21] is used as the library for accessing this database. Since there are different tags between Penn Treebank tags and MRC tags, tag mapping as shown by Table 3 was needed. If a sentence ***s*** has ***nb_mrc***, which is number of words in sentence ***s*** that are recognized by MRC database, the concreteness index of sentence ***s*** is calculated using formula 5.

$$concreteness(s) = \frac{\sum_{1}^{nb\_mrc} concreteness(w_i)}{nb\_mrc} \quad (5)$$

Table 3 Tag mapping between Penn Treebank tags and MRC tags.

Penn Treebank Tags	MRC Tags
NN, NNS, NP, NPS	Noun
JJ, JJR, JJS	Adjective
RB, RBR, RBS, EX	Adverb
MD, VB, VBD, VBG, VBP, VBZ	Verb
VBN	past participle
IN	Preposition
CC	Conjunction
PRP,PRP$, WDT, WP, WP$, WRB	Pronoun
UH	Interjection
CD, DT, FW, LS, PDT, POS, RP, SYM, TO	Other

The last process is matching parse-tree with syntax pattern of metadiscourses to identify types of formulaic expression, agent, and action found in a sentence. Figure 4 shows the sentence "*While it may be worthwhile to base such a model on preexisting sense classes (Resnik, 1992), in the work described here we look at how to derive the classes directly from distributional data*." matches five types of formulaic expression, four action types, and two agent types. For example, this sentence matches with pattern "^ *look at how*" so that the system extracted action type *interest* shown by v3. Character ^ in a pattern means the word look is a trigger word for the pattern. A pattern will be considered if the trigger word is found in the sentence.

While[f1] it may be worthwhile to base[v1] such a model on preexisting sense classes (Resnik, 1992), in the[a1] work[f2] described[v2] here[f3] we[a2] look at how[v3] to derive[v4] the classes directly from distributional data.		
Types of formulaic expression: f1: comparison_formulaic f2: method_formulaic f3: here_formulaic	Actions types: v1: continue v2: presentation v3: interest v4: change	Agent types: a1: ref_agent a2: us_agent

Figure 4 An example of metadiscourse features of a sentence; text in the top row is the example sentence, and the three columns below it contain the corresponding metadiscourse features of the sentence.

For each sentence, feature extraction returns a feature vector that consists of the value of each feature. For example, sentence, "*All the editing and visualizing operations are performed through this window (see Figure 3).*" that is the first sentence of a paragraph, has a feature vector consists of 58 values as shown in Figure 5. Each feature corresponding with metadiscourse is represented by boolean: 0 for false, and 1 for true.

1. Position:1.0
2. sentence length:14
3. number of words before a main verb: 6
4. adjective incidence: 0
5. existential there incidence: 0.0
6. incidence of 3rd person singular grammatical form: 0.0
7. anaphora incidence: 0.0
8. coordinators incidence: 142.857
9. cardinal number incidence:71.428
10. incidence of past tense endings:0.0
11. Hypernymy: 4.231
12. Polysemy: 8
13. concreteness index:328.25
14. affect_formulaic:0
15. bad_formulaic:0
16. comparison_formulaic:0
17. continue_formulaic:0
18. contrast_formulaic:0
19. detail_formulaic:0
20. future_formulaic:0
21. gap_formulaic:0
22. good_formulaic:0
23. here_formulaic:0
24. in_order_to_formulaic:0
25. method_formulaic:0
26. no_textstructure_formulaic: 1
27. similarity_formulaic:0
28. them_formulaic:0
29. textstructure_formulaic:0
30. tradition_formulaic:0
31. us_previous_formulaic:0
32. affect:0
33. argumentation:0
34. better_solution:0
35. change:0
36. comparison:0
37. continue:0
38. contrast:0
39. interest:0
40. need:0
41. presentation:0
42. problem:0
43. research:0
44. solution: 1
45. textstructure:0
46. use:0
47. copula: 1
48. aim_ref_agent:0
49. gap_agent:0
50. general_agent:0
51. problem_agent:0
52. ref_agent:0
53. ref_us_agent:0
54. solution_agent:0
55. textstructure_agent:0
56. them_agent:0
57. them_pronoun_agent:0
58. us_agent:0

Figure 5 Example 58 values constructed a feature vector for a sentence. Features 1-13 are as previously described in this section, features 14-31 represent formulaic types, features 32-47 represent action types, and features 48-58 represent agent types.

3.4 Learning

This component accepts feature vectors as its input, and produces a classifier as its output. Using supervised learning, the training data are pairs of <data, label> whose patterns are analyzed so that new data can be accurately classified. The resulting pattern set is called classification model or classifier.

For the topic sentence problem, we have paragraph corpus to be prepared as training data in the form <sentence, label>. Each sentence is represented as a feature vector. As shown in Figure 2, each paragraph has topic sentences information. A sentence is labeled yes if the sentence is a member of topic sentences of its paragraph, and no if otherwise.

Before training the final classifier, feature selection is performed to identify the relevant features in the training set. Only a subset of relevant features returned by the feature selection process is used as the input for the learning algorithm. Kohavi and John [22] classified feature selection approach into two categories: wrapper and filter. They concluded that wrapper approach naturally fits to find the optimal features, which depend on the specific biases and heuristics of the learning algorithm. Therefore, backward elimination, one technique of the wrapper approach, is used in this research. It treats an induction algorithm as a black box that is used to evaluate each candidate feature subset [23].

```
Input: training Data is collection of feature vectors;
       featureset: set of features to be optimized
Output: optimized feature set
//initial state: featureSet ≠ {}, trainingData is defined
//final state: featureSet is optimum subset of featureSet
max←-9999
Repeat
   //performance:training performance
   performance ← training(trainingData, featureSet)
   fsElim ← null
   for (f ∈ featureSet)
       performanceElim← training(trainingData, featureSet-f)
       if performanceElim≥max then
           fsElim ←f
   if max>performance then
       featureSet←featureSet-fsElim
   else fsElim←null
until (fsElim=null)  //no more feature eliminated
return featureSet
```

Figure 6 Pseudo code Backward Elimination.

Backward elimination begins with the full set of features and iteratively deletes the features whose elimination gives better accuracy than before [22]. Figure 6

shows pseudo code of the backward elimination procedure implemented in this research. We assumed that procedure training returns training performance.

3.5 Classification

The classifier is used for selecting the topic sentences of a paragraph. Prior to this selection, the paragraph is preprocessed and feature vectors of its sentences are extracted. Then classifier computes the probabilities each label based on the values of these feature vector. Topic sentences are all sentences that have higher probabilities as topic sentences.

A sentence classifier is used for analyzing each sentence and estimating the sentence probability for each label. Given sentential features $\mathbf{F}$ of sentence s, the sentence classifier will estimate posterior probability $P(\ell_i|\mathbf{F})$ for each label $\ell_i \in L$, and assign label with maximum probability as follows. So, the sentence classifier will assign topic sentence to sentence s with sentential features $\mathbf{F}$ if and only if the $P(yes|\mathbf{F}) > P(no|\mathbf{F})$. If the label is yes, the sentence is topic sentence.

A paragraph may have no or more than one topic sentence. By using free model, the system described here is able to handle such cases. Because of no paragraph context, each sentence is classified independently with other sentences. It is possible that the classifier assigns all sentences as topic sentence, and vice versa.

4 Experiment

The objective of experiments is to develop a sentence classifier that could extract topic sentences of a paragraph. These experiments also investigate influence of feature sets, and influence of wrapper as feature selection.

4.1 Data

To achieve the goal, this experiments use a corpus constructed in our previous research [4]. The paragraph corpus used is the one constructed in our previous research. This corpus is split randomly into a training set and a test set. The training set consists of two-thirds of the total number of paragraphs in the corpus, and the remaining paragraphs are used as test set. Table 4 shows detail description of training set and test set.

Table 4 Description of training-set and test set.

Description	Training set	Test set	Total
Number of paragraphs	681	337	1018
Number of sentences	2951	1398	4349
Number of topic sentences	742	363	1108
Number of paragraphs with 0 topic sentences	7	3	10
Number of paragraphs with 1 topic sentences	608	306	913
Number of paragraphs with 2 topic sentences	62	27	90
Number of paragraphs with 3 topic sentences	4	1	5
Number of topic sentences at the beginning of paragraph	588	281	870
Number of topic sentences at the end of paragraph	74	32	72
Number of topic sentences at the other position in paragraph	82	50	166

4.2 Method

This research uses Support Vector Machines (SVM) as learning algorithm. Training data are mapped into a higher dimensional space by kernel function, and then SVM finds the hyperplane with maximal margin between classes in higher dimensional space [24].

RBF (Radial Basis Function), which is the default kernel in LibSVM [25], is used. When using RBF kernels, there are two parameters whose values need to be set: C and γ. Besides that, parameter of margin weight is also optimized because of unbalanced training set. This research uses complete grid searches for finding the best parameters values that optimizes classifier performance.

Since SVM classifier is originally not a probabilistic classifier [25-27], probabilistic option is set. Before the learning algorithm is applied, the value of each attribute is scaled to fit the range [0,1]. Such scaling is recommended to avoid attributes in greater numeric ranges dominate those in smaller numeric ranges and numerical difficulties during the calculation [28].

Performance measures used to evaluate the SVM classifier are F-measure and accuracy. F-measure is expressed by formula 6 [24], and accuracy is expressed by formula 7.

$$F - measure = \frac{2 * TP}{2 * TP + FN + FP} \tag{6}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} \quad (7)$$

where TP is true positive (classifier judgment=label=yes), true negative (classifier judgment=label=no), FN is false negative (classifier judgment=no, label=yes), and FP is false positive (classifier judgment=yes, label=no).

4.3 Results and Discussion

The first experiment investigates influence of meta-discourse features. There are three variations of feature sets: all features (position + syntactic features + meta-discourse features), feature set without meta-discourse features (position + syntactic features), and feature set without syntactic features. Position feature is always used in each variation because this feature is key feature in identifying topic sentences [5].

The best performer is SVM classifier with all features, and the worst performer is SVM classifier without meta-discourse features. Based on this result, meta-discourse features are important features in extracting topic sentences because elimination of these features decreases the classifier performance. Besides that, performances of SVM classifiers except the second one are better than the baseline that extracts initial sentence as topic sentence. Although the improvement of the best performer to baseline is only 0.42% (F-measure), its accuracy (90.20%) is better than McCarthy's accuracy that ranged 73% to 78%.

Table 5 Influence of feature set to SVM classifier performances on testing.

Feature set	F-measure	Accuracy
All features	**80.62%**	90.20%
Feature set without meta-discourse features	0%	74.03%
Feature set without syntactic features	80.51%	90.13%
Baseline: 1st sentence	80.29%	90.13%

Table 5 shows that in extracting topic sentences, F-measure is better than accuracy as performance measure because F-measure is more sensitive to variations in the number of correct decision than accuracy [29]. Although the accuracy is still high (74.03%) for the second classifier, its F-measure is zero, meaning that no topic sentence can be identified. On the other hand, the third and fourth classifiers have the same accuracies, but have different F-measure. Both classifiers have the same correct prediction (total number of true positive and true negative), and different number of true positive. The next discussion only uses F-measure as performance measure.

The second experiment investigates influence of feature selection that applies backward elimination. When the final classifiers are trained by using those optimum subsets of relevant features, the classifier performance for identifying topic sentences in test set is shown at the first and second rows of Table 6. It shows that optimization by feature selection provides over-fitting problem that gives better training performance, but also gives bad testing performance.

Table 6 Influence of feature selection to SVM classifier performance.

Feature Selection – Classifier Constructs	Training F-measure	Testing F-measure
Without feature selection - conventional	73.68%	80.62%
With feature selection – conventional	84.42%	71.24%

Since position is key feature in identifying topic sentence, the model can be a set of three position-specific classifier. If the training set is divided three based on sentence position and each one used to train the classifier, Table 7 shows set of classifiers has different performance than one conventional classifier. Adding feature selection decrease the total training performance, but increase the testing performance. Besides that, the training requires more computational resources.

Table 7 Influence of feature selection to SVM classifier-set performance.

Feature Selection – Classifier Constructs	Training F-measure				Testing F-measure
	Initial	Ending	Other	Total	
Without feature selection – 3 classifiers	92.67%	88.11%	70.18%	86.63%	80.29%
With feature selection – 3 classifiers	92.67%	64.41%	36.96%	83.95%	**80.68%**

5 Conclusions and Future Research

In summary, we have developed SVM sentence classifier for extracting topic sentence by considering variation in feature set and usage of backward elimination. The best performer classifier used all features (position + syntactic features + meta-discourse features) without feature selection. We also found that position and meta-discourse features are more important than syntactic features.

Feature selection provides over-fitting problem for conventional classifier, but also improves the performance for position-specific classifier set. This process is potential for improvement and will be investigated in the future.

Another effort to improve this baseline system is using heuristics. Simple heuristics to identify all paragraphs with no topic sentence can be applied. This approach uses paragraph context. For example, all sentences in a paragraph are checked to have textstructure_formulaic. If all sentences have true values for textstructure type of formulaic expression, the paragraph has no topic sentences. This heuristics improve the performance by decrease false positive.

In the future, our research will explore how to improve performance of classifier for extracting topic sentences. On the other hand, this subsystem will be integrated to paper-summarization system based on RDP.

Acknowledgements

This research work has been supported in partial by the *Hibah Penelitian Disertasi Doktor* of *Institut Teknologi Bandung*, which is sponsored by the Directorate General of Higher Education, the Ministry of National Education of the Republic Indonesia. Thanks to Yudi Wibisono, Renny Pradina Kusumawardani, and Nur Ulfa Maulidevi for the initial review.

References

[1] Jinha, A.E., *Article 50 Million: An Estimate of the Number of Scholarly Articles in Existence*, 2010.

[2] Jones, K.S., *Automatic summarising: The state of the art.* Information Processing and Management, **43**, pp. 1449-1481, 2007.

[3] Teufel, S. *Argumentative Zoning: Information Extraction from Scientific Text.* PhD Dissertation, University of Edinburgh, 1999.

[4] Khodra, M.L., Widyantoro, D.H., Aziz, E.A., Trilaksono, B.R., *Konstruksi Koleksi Utama Paragraf*, in Proc. Konferensi Nasional Informatika, 2010.

[5] McCarthy, P.M., et al., *Identifying Topic Sentencehood, Behavior Research Methods*, http://brm.psychonomic-journals.org/ , 2008.

[6] Kaplan, R., *Cultural Thought Patterns in Inter-Cultural Education.* Landmark Essay on ESL Writing, 1966.

[7] Baxendale, P.B., *Machine-made index for technical literature—an experiment.* IBM Journal of Research and Development, 1958.

[8] Smith, C.G., *Braddock Revisited: The Frequency and Placement of Topic Sentences in Academic Writing*, The Reading Matrix, **8**(1), pp. 78-95, 2008.

[9] Theijssen, D., *Features for Automatic Discourse Analysis of Paragraphs: Finding Features to Detect Rhetorical Relations Between Sentences Within Paragraphs*, Master thesis, Department of Linguistics, Radboud University Nijmegen, 2007.

[10] Hyland, K. & Tse, P., *Metadiscourse in Academic Writing: A Reappraisal*, Applied Linguistics 25/2, pp. 156-177, Oxford University Press, 2004.

[11] Kupiec, J., et al., *A Trainable Document Summarizer*, ACM SIGIR, 1995.

[12] Teufel, S., Moens, M., *Sentence Extraction as A Classification Task*, Proceedings of the ACL, 1997.

[13] ACL Anthology Reference Corpus (ACL ARC): http://acl-arc.comp.nus.edu.sg/ (August 2009).

[14] Bird, S., et al., *The ACL Anthology Reference Corpus: A Reference Dataset for Bibliographic Research in Computational Linguistics*, in Proc. of Language Resources and Evaluation Conference (LREC 08), Marrakesh, Morocco, May 2008.

[15] *Stanford Parser: a statistical parser*, The Stanford Natural Language Processing Group, http://nlp.stanford.edu/software/lex-parser.shtml, March 18th, 2010.

[16] *The Penn Treebank Project*, http://www.cis.upenn.edu/~Treebank/, October 2nd, 2010.

[17] *Relationship between sentences*, http://www1.fccj.org/lchandouts/readinglabhandouts/R6%20Rel.%20bet.%20Sentences.doc , April 22nd, 2010.

[18] *WordNet: a lexical database for English*, Princeton University, http://wordnet.princeton.edu/, December 10th, 2009.

[19] *MIT Java Wordnet Interface*, MIT, http://projects.csail.mit.edu/jwi/ , December 10th, 2009.

[20] *MRC psycholinguistics database*, http://www.psy.uwa.edu.au/mrcdatabase/uwa_mrc.htm , March 22nd, 2010.

[21] *jMRC - MRC Psycholinguistic Database Java Interface v0.9*, http://mi.eng.cam.ac.uk/~farm2/jmrc/index.html , March 22nd, 2010.

[22] Kohavi, R. & John, G., *Wrappers for feature subset selection*, Artificial Intelligence, **97**(1-2), pp. 273-324, 1997

[23] Paz, E.C., et al., *Feature Selection in Scientific Applications, in Proc. International Conference on Knowledge Discovery and Data Mining*, 2004

[24] Joachims, T., *Learning To Classify Text Using Support Vector Machines: Methods, Theory, and Algorithms*, Dissertation, University Dortmund, Kluwer Academic Publishers, 2001.

[25] Chih-Chung Chang and Chih-Jen Lin, *LIBSVM -- A Library for Support Vector Machines*, http://www.csie.ntu.edu.tw/~cjlin/libsvm/ , November 19th , 2009

[26] Platt, J.C., *Probabilistic outputs for support vector machines and comparison to regularized likelihood methods*, in Advances in Large Margin Classifiers, MIT Press, 2009.
[27] Lin, H.T., et al., *A Note on Platt's Probabilistic Outputs for Support Vector Machines*, Technical Report, Department of Computer Science, National Taiwan University, 2004
[28] Hsu, C.W., et al., *Practical Guide to Support Vector Classification*, http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf, December 16th, 2009.
[29] Sebastiani, F., *Machine Learning in Automated Text Categorization*, ACM Computing Surveys, **34**(1), March 2002.