ICIC Express Letters
Part B: Applications
Volume **12**, Number **2**, February **2021**

ICIC International ©2021 ISSN 2185-2766
pp. 111–119

TOWARDS BETTER ARGUMENT COMPONENT CLASSIFICATION IN ENGLISH ESSAYS

Rheza Winata, Evan Giovanni Haryono and Derwin Suhartono

Computer Science Department
School of Computer Science
Bina Nusantara University
JL. K. H. Syahdan No. 9, Kemanggisan, Palmerah, Jakarta 11480, Indonesia
{ rheza.winata; evan.haryono }@binus.ac.id; dsuhartono@binus.edu

Received June 2020; accepted August 2020

Abstract. *Due to recent technologies advancement, research in argument mining has been growing very fast. A huge gap of different technical environment might affect the studies in a form of performance comparison. In this paper, we attempted to test all previous work in one test environment for more accurate feature extraction analysis. Our research focused on the clause-level argument classification task. We measured each feature group from other studies including our own features which combine several well-defined features. For feature extraction, group 54 features into 8 sub-groups: structural, lexical, syntactic, contextual, indicator, embedding, probability, and similarity. With XGBoost (eXtreme Gradient Boosting) and SMO (Sequential Minimal Optimization) as the learning model, we were able to obtain the slightly higher accuracy than the state-of-the-art, which is 81.70%. We also found that some extraction methods from the previous study performed better when tested in other environment.*
Keywords: Argument mining, Feature extraction, Clause-level, Argument classification

1. **Introduction.** Argumentation mining is a research area within the natural language processing field that aims to extract the idea from dialogues, essays, and other texts. There are tasks that were found to be beneficial by utilizing argumentation information mining from the texts, such as troll post-detection, knowledge retrieval, review analysis [1], information validation and argument assessment in an essay [2]. One of argument mining implementation is the fact detection and source identification in social media [3]. Improving the ability of argumentation mining may lead to more possibilities and abilities for artificial intelligence applications [4].

The goal of argument mining is the automatic extraction and identification of argumentative structures from natural language text with the aid of computer programs [5]. By integrating argumentation mining in writing environments, we would be able to inspect argument text and improve the quality of the argumentation [6].

One of argumentation mining tasks is to classify an argument. Based on well-established argumentation theories [7], an argumentation consists of several argument components, such as claims and premises. Claim is a controversial statement or arguable statement that should be accepted by the reader without additional support, and premise are reasons for justifying the claim [6].

"...Moreover, (1) earning money is the fundamental reason why people work. (2) The amount of money is always the most important criterion for everyone to choose a job. If I work for other people, I have to give my ideas to employers cheaply. Therefore, I never have been rich in my life, because my salary is not dramatically changed. In spite

DOI: 10.24507/icicelb.12.02.111

of many problems of having own business, I can earn more money based on my creativity and personal ideas.

To sum up, in spite of the fact that (3) many people are not eager to work for themselves, (4) I would like to run my own business to control my own time, and to earn more money based on my noble ideas."

Paragraph above represents example of argument annotations. In this example, arguments (1) and (2) are premises, supporting the claim for (3) and (4). The argument (3) is a claim, while the argument (4) is the major claim.

The classification of argument component and visualization has several advantages, such as to show clear, strong, and structured/organized arguments. Thus, having better accuracy in classifying argument components becomes a major problem [8]. A study from Aker et al. [5] shows that there is more available improvement by analyzing previous work from Stab and Gurevych [6]. With the aim of better classification and analysis, we test all previous work in one test environment. We managed to improve the argument classification performance by modifying the feature extractors.

In Section 2, we examine the works that lead to our research. The data, feature extractions and algorithms are covered in Section 3. The results are reviewed by comparing the performance of each result in Section 4. Finally, conclusions are presented in Section 5.

2. **Related Works.** Most previous research separates the argument mining into three subtasks: argument identification, argument classification, and argument relation classification. Argument identification task is to identify argument and non-argument in text. Argument classification puts each argument into Argument Discourse Units (ADU) types: thesis, conclusion, premise, and none. Lastly, argument relation classification connects the argument components in terms of support or attack between two arguments. Recent study [9] addresses all of three subtasks in end-to-end argument mining analysis. They examined the pipeline approach problem and proposed a novel objective function that enables F-score to be maximized directly by an Integer Linear Programming (ILP) framework solver.

Experiment from Palau and Moens [10] classified a sentence-level argument into a claim, premise, and non-argumentative. Rooney et al. applied kernel methods for classifying sentence-level argument as either claim, premise, or non-argumentative [11]. Stab and Gurevych initiated a novel approach for identifying argumentative discourse structures in persuasive essays [6]. They did research on all of three subtasks. The argument classifications annotate the clause-level argument into four classes: *Major Claim*, *Claim*, *Premise*, and *None*. These annotations are related to ADU types, for example, Major Claim is synonymous to Thesis since they both represent the main idea of the argument. They used 5 main features: structural, lexical, syntactic, indicators, and contextual for identifying an argument. In their research, they used their own corpus, containing 90 persuasive essays compiled by themselves, trained with Support Vector Machine (SVM) [12] classifier with 10-fold cross-validation method. They achieved 77.30% accuracy in terms of argument classification.

While they classify a component into non-argument, other study separates an argumentative and non-argumentative in argument identification before classifying the argument types; hence removing the None class is plausible. For instance, some sentences were manually annotated to be non-argument and argument, and then extraction of tenses and moods of verbs in the argument were made as well [13]. Kwon et al. proposed 2 consecutive steps for classifying subjective claims [14]. First, they identified claims in sentences, and then classified each claim as either support, oppose, or propose. They achieved 67% accuracy in their classification system, which used a boosting algorithm implemented in BoosTexter [15].

As continuation from previous research [12], they use a new corpus consisting of 402 persuasive essays and DKPro Core framework for feature extraction and classification [16]. They classified the clause-level argument as major claim, claim, and premise. Non-argumentative class were excluded because non-argumentative were classified in argument identification task.

There is also an attempt to combine features in argument component classification in persuasive essays by putting additional features in purpose of increasing the accuracy [8]. They classified each sentence into four classes: major claim, claim, premise, and non-argumentative. They have added 7 main features in purpose to explore further in classifying argument components: structural, lexical, indicators, contextual, syntactic, prompt similarity, and discourse features. As for the result, they obtained 79.98% accuracy by combining all features without discourse features. However, their classification method seems to be different from studies they compared with.

Other researches put their focuses on performance improvement in classification task. Nguyen and Litman [17] have proposed an approach for more compact feature spaces, by replacing a sparse feature like n-gram with argument words and domain word. They were able to slightly improve the performance, while also reduce a huge amount of feature space from n-gram. Contextual features were also considered to be the crucial part of the classification performance [18]. Therefore, removing contextual feature might be an improvement in the performance.

Feature extraction is proven to be the most powerful approach in many tasks, such as for speech recognition [19] as well as in compressing traffic classification [20]. This arrives in an opinion that defining a lot of features to get significant better results is quite promising.

3. **Proposed Methods.**

3.1. **Data.** In this work, we used corpus compiled by Stab and Gurevych [16]. The dataset contains 402 persuasive essays, and in total it has 7,116 sentences with 147,271 tokens, with 751 major claims, 1,506 claims, and 3,832 premises. We implement train-test system and 10-fold cross-validation system. For the train-test system, we randomly split the data into 70% train set and 30% test set with equal class distribution. Each set contains approximately $1 : 2 : 5$ distribution ratio. Because we only classify an annotated argumentative component, we excluded the None class from the classification. The class distribution can be seen in Table 1.

Table 1. Class distribution in corpus

Description	Major Claim	Claim	Premise
Total	751	1506	3832
Train data	519	1049	2697
Test data	232	457	1135

For preprocessing, we use LanguageToolSegmenter to identify tokens and sentences as well as ParagraphAnnotator to identify paragraph. We lemmatize the token by using MateLemmatizer [21] and SnowballStemmer. For Part-of-Speech tag (POS-tag), constituent and dependency parser, we use StanfordParser. All of pre-processors are available in the framework made by UKP Lab, DKPro [22].

3.2. **Features.** For our custom feature extraction, we combined all features that have been proposed by other research, such as from Stab et al. [6,8,16], in attempt to measure the highest possible accuracy in exchange of longer processing time. In total, there are 54 features divided into 8 groups: structural, lexical, syntactic, indicator, contextual, word embedding, similarity, and probability.

3.2.1. *Structural features.* Structural features define the argument position and size in the essay. For statistic features, we define a number of tokens for the argument and its covering sentence, then a number of sentences in the covering paragraph and number of punctuations in the covering sentence. In addition, we count the number of following and preceding components in the covering paragraph. For location features, we define the position of argument in covering paragraph, and in the covering sentence. For binary features, we checked if the covering sentence contains a question mark. We also checked if the argument component is the first or the last argument in the paragraph, and whether it is present in the introduction or conclusion of the essay. Lastly, we add ratio features proposed by Desilia et al. [8], consisting of token ratio between covering sentence and paragraph, covering sentence and essay, and paragraph with essay.

3.2.2. *Lexical features.* For lexical features, we define verbs, adverbs, modals, binary n-gram, binary lemmatized n-gram, and dependency word pairs. For n-gram, we use 500 top 1-3 grams. Some minor test showed that by using both n-gram and lemmatized n-gram we were able to obtain the maximum performance as shown in Table 2, with the cost of more time consumption.

3.2.3. *Syntactic features.* We checked if the argument component has past tense sentence or not by checking the Part-of-Speech (POS) of the argument based on Penn Treebank POS-tag annotation. We also count the depth of parse tree, number of sub-clauses, POS distribution and production rules. For POS distribution we count every POS for each token in argument component as numeric features.

3.2.4. *Indicator features.* Indicator features are mostly binary features that represent one word in a category that is set to true if they are included in the argument component. There are 12 categories for indicator words: first person, connective, thesis, forward, backward, rebuttal, time, conclusion, evidence, cue, contrast and comparison [8]. For connective indicators, we use 55 discourse markers, each as a binary feature that indicates if the marker exists in the preceding argument component. First-person indicators consist of possessive words, such as *I, my, mine, myself,* and *me*.

3.2.5. *Contextual features.* Contextual features are features that depend on the sentence preceding and following the current argument component. We checked whether those sentences have a modal verb or not. We also count their amount of punctuations, tokens, and subclauses. In addition, indicator features were applied to these sentences. Lastly, we count the number of shared noun phrases of those sentences with the introduction sentence and conclusion sentence.

3.2.6. *Probability features.* Probability features are the probability of the argument component to be Major Claim, Claim or Premise, given the sequence of lemmatized tokens from the preceding of argument component. For our experiment, we use the maximum likelihood estimation of the training data [16].

3.2.7. *Word embedding features.* Word embedding features were created by averaging the sum of all word vectors from the argument component's tokens. We use GloVe, a pre-trained 100-unit vector to create the vector representation of each word from all essays [21].

3.2.8. *Prompt similarity features.* Prompt similarity features compare the cosine value of token frequency from two sentences [8]. We compare the covering sentences with each of the following sentences: the previous sentence, the next sentence, the first sentence, and the last sentence.

3.3. **Classifiers.** We used 2 learning models: Sequential Minimal Optimization (SMO) and then eXtreme Gradient Boosting (XGBoost). SMO is known to yield good results in argument classification. Meanwhile, XGBoost is an implementation of gradient boosted decision trees designed for speed and performance.

3.3.1. *Sequential Minimal Optimization (SMO).* SMO is a simple algorithm proposed by Platt [24] with a purpose to solve QP (Quadratic Programming) problem on the SVM (Support Vector Machine). It was later improved by Keerthi et al. [25]. As a result, SMO is able to solve the smallest possible optimization problem and requires no extra matrix storage at all. SMO also has linear scale of time consumption with training data size, compared to standard chunking SVM algorithm which scales linear to cubic.

3.3.2. *eXtreme Gradient Boosting (XGBoost).* XGBoost is a new open-source learning model made by [26]. XGBoost is a decision-tree based machine learning model that uses gradient boosting frameworks and it aims to provide a "Scalable, Portable, and Distributed, Gradient Boosting". XGBoost algorithm can also handle large-scale data using a fast-parallel tree construction which cuts the training time significantly and yet still provides the effectiveness of gradient boosting algorithm. Therefore, many data scientists used this to achieve state-of-the-art results on many machine learning challenges. One of them is a research for attention-based argument mining [27], where XGBoost classifier outperforms the fully connected layer due to the massive number of parameters or weights that are needed to be trained in the fully connected neural network.

4. **Results and Discussion.** We followed the test environment from Stab [14] as the baseline of performance shift. The test was done with DKPro TC [18], a framework that is extended from DKPro Core. The processing pipeline can be seen in Figure 1. We classify the data into 3 classes: major claim, claim, and premise.

There are 3 works that we decided to reproduce. The first and second are from Stab and Gurevych [6,16], which were done with DKPro Core framework. The third benchmark is from Desilia et al. [8], which was made without a framework. By unifying the environment,

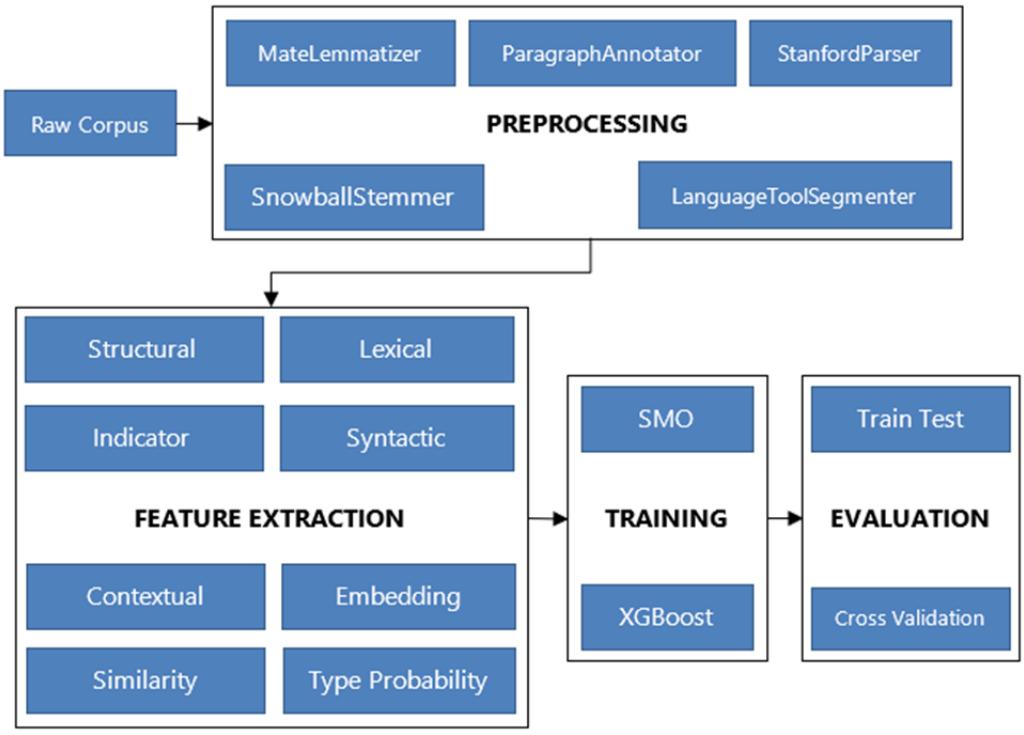

FIGURE 1. Argument classification process pipeline

Table 2. Performance comparison

Related works	SMO	XGBoost
– 10-Fold Cross-Validation –		
Stab and Gurevych (2014)	81.31%	75.08%
Stab and Gurevych (2016)	77.40%	77.18%
Desilia et al. (2017)	80.68%	78.32%
Combined	81.70%	80.39%
– Train Test –		
Stab and Gurevych (2014)	80.86%	75.43%
Stab and Gurevych (2016)	77.35%	77.08%
Desilia et al. (2017)	80.64%	77.85%
Combined	81.63%	79.22%

we hypothesized that the result will be less biased and more comparable. Table 2 shows all test results in both train-test system and cross validation system.

We found that the both experiments [6,8] were able to obtain higher accuracy than they were initially proposed. Meanwhile, the baseline experiment itself does not change. This indicates the shifting of performance result when it was done in different test environment. Our combined features also obtained a slightly better performance in both test systems. The learning model comparison shows that SMO performs better than XGBoost. However, XGBoost's performance is worth to be considered for its computer resource management which results in less time consumption. The SMO learning model takes 26-57 seconds and meanwhile the XGBoost learning model only takes 1-4 seconds.

We took a deeper look at each of the confusion matrixes obtained from train-test system. Each matrixes is shown in Table 3.

Table 3. Confusion matrix

(a) Stab and Gurevych (2014)

Actual	Prediction		
	Major Claim	Claim	Premise
Major Claim	204	27	1
Claim	45	293	119
Premise	8	149	978

(b) Stab and Gurevych (2016)

Actual	Prediction		
	Major Claim	Claim	Premise
Major Claim	205	27	0
Claim	47	148	262
Premise	14	63	1058

(c) Desilia et al. (2017)

Actual	Prediction		
	Major Claim	Claim	Premise
Major Claim	199	33	0
Claim	45	296	116
Premise	4	155	976

(d) Combined Features

Actual	Prediction		
	Major Claim	Claim	Premise
Major Claim	203	29	0
Claim	39	300	118
Premise	8	141	986

By comparing Table 3(a) and Table 3(d), we found that our customized feature groups were able to detect 7 more claims and 8 more premises. However, our customized feature group consumes half more time to extract features compared to works from Table 3(a). Further modifications are needed to improve our feature extraction in purpose of optimization for both accuracy and time consumption. Meanwhile, results from Table 3(c) performed almost as good as the second-best. The confusion matrix shows that it tends to predict more claims than any other works. We also found that Table 3(b) was the best

in terms of predicting the premise, with 72 more premises predicted than our combined features.

Furthermore, we customize our combined feature by removing some of the subgroup of features to analyze the impact of each feature group. We test these feature groups by 10-fold cross-validation to prevent dataset bias from the train-test system. The results can be seen in Table 4.

Table 4. Combined features customization using 10-fold cross-validation evaluation

Features	**SMO**	**F1 MC**	**F1 C**	**F1 P**
Structural only	77.76%	77.40%	53.07%	86.94%
Lexical only	70.76%	28.65%	66.58%	81.89%
Contextual only	69.50%	57.63%	21.77%	83.35%
Syntactic only	65.36%	33.02%	10.40%	78.79%
Indicator only	70.71%	65.99%	30.20%	81.41%
Embedding only	62.93%	0%	0%	77.25%
Probability only	65.23%	43.91%	0%	79.25%
Similarity only	68.82%	0%	51.99%	81.47%
...				
w/o Structural	79.78%	78.96%	61.04%	87.38%
w/o Lexical	81.09%	83.47%	62.98%	87.96%
w/o Contextual	81.70%	84.55%	64.16%	88.24%
w/o Syntactic	81.70%	84.59%	64.14%	88.23%
w/o Indicator	81.70%	84.44%	64.25%	88.22%
w/o Embedding	81.67%	84.10%	64.35%	88.24%
w/o Probability	81.34%	83.95%	63.45%	88.02%
w/o Similarity	81.65%	84.31%	64.23%	88.19%

Structural and lexical features have been proven to be important features for argument component classification. Structural features contribute a lot for identifying major claim and premise, because most major claims are structurally identical as they are in the first sentence or the last sentence, while premise usually comes in the middle of essays. Both structural and lexical features contribute the most for detecting claim. We hypothesize that claims are almost identical to a major claim structurally and most of claims are identifiable by having a modal, such as '*should*, *would*, *might*, ...' to show the idea of the argument.

Meanwhile, omitting either contextual, syntactic, and indicator features did not seem to affect the classification accuracy. However, omitting all three of those features would affect the performance. By observing Table 4, we can see that each omitted feature group has low impact in the accuracy and F-measure, except for structural and lexical features.

5. **Conclusion.** We analyzed each work, by reproducing all previous works with an equal environment. We used an annotated corpus, and classified the argument component into major claim, claim, and premise. From whole conducted experiments, we conclude that:

- Our feature combination works slightly better, increasing performance by 0.8% with train-test system. However, it consumes half more time than the previous work [6].
- It is possible to improve the performance for our feature combination, such as by replacing the n-gram features into argument and domain words [16], or by using partial tree kernel instead of contextual features [17].
- The previous work [6] was able to perform better in our test environment.
- By reproducing all previous works with equal environment, we reduced the performance bias and were able to analyze in more specific context.

Acknowledgment. This work is fully supported by Bina Nusantara University. The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

REFERENCES

[1] H. Liu, Y. Gao, P. Lv, M. Li, S. Geng, M. Li and H. Wang, Using argument-based features to predict and analyse review helpfulness, *arXiv Preprint*, arXiv:1707.07279, 2017.

[2] H. Wachsmuth, K. Al Khatib and B. Stein, Using argument mining to assess the argumentation quality of essays, *Proc. of the 26th International Conference on Computational Linguistics: Technical Papers (COLING 2016)*, pp.1680-1691, 2016.

[3] M. Dusmanu, E. Cabrio and S. Villata, Argument mining on Twitter: Arguments, facts and sources, *Proc. of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp.2317-2322, 2017.

[4] E. Cabrio and S. Villata, Five years of argument mining: A data-driven analysis, *The 27th International Joint Conference on Artificial Intelligence*, pp.5427-5433, 2018.

[5] A. Aker, A. Sliwa, Y. Ma, R. Liu, N. Borad, S. F. Ziyaei and M. Ghbadi, What works and what does not: Classifier and feature analysis for argument mining, *Proc. of the 4th Workshop on Argument Mining*, pp.91-96, 2017.

[6] C. Stab and I. Gurevych, Identifying argumentative discourse structures in persuasive essays, *Proc. of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp.46-56, 2014.

[7] D. Walton, C. Reed and F. Macagno, *Argumentation Schemes*, Cambridge University Press, 2008.

[8] Y. Desilia, V. T. Utami, C. Arta and D. Suhartono, An attempt to combine features in classifying argument components in persuasive essays, *CMNA@ICAI*, pp.71-75, 2017.

[9] I. N. V. Persing, End-to-end argumentation mining in student essays, *Proc. of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp.1384-1394, 2016.

[10] R. M. Palau and M. F. Moens, Argumentation mining: The detection, classification and structure of arguments in text, *Belgian/Netherlands Artificial Intelligence Conference*, pp.98-107, 2009.

[11] N. Rooney, H. Wang and F. Browne, Applying kernel methods to argumentation mining, *Proc. of the 25th International Florida Artificial Intelligence Research Society Conference*, 2012.

[12] L. Wang, *Support Vector Machines: Theory and Applications*, Springer Science & Business Media, 2005.

[13] E. Florou, S. Konstantopoulos, A. Koukourikos and P. Karampiperis, Argument extraction for supporting public policy formulation, *Proc. of the 7th Workshop*, pp.49-54, 2013.

[14] N. Kwon, L. Zhou, E. Hovy and S. W. Shulman, Identifying and classifying subjective claims, *Proc. of the 8th Annual International Conference on Digital Government Research: Bridging Disciplines & Domains*, pp.76-81, 2007.

[15] R. E. Schapire and Y. Singer, BoosTexter: A boosting-based system for text categorization, *Machine Learning*, vol.39, nos.2-3, pp.135-168, 2000.

[16] C. Stab and I. Gurevych, Parsing argumentation structures in persuasive essays, *Computational Linguistics*, vol.43, no.3, pp.619-659, 2016.

[17] H. Nguyen and D. Litman, Extracting argument and domain words for identifying argument components in texts, *Proc. of the 2nd Workshop on Argumentation Mining*, pp.22-28, 2015.

[18] M. Lippi and P. Torroni, Context-independent claim detection for argument mining, *Proc. of the 24th International Joint Conference on Artificial Intelligence (IJCAI 2015)*, 2015.

[19] H. M. S. Naing, R. Hidayat, B. Winduratna and Y. Miyanaga, Psychoacoustical masking effect-based feature extraction for robust speech recognition, *International Journal of Innovative Computing, Information and Control*, vol.15, no.5, pp.1641-1654, 2019.

[20] Z. Tang, X. Zeng and Y. Sheng, Entropy-based feature extraction algorithm for encrypted and non-encrypted compressed traffic classification, *International Journal of Innovative Computing, Information and Control*, vol.15, no.3, pp.845-860, 2019.

[21] B. Bohnet, J. Nivre, I. Boguslavsky, R. Farkas, F. Ginter and J. Haji, Joint morphological and syntactic analysis for richly inflected languages, *Transactions of the Association for Computational Linguistics*, vol.1, pp.415-428, 2013.

[22] J. Daxenberger, O. Ferschke, I. Gurevych and T. Zesch, DKPro TC: A Java-based framework for supervised learning experiments on textual data, *Proc. of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp.61-66, 2014.

[23] J. Pennington, R. Socher and C. D. Manning, GloVe: Global vectors for word representation, *Proc. of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp.1532-1543, 2014.

[24] J. Platt, *Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines*, 1998.

[25] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya and K. R. K. Murthy, Improvements to Platt's SMO algorithm for SVM classifier design, *Neural Computation*, vol.13, no.3, pp.637-649, 2001.

[26] T. Chen and C. Guestrin, XGBoost: A scalable tree boosting system, *Proc. of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp.785-794, 2016.

[27] D. Suhartono, A. Gema, S. Winton, T. David, M. I. Fanany and A. M. Arymurthy, Attention-based argument mining, *International Journal of Computational Vision and Robotics*, vol.9, no.5, pp.414-437, 2019.