Journal of Big Data

RESEARCH Open Access

Enhancing argumentation component classification using contextual language model

Hidayaturrahman[1*], Emmanuel Dave[1], Derwin Suhartono[1] and Aniati Murni Arymurthy[2]

*Correspondence:
hidayaturrahman@binus.ac.id
[1] Computer Science Department, School of Computer Science, Bina Nusantara University, Jakarta 11480, Indonesia
Full list of author information is available at the end of the article

Abstract

Arguments facilitate humans to deliver their ideas. The outcome of the discussion heavily relies on the validity of the argument. If an argument is well-composed, it is more effective to grasp the core idea behind the argument. To grade the argument, machines can be utilized by decomposing into semantic label components. In natural language processing, multiple language models are available to perform this task. It is divided into context-free and contextual models. The majority of previous studies used hand-crafted features to perform argument component classification, while state of the art language models utilize machine learning. The majority of these language models ignore the context in an argument. This research paper aims to analyze whether by including the context in the classification process may improve the accuracy of the language model which will enhance the argumentation mining process as well. The same document corpus is fed into several language models. Word2Vec and GLoVe represent the context free models, while BERT and ELMo as context sensitive language models. Accuracy and time from each model are then compared to determine the importance of context. The result shows that contextual language models are proven to be able to boost classification accuracy by approximately 20%. However, time comes as a cost where contextual models require longer training and prediction time. The benefit from the increase in accuracy outweighs the burden of time. Thus, as a contextual task, argumentation mining is suggested to use contextual model where context must be included to achieve promising results.

Keywords: Argumentation mining, Language model, Context free model, Contextual model, BERT, ELMo, Word2Vec, GloVe

Introduction

Arguing is a daily activity carried out both in speaking and thinking. Arguments are made to increase or decrease the level of acceptance of a controversial idea [1]. They emphasize one's opinions as well by showing their point of view on certain discussions. Good arguments can support the decision-making process and reach a conclusion that can be accepted by many people. Arguments can appear in everyday conversation or in written form. In order to strengthen the validity of an argument, several premises or supporting sentences in form of facts must support the argument. In certain literatures, they can be called as claims and premises [2], fact, value [3], policy [4] and many other proposed argument schemes.

By understanding the structure of an argument, it can help us improve the quality of an argument even to the point of being able to test the validity of the argument itself. This process is called as argumentation mining. It aims to provide machines with the capability to decompose a text and retrieve its main argument and point of view by performing sentiment analysis. With the help of machines, students and authors can be assisted in real time to structure their writing that can strengthen their argumentation position in their texts. There are several steps that must be taken to arrive at a more practical job and to be able to provide direct insight into an argument. Currently, several techniques have been developed to carry out the argumentation mining process.

According to previous research, there are several examples of tasks performed in argumentation mining. Among them are identifying arguments that do not have sufficient support [5], classifying argument component [6], identifying whether an argument is present or not [7], identifying argumentative relations [8], making argument graph [9], and automatic annotator for text containing argument [10].

The majority of argument mining research starts with an argument component classification. Currently, several approaches are available to perform the classification process which depends on the goal of the task. In general, an argument can be divided into 2 main components, namely claim and premise.

In the argumentation mining process, there are several mandatory steps. Initially, sentences must be categorized into claim and non-claim. Afterwards, sentences in the claim category are further classified into the multiple types of claim (11). Another approach would be by classifying sentences directly into claim, premise and non-argumentative (12). Other research grouped sentences into claim and premise types (13). Lastly, sentences can also be classified into major claim, claim, premise, and non-argumentative categories (14).

In terms of the techniques used to perform argument component classification, two approaches are commonly applied, handcrafted features and deep learning. For this research, the deep learning approach is taken as it is proven in the literature that it achieves better results compared to the machine learning approach as it has the ability to learn high-level features from the input data. For techniques that utilize deep learning, two types language models can be used for feature extraction process which are context-free models, such as the Global Vector (GloVe) [11], Word2Vec [12] and FastText [13] and contextual models such as the Bidirectional Encoder Representations from Transformers (BERT) [14] and the Embeddings from Language Models (ELMo) [15].

In argumentation mining, context has an important role to provide additional information, especially to classify argumentation components. This paper will show how a context-sensitive language model can outperform a context-free language model. This paper will also show how a context-sensitive language model trained in a bidirectional manner (e.g. BERT) performs better than a shallow bidirectional model (e.g. ELMo). In order to make the comparison is apple to apple, the experiment is only make use of language model itself then classify the word vector directly using perceptron. Since the model is the only independent variable, time cost will be considered to measure the performance of language models.

Related work

Argumentation mining

Argumentation is a type of discourse where speakers try to persuade their audience about the reasonableness of a claim by presenting supportive arguments. Argumentation mining usually aims to identify argument components contained in the text and then predict the relations between them.

Argumentation mining has been performed using data in the form of an abstract Randomized Controlled Trial (RCT) from the MEDLINE database which is self-annotated. The dataset used in this study has 4,198 argument components and 2,601 argument relations retrieved from several types of diseases (i.e., neoplasm, glaucoma, hepatitis, diabetes, hypertension). In this study, a pipeline is proposed in which argument components are classified into evidence or claim, then predicting the relationship as support or attack. Experiments were performed using bidirectional transformers in combination with various neural architectures, such as the Long Short Term Memory (LSTM), the Gated Recurrent Units (GRU), and the Conditional Random Field (CRF). The result of this experiment show an F1-score of 0.87 for component detection and 0.68 for relation prediction [16].

Another approach is proposed to analyse the support relation between two-argument sentences. In conducting this research, the presence or absence of support is emphasized on the similarity of the nuances in the two sentences. Experiments were carried out using a siamese network whose extractor feature was replaced with Long Short Term Memory and using the cosine distance as an energy function. This model accepts input in the form of a pair of sentences then tries to predict whether there is a support relation or not. The main objective of this research is to prove that the level of similarity between two sentences is related to the support relationship between sentences. In the experiment, an attention mechanism was also implemented. The accuracy value of this experiment is 67.33% [17].

The trend in classifying argument components is polarized into the use of statistical models with heavy feature-engineering or the use of direct deep neural-networks without considering prior knowledge or another secondary feature. Therefore, this research proposes to add lightweight features to deep models to classify argument components. Experiments were carried out by comparing the performance of the proposed model with previous studies. It managed to outperform the others by achieving a macro F1-Score of 0.805. Based on these findings, it is concluded that adding prior knowledge, can help improve the performance of the model [7].

Argumentation mining is used to enhance a lot of activity such as peer review assistance in research paper [18], mining argumentation on tweets in twitter [19] and other social media [20]. Multilingual argumentation mining has also been done in a research that is conducted by performing transfer learning and utilizing machine translation [21].

Language model

Since the deep learning trend has increased, many language models have been built using a deep learning approach in order to achieve a more promising result. From the perspective of the representation of a word in the language model, there is a context-free model and a contextual model. In the context-free model the word "bank" in "deposit bank" and "river bank" have the same value. On the other hand, a contextual model will assign different values to the word “bank” in this scenario as it has different meanings. Examples of context-free models are Word2Vec [12], GloVe [11] and FastText [13]. Meanwhile, examples of contextual models are BERT [14] and ELMo [15].

Word2Vec is a language model that is trained using the skip-gram approach. The improvement made in this training model is by subsampling frequent words to make the training process faster and also learn more regular word representations. This technique is later known as negative sampling.

The GloVe is a language model that is created to produce a more explainable model. During the modeling process, its properties are given explicitly, such as regularities to emerge in word vectors. The resulting model is a global log-bilinear regression that combines a global matrix factorization and a local context window. This model can produce vector space with a meaningful substructure.

As for FastText, it is actually extended version of Word2Vec. The different is FastText optimize the use of subword to do skipgram training. It make FastText is more sensitive to rare world. In their research, it found that the time needed to do the traning is $1.5 \times$ slower than the skipgram baseline.

Unlike recent language representation models, BERT is designed to pre-train deep bidirectional representations from an unlabeled text by jointly conditioning on both left and right context in all layers. BERT does not simply concatenate left-to-right and right-to-left contexts like in ELMo model. Concatenation is a weakness as it is unable to simultaneously consider left and right contexts. The transformer encoder in each layer reads the whole sequence of inputs all at once. This way, transformers can learn the context of a specific word based on its neighboring words (both left and right). As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art model for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications. At the time BERT was created, it was reported that this model obtained new state-of-the-art results on eleven natural language processing tasks, including pushing the General Language Understanding Evaluation (GLUE) score to 80.5% (7.7% point absolute improvement), the Multi-Genre Natural Language Inference (MultiNLI) accuracy to 86.7% (4.6% absolute improvement), the Stanford Question Answering Dataset (SQuAD) v1.1 question answering Test F1 to 93.2% (1.5% points absolute improvement) and SQuAD v2.0 Test F1 to 83.1% (5.1% points absolute improvement).

Several preprocessing methods are required to fit data into the BERT model. Data should be initially tokenized using Bert Tokenizer as the model has its own fixed vocabulary. In addition, data should be annotated using special tokens recognized by BERT. For instance, [CLS] token symbolizes the start of a sentence and [SEP] token separates one sentence from another. In addition, the [CLS] token signals a classification layer in order to perform classification tasks. Even though the input is a single sentence, [SEP] token is still required to be added before fitting inputs into the model. Another important token is [PAD]. This token is represented with value 0. Considering that BERT is an absolute positional embedding, padding tokens are preferably placed on the right of the inputs.

An additional sequence of input is needed to annotate which tokens that the BERT model should pay attention to. This way the model can differentiate positions of a padded token with a tokenized input. In BERT, attention mask with value of 1 represents values that it should pay attention to and 0 represents a padded value.

DistilBERT is a smaller and lighter version of BERT. It is designed to be more effective in terms of speed as it is trained for a general-purpose language model. The main objective is to cut cost to make it more reliable for production. Other BERT models are too large for production, especially when there are certain constraints to be followed, such as latency and servers. This model mimics the BERT model with the addition of several changes. Starting from cutting token-type embeddings and the pooler. This model is also trained using the same corpus used for training the BERT model (English Wikipedia and Toronto Book Corpus). The technique used that can make DistilBERT much lighter compared to BERT model is distillation, a compression technique that helps a child model (DistilBERT) to be trained and show similar behavior with the parent model (BERT) [22] [23]

ELMo is also a contextual model. The word representation in ELMo is a combination of each existing layer to produce a deep representation. Also, the word representation in this model is character-based. If there is a word that is not densely populated in the corpus, the similarity will be sought. This model is trained using a bidirectional language model.

Methodology

Data

Argumentation dataset was gathered from an online forum and classified into 4 classes, namely: Major Claim, Claim, Premise and None. Data were gathered from a total of 90 essays. The distribution of data is represented in Fig. 1. The dataset is dominated by the Premise class with 3,832 statements, followed by the None class which contains statements that cannot be classified to neither Major Claim, Claim nor Premise with 1,610 statements. Similarly to the None class, the claim class contains 1,506 statements. The class with the smallest data is Major Claim with only 751 sentences. [6]

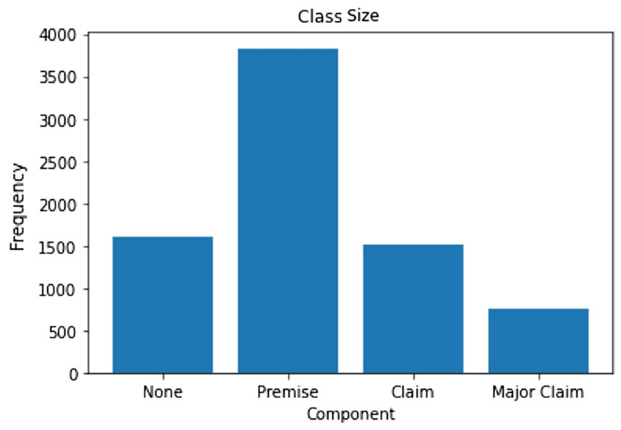

Fig. 1 Dataset distribution

After dividing data into each class, data was randomly separated for training and testing by using a random seed. Random seeding is crucial to ensure that the training and testing dataset are the same for all models. The ratio between the 2 classes is 80% training and 20% testing.

Preprocessing

Contextual language model

Before fitting data into the model, raw text data were encoded based on the corresponding tokenizer of each model. The example code below shows the encoding process for the BERT base cased model by using a BERT tokenizer. BERT tokenizer is required as it

```
tokenizer = BertTokenizer.from_pretrained('be
rt-base-cased')

input_ids = []

for sent in sentences:
  encoded_sent = tokenizer.encode(sent, add_s
  pecial_tokens = True)
  input_ids.append(encoded_sent)
```

has its own fixed vocabulary.

From our data, the maximum statement length is 76, thus statements were padded to a single constant length of 80. Padding tokens are set to value of 0 as BERT will automati-

```
input_ids = pad_sequences(input_ids, maxlen=M
AX_LEN, dtype="long", value=0, truncating="po
st", padding="post")
```

cally treat it as a padding token.

```
attention_masks = []

for sent in input_ids:
  att_mask = [int(token_id > 0) for token_id
in sent]
  attention_masks.append(att_mask)
```

Finally, attention masks were added to differentiate tokens with padding tokens.

For comparison purposes, the dataset was tested using multiple BERT models (i.e. BERT Base Cased with 80 sequences, BERT Base Cased with 256 sequences, and BERT Base Uncased with 80 sequences). On BERT base cased, BERT model is trained using cased word data. It would be compared with the same BERT model but with a longer sequence length of 256. The difference between BERT Base Cased and BERT Base Uncased is the BERT base uncased is trained using uncased words. To compare these BERT models, the training time and final accuracy result from each model were compared.

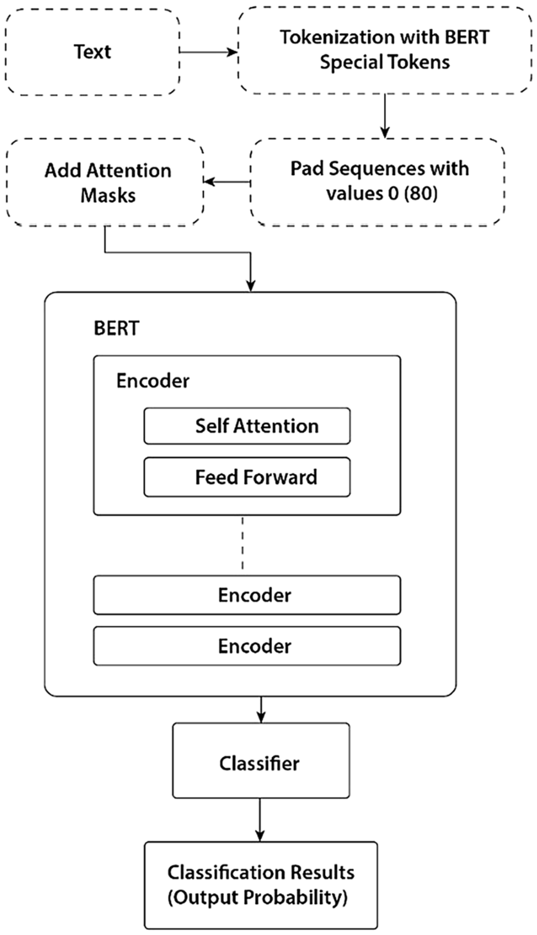
Text
Tokenization with BERT Special Tokens
Pad Sequences with values 0 (80)
Add Attention Masks
BERT
Encoder
Self Attention
Feed Forward
Encoder
Encoder
Classifier
Classification Results (Output Probability)

Fig. 2 BERT training pipeline

Finally, data were converted into PyTorch data types and fitted into a torch data loader for iterative sequence insertion so that the entire dataset is not required to be loaded into memory for training. Figure 2 represents the training pipeline for BERT models.

In this research, the ELMo embedding is added as a lambda layer in the deep neural network architecture as shown in Fig. 3. Therefore, the tokenization and embedding process are executed inside the training pipeline. As soon as the neural network accepts an input text, the input text is directly tokenized and embedded by the ELMo language model. Prior to passing the input text to the neural network model for training, the dataset is initially preprocessed to convert label values into categorical values.

Context free language model

Similar with another language model, context free language model has its own encoding for each token in its dictionary. So before doing the modeling process, all sentences in datasets are converted into tokens using its own dictionary.

For the FastText model, labels must be configured to meet the required format as shown below. It requires adding a "__label__" prefix before passing it into the supervised

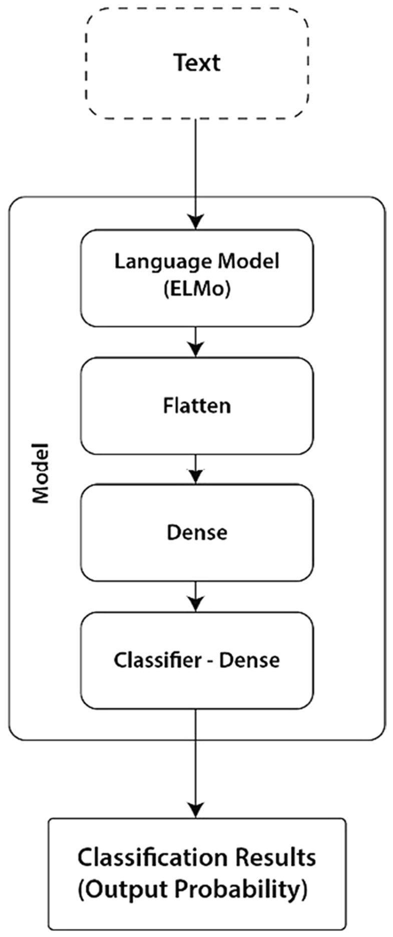

Fig. 3 ELMo training pipeline

training. This preprocessing is needed as the model will treat the label as the subword

```
majorclaim -> __label__majorclaim
```

marker.

Modeling

For comparison purposes, all models were trained using 4 epochs and batch size of 32. All models were trained and tested in a Google Colab notebook with a Tesla T4 GPU.

Table 1 BERT architectures

BERT model	Encoders	Attention heads	Hidden layers
BERT Base	12	12	768
DistilBERT Base	6	12	768

Context-free

In this research, Word2Vec, GloVe and FastText are used as Context-Free Language Models. In context-free language models, each word has its own fixed embedding. Since these models share the same behavior, the modelling process is similar.

Word2Vec and GloVe models experienced similar preprocessing pipeline. First, the text that have been padded with length of 80 is transformed into their token. Then, these tokens are fed into the language models. In this modelling, a 100 dimension language model is used. This means that for each word, it will be translated into 100 array. After that, these sentences vector will be flattened by concatenating all of the word array. These vectors will go through 768 Dense layers before being classified into 4 different classes.

FastText model underwent a different pipeline as the model is already well encapsulated and further data preprocessing occurred during training. After labels are added by the "__label__" prefix, it is directly passed onto training.

Contextual

The learning rate used is 2e-5 and the epsilon is 1e-8.

There are Two types of BERT model, DistilBERT and BERT were tested and compared. Details regarding the architecture of these models are shown in Table 1. The differences between cased and uncased models are not shown in the table as there is only a small difference in the number of parameters, while the other attributes are the same. Uncased BERT models are trained on text that are initially lower cased before it is passed into training, while cased BERT models are trained on case sensitive text. In other words, cased BERT models process input text as it is. Figure 4 shows how cased BERT models and uncased BERT models process the same input text differently.

Both BERT and DistilBERT follow the same training pipeline as DistilBERT is derived from the BERT model.

The training pipeline used for ELMo text classification is shown in Fig. 3. The training pipeline is similar to the context-free training pipeline shown in Fig. 5. However, the difference is that this model can take in sequences of various lengths. Therefore, it is unnecessary to pad to a fixed length of 80. The embedding is inserted as a lambda layer in the model architecture, followed by a fully connected layer to complete the neural network model.

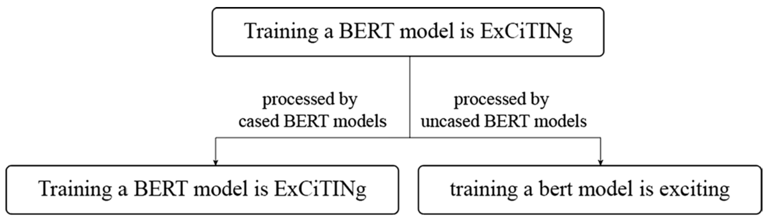
Training a BERT model is ExCiTINg
processed by cased BERT models
processed by uncased BERT models
Training a BERT model is ExCiTINg
training a bert model is exciting

Fig. 4 Cased BERT vs uncased BERT

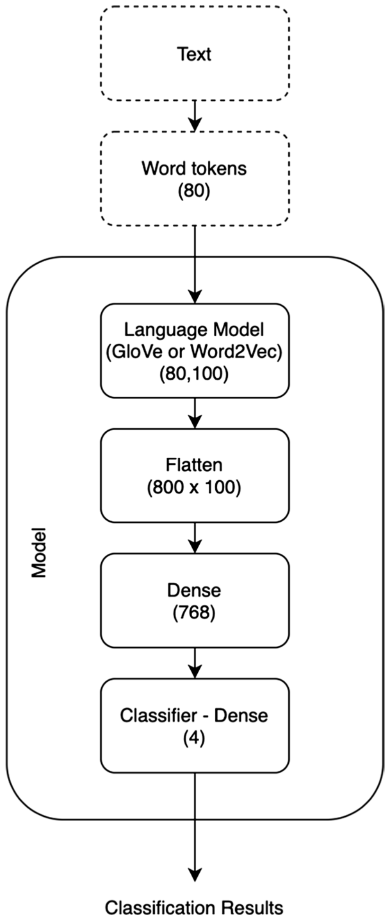

Fig. 5 Context free training pipeline

Results and discussion

Model performance

The difference between using a cased and uncased model does not produce a significant difference on the result. For this specific task, letter casing can be ignored as it does not affect the accuracy of the model. It may be important to pick either cased or uncased model when working with other languages, where cased and uncased letters should be highlighted as it may have different context/meaning.

For context-free models, GloVe model and Word2Vec model share similar performance in general regarding to argumentation classification task. However, for classifying Claim text, Word2Vec perform better than GloVe. In contrast, GloVe model is better on evaluating Major Claim class. For both Premise and None class, these two models performance is quite similar. The difference advantage of these two models is really affected by the corpus that used to train these models. Also Word2Vec model has disadvantages handling Out-of-Vocabulary (OOV) words while GloVe can handle it.On the other hand, the FastText model is the worst performing context-free model. However, it achieved best performance in the largest class dataset which is the Premise class.

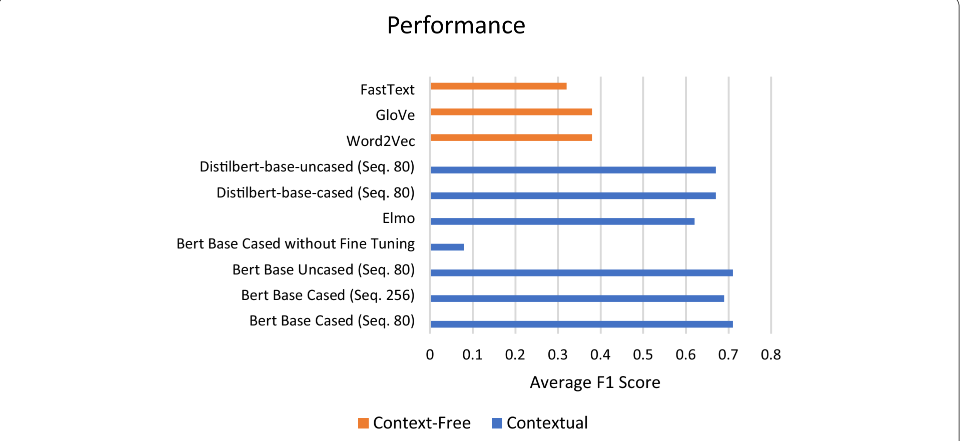

Fig. 6 Performance results

Table 2 F1 score

Language Model	**Class**				**Average**
	Major Claim	**Claim**	**Premise**	**None**	
Bert Base Cased (Seq. 80)	**0.54**	**0.50**	0.80	**1.00**	**0.71**
Bert Base Cased (Seq. 256)	0.47	0.47	**0.82**	**1.00**	0.69
Bert Base Uncased (Seq. 80)	0.53	0.49	**0.82**	**1.00**	**0.71**
Bert Base Cased without Fine Tuning	0.00	0.31	0.00	0.00	0.08
Elmo	0.47	0.23	0.79	0.98	0.62
Distilbert-base-cased (Seq. 80)	0.44	0.44	0.79	**1.00**	0.67
Distilbert-base-uncased (Seq. 80)	0.44	0.43	0.80	**1.00**	0.67
Word2Vec	0.10	0.31	0.66	0.44	0.38
GloVe	0.21	0.21	0.67	0.43	0.38
FastText	0.10	0.00	0.72	0.44	0.32

Bold emphasis data highlight the best performing model for each column

To measure the performance of each model, the F1 metric is used because it can fairly judge the performance of each model considering that there is an uneven class distribution in the dataset. In addition, the performance metric is also unbiased towards either false positives or false negatives as it is a weighted average of precision and recall. Based on Fig. 6, the accuracy of contextual models is significantly higher compared to context-free models.

The least accurate fine-tuned contextual model is ELMo. This model achieves an average F1 score of 0.62. However, this model still outperforms the best performing context free model that achieved an average F1 score of 0.38. Details of how each model performed in each class is shown in Table 2. Hence, it shows that context is critical for text classification as contextual models outperform context-free models.

Since time is equally important to the performance of each model, the prediction time and training time of each model are compared and shown in Table 3. In order to keep this comparison fair, the training and testing dataset are kept constants for all

Table 3 Modelling time

Time	Training time	Prediction time
Bert Base Cased (Seq. 80)	0:06:12	0:00:04
Bert Base Cased (Seq. 256)	0:17:50	0:00:08
Bert Base Uncased (Seq. 80)	0:05:47	0:00:04
Bert Base Cased without Fine tuning	0:00:00	0:00:03
Distilbert Base Cased (Seq. 80)	0:03:00	0:00:02
Distilbert Base Uncased (Seq. 80)	0:03:03	0:00:02
Elmo	1:33:28	0:04:55
Word2Vec	0:00:03.95	0:00:00.11
Glove	0:00:05.82	0:00:00.16
FastText	0:00:00.01	0:00:00.01

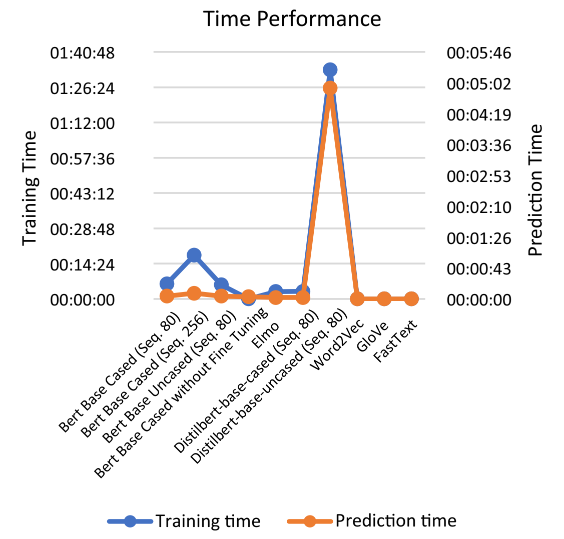

Fig. 7 Time results

models by using a random seed. In addition, the environment that the training and testing took place is also the same for all models. The purpose of tracking the time is to compare the speed between models on the same device and dataset and to determine whether there is a tradeoff between time and performance. It is to answer the question "Does high performing models require high training time?".

Elmo model takes the longest training time with 1 and a half hour training time from all models as shown in Fig. 7. This figure also proves that transformer-based contextual models succeed to achieve state-of-the-art results with minimum time.

The shortest processing time for the contextual model is in DistilBERT models taking only 3 min training time and 2 s prediction time. This shows that DistilBERT is lighter and faster compared to BERT models that approximately needs 5 min training time and 4 s prediction time. The DistilBERT model takes only half of the prediction time taken in the BERT model. Compared to the context free model that took the longest time, the DistilBERT model still takes 30 times more training time and approximately 13 times more prediction time. This proves that even though DistilBERT is able to outrun BERT model by half the time, it is still much longer compared to context free models.

In contrast with the model's accuracy, changes in sequence length affects the processing time quite significantly. The same BERT model with sequence length of 256 needs an additional 11 min and 38 s training time compared to a sequence length of 80. In addition, prediction time in the 256-sequence model is almost twice of the 80-sequence model.

Cased and Uncased BERT model does not affect the processing time quite significantly. For this task, the cased BERT model needs an additional 25 s compared to the uncased model. This is because more variations of embeddings are present in the cased model, as same words with the same context may have different word embeddings when there are differences in letter case.

For context free models, the time taken for training and prediction varies between models. In general, they all took less than 6 s training time and less than a second prediction time. The longest training time is with the GloVe model taking almost 6 s training time. On the other hand, the FastText model is the fastest context free model. It only took less than a second training time and prediction time. It probably happened because GloVe model deals with subwords.

Context free model takes significantly lesser time to train compared to contextual model. This is proven by the fastest contextual model is trained for 3 min, while the longest context free model is only trained for approximately 6 s. This is because a contextual model has a far larger vocabulary size compared to a context free model, as same words with different context will have different embeddings. As mentioned before, the word "bank" in a context free model only has 1 token, while it can potentially have multiple different tokens in a contextual model depending on the context of the word "bank" used in the statement. Hence, contextual models will have a far larger embed/token input size

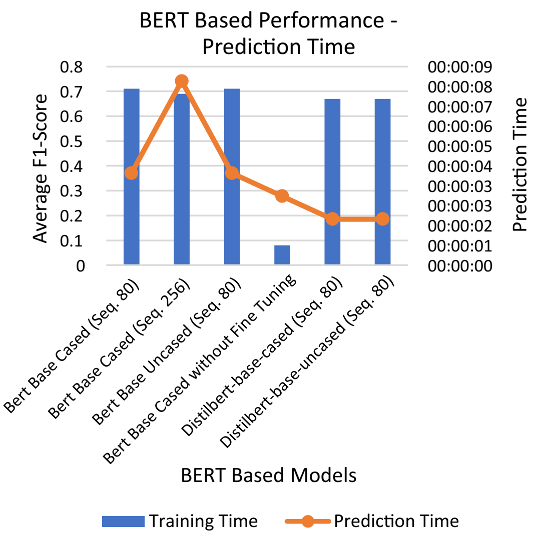

Fig. 8 BERT based performance—prediction time

Table 4 Premise sentence prediction

Sentence	True label	Predicted label
There are several reasons why animal testing should be banned	Claim	Claim
Sentence	**True label**	**Predicted label**
From the health point of view, schools should not only deliver academic subjects	Premise	Major claim

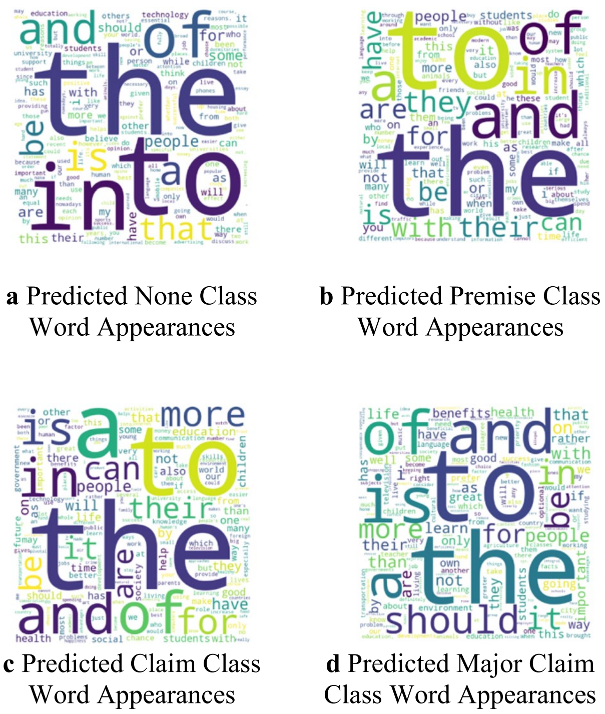

Fig. 9 Claim sentence prediction

which is why the time taken for contextual models is much longer compared to context free models.

Prediction time is more important to be taken into consideration for performance analysis compared to training time. Because training time can be handled using a high computing system, while the prediction process can take place in a variety of devices when deployed for commercial use. Figure 8 highlights the comparison of performance and prediction time for the BERT-based models. It shows that DistilBERT models are more efficient compared to BERT based models, as it achieves similar F1-Score but with lesser time.

Error analysis

For analysis, the BERT based cased model with sequence length of 80 is selected as it is the best performing model.

As shown in Table 4 a premise statement is falsely classified as a Major Claim. To get deeper insights towards the behavior of the model, frequencies of words appeared in each predicted class were calculated. Figure 9a–d shows the visualization of word appearances.

Interestingly, a majority of 46% of statements that contain the word "should" in the testing dataset is predicted as a Major Claim. This is because the original dataset that is classified as a Major Claim has the word "should" as the 8th most frequent word in the dataset, while the words that appear more than "should" are articles and conjunctions like "the", "a", "and", "of" and more that have little meaning towards the class of the statement as shown in Fig. 9d. This can potentially show that the word "should" is biased towards the Major Claim class, hence it can be the reason why Table 4 exists.

The sentence in Fig. 9 contains the word "should", but the model can accurately classify the sentence as a Claim instead of a Major Claim. The BERT model is able to identify the context of the word "should" used in the statement. Even though this statement uses the word "should", but it does not emphasize any arguments that a major claim should has. The BERT model successfully identifies this problem.

Conclusion

In terms of classifying argument component, contextual language models outperform context-free models. This is shown by contextual models achieving a minimum F1 of 0.69, while context free models achieve an average F1 of 0.38 each. However, context free models are significantly faster than contextual models as all context free models can predict in less than a second. The advantage of the performance still far outweighs the cost. Context is proven to have a critical role in an argument. Thus, it is a factor that must be included in performing context sensitive tasks such as argumentation mining with the help of contextual language models.

Context is proven to be able to boost classification accuracy by approximately 20%. Unfortunately, the highest F1 score obtained in this research is only 71%. This shows that there is still a lot of room for improvement. As shown in Table 4, some contextual errors still exist. For future work, it is hypothesized that instead of considering only contextual individual words to classify the semantic label of the statement, contextual phrases can be considered which may have higher contribution towards the semantic label of the statement. For instance, instead of just using the context of individual words like "believe", phrases like "we strongly believe that" may have a deeper meaning towards the semantic label of the statement.

In this work, the models used for classification is just a vanilla network. Since this work is about enhancing the performance of classification for argument component using contextual model, then vanilla network is apparently enough. For the future works, the similar testing scenario like what mayers had done [16] can be applied by using argumentation data.

Abbreviations

BERT: Bidirectional Encoder from Transformer; CRF: Conditional Random Field; DistilBERT: Distilated BERT; ELMo: Embeddings from Language Models; GloVe: Global Vector; GLUE: General Language Understanding Evaluation; GRU: Gated Recurrent Units; LSTM: Long Short Term Memory; MultiNLI: Multi-Genre Natural Language Inference; RCT: Randomized Controlled Trial; SQuAD: Stanford Question Answering Dataset; Word2Vec: Word to Vector.

Acknowledgements

We would like to thank Universitas Indonesia for Grant namely TADOK (Tugas Akhir Doktor) 2018.

Authors' contributions
H contributed as the research principal in this work as well as the technical issues. ED contributed on technical issues. DS and AMA advise all process for this work. Regarding the manuscript, H, ED, DS and AMA wrote and revised the manuscript. All authors read and approved the final manuscript.

Authors' information
Hidayaturrahman is faculty member of Bina Nusantara University, Indonesia. He got his Master Degree in computer science from Institut Teknologi Bandung in 2018. His research fields are computer vision, natural language processing, and machine learning. Recently he is doing research in neural style stranfer. He also do some research on chatbot and agnostic modelling. Priorly, he was a data scientist that built credit scoring system and collection intelligent systems.

Emmanuel Dave is a third year student at Bina Nusantara University, Indonesia. His first research is forecasting Indonesian Exports with a hybrid model LSTM-ARIMA. He is passionate to conduct more research especially in the field of Computer Vision.

Derwin Suhartono is faculty member of Bina Nusantara University, Indonesia. He got his PhD degree in computer science from Universitas Indonesia in 2018. His research fields are natural language processing. Recently, he is continually doing research in argumentation mining and personality recognition. He actively involves in Indonesia Association of Computational Linguistics (INACL), a national scientific association in Indonesia. He has his professional memberships in ACM, INSTICC, and IACT. He also takes role as reviewer in several international conferences and journals.

Aniati Murni Arymurthy is professor in computer science with specialty in computer vision and image processing. She got her MSc from Computer and Information Sciences Department in The Ohio State University (OSU), Columbus, Ohio, USA. She got her PhD from Universitas Indonesia with sandwich program in Pattern Recognition and Image Processing Lab (PRIP Lab), Department of Computer Science, Michigan State University (MSU), East Lansing, Michigan, USA. Currently, she is active as lecturer in Faculty of Computer Science, Universitas Indonesia. Her research interests include pattern recognition, image processing, and spatial data.

Funding
All of this works is fully supported by Universitas Indonesia research Grant namely TADOK (Tugas Akhir Doktor) 2018.

Availability of data and materials
The datasets for this study are available on request to the corresponding author.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Author details
[1]Computer Science Department, School of Computer Science, Bina Nusantara University, Jakarta 11480, Indonesia. [2]Machine Learning and Computer Vision Laboratory, Faculty of Computer Science, Universitas Indonesia, Depok 16424, Indonesia.

Received: 15 April 2021 Accepted: 28 June 2021
Published online: 22 July 2021

References
1. Van Eemeren FH, Grootendorst R, Henkemans FS, Blair JA, Johnson RH, Krabbe ECW, Plantin C, Walton DN, Willard CA, et al. Fundamentals of argumentation theory: a handbook of historical backgrounds and contemporary developments. Mahwah: Lawrence Erlbaum Associates, Inc; 1996.
2. Stab C, Gurevych I. Annotating argument components and relations in persuasive essays. In: The 25th International Conference on Computational Linguistics, Ireland, Dublin, 2014.
3. Hollilan TA, Baaske KT. Arguments and arguing: the products and process of human decision making. Long Grove: Waveland Press Inc; 2015.
4. Wagemans J. Constructing a periodic table of arguments. In: Proceedings of the 11th international conference of the Ontario Society for the Study of Argumentation (OSSA), Windsor, 2016.
5. Suhartono D, Gema AP, Winton S, David T, Fanany MI, Arymurthy AM. Hierarchical attention network with XGBoost for recognizing insufficiently supported argument. In: 11th Multi-disciplinary international workshop on artificial intelligence, MIWAI 2017, Gadong, 2017.
6. Stab C, Gurevych I. Parsing argumentation structures in persuasive essays Christian stab, Iryna Gurevych. Comput Linguist. 2017;43(3):619–59.
7. Xue L, Lynch C. Incorporating task-specific features into deep models to classify argument components. EDM 2020.

8. Paul D, Opitz J, Becker M, Kobbe J, Hirst G, Frank A. Argumentative relation classification with background knowledge. In: Frontiers in artificial intelligence and applications, 2020, pp. 319–30.
9. Lenz M, Sahitaj P, Kallenberg S, Coors C, Dumani L, Schenkel R, Bergmann R. Towards an argument mining pipeline transforming texts to argument graphs. ArXiv 2020.
10. Eger S, Daxenberger, Gurevych I. Neural end-to-end learning for computational argumentation mining. In: Proceedings of the 55th annual meeting of the association for computational linguistics, Vol 1, Long Papers, Vancouver, Canada; 2017.
11. Pennington J, Socher R, Manning C. Glove: global vectors for word representation. In: EMNLP, 2014.
12. Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. In: ar Xiv:1301.3781, 2013.
13. Joulin A, Grave E, Bojanowski P, Mikolov T. Bag of tricks for efficient text classification; 2016.
14. Jacob Devlin KT. BERT: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, Vol 1, Long and Short Papers, Minneapolis, Minnesota; 2019.
15. Peters M, Neumann M, Iyyer M, Gardner M, Clark C, Lee K, Zettlemoyer L. Deep contextualized word representations. In: Proceedings of the 2018 conference of the North American chapter of the association for computational linguistics: human language technologies, Vol 1 (Long Papers), New Orleans, Louisiana; 2018.
16. Mayer T, Cabrio E, Villata S. Transformer-based argument mining for healthcare applications. In: 24th European conference on artificial intelligence—ECAI, Santiago de Compostela, Spain, 2020.
17. Gema AP, Winton S, David T, Suhartono D, Shodiq M, Gazali W, Shodiq M, Gazali W. It takes two to tango: modification of siamese long short term memory network with attention mechanism in recognizing memory network with attention mechanism in recognizing argumentative relations in persuasive essay argumentative relations in persuasive. In: 2nd international conference on computer science and computational intelligence, Bali; 2017.
18. Fromm M, Faerman E, Berrendorf M, Bhargava S, Qi R, Zhang Y, Dennert L, Selle S, Mao Y, Seidl T. Argument mining driven analysis of peer-reviews. In: https://arxiv.org/abs/2012.07743, 2020.
19. Schaefer R, Stede M. Annotation and detection of arguments in tweets. In: Proceedings of the 7th workshop on argument mining; 2020.
20. Bauwelinck N, Lefever E. Annotating topics, stance, argumentativeness and claims in Dutch Social Media comments: a pilot study. In: Proceedings of the 7th workshop on argument mining; 2020.
21. Toledo-Ronen O, Orbach M, Bilu Y, Spector A, Slonim N. Multilingual argument mining: datasets and analysis. In: EMNLP, 2020.
22. Sanh V, Debut L, Chaumond J, Wolf T. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. In: arXiv preprint ar Xiv:1910.01108; 2019.
23. Sun C, Qiu X, Xu Y, Huang X. How to fine-tune BERT for text classification. In: China National Conference on Chinese Computational Linguistics, 2019, October, p. 194–206.

Publisher's Note