BAS: An Answer Selection Method Using BERT Language Model

Jamshid Mozafari\textsuperscript{a}, Afsaneh Fatemi\textsuperscript{a,}\textdagger, Mohammad Ali Nematbakhsh\textsuperscript{a}

\textsuperscript{a}Faculty of Computer Engineering, University of Isfahan, Isfahan, Iran.

\textbf{Abstract}

In recent years, Question Answering systems have become more popular and widely used by users. Despite the increasing popularity of these systems, their performance is not even sufficient for textual data and requires further research. These systems consist of several parts that one of them is the Answer Selection component. This component detects the most relevant answer from a list of candidate answers. The methods presented in previous researches have attempted to provide an independent model to undertake the answer-selection task. An independent model cannot comprehend the syntactic and semantic features of questions and answers with a small training dataset. To fill this gap, language models can be employed in implementing the answer selection part. This action enables the model to have a better understanding of the language in order to understand questions and answers better than previous works. In this research, we will present the ‘BAS’ stands for BERT Answer Selection that uses the BERT language model to comprehend language. The empirical results of applying the model on the TrecQA Raw, TrecQA Clean, and WikiQA datasets demonstrate that using a robust language model such as BERT can enhance the performance. Using a more robust classifier also enhances the effect of the language model on the answer selection component. The results demonstrate that language comprehension is an essential requirement in natural language processing tasks such as answer selection.

1 Introduction

Humans have always sought to find answers to their questions. In the past, one found the answer of their questions by searching in the books. However, with the advent of the Internet and search engines, humans could utilize web resources and answer questions using search engines \cite{1}. A search engine receives the user’s question and returns the documents containing the answer to the question \cite{2}. One of the drawbacks of search engines is the need to go through the returned documents to find answers. To overcome this issue, question answering systems were developed. Instead of the whole document, these systems return a word, phrase, or sentence as the exact answer to questions.

Question answering systems are categorized into two categories: (1) Knowledge-based systems, (2) Information retrieval-based (IR-based) systems. Knowledge-based systems employ massive knowledge graphs as information resources. In these graphs, there are many entities - objects, events - as nodes connected by edges.
The edge captures the relationship between a pair of entities. To retrieve information from massive knowledge graphs, we have to use some query languages such as SPARQL. One of the drawbacks of knowledge-based systems is the need to upgrade and produce knowledge graphs because this is a time-consuming process. IR-based systems do not require knowledge graphs and attempt to extract the answer from unstructured documents. These systems use the machine reading comprehension task to find the exact answer to questions. IR-based question answering systems consist of four parts: (1) Question Analysis, (2) Document Retrieval, (3) Answer Selection, (4) Answer Extraction. Figure 1 shows the general pipeline architecture of the IR-based systems.

The question analysis part receives the user question and detects the answer type and also generates a query for the document retrieval part. The generated query is passed to the document retrieval part, and the most relevant documents to the query are retrieved. The answer selection part selects the most relevant from candidate sentences as the most relevant answer to the question. The answer selection part is divided into two general categories: (1) Extracted answer, (2) Generated answer. In the extracted answer category, the answer sentence is retrieved from the returned documents and is directly passed to the answer extraction part. In the generated answer category, the answer sentence is generated by processing the most relevant document’s sentences. Finally, the answer extraction part extracts the exact answer from the relevant answer sentence and returns it as the final exact answer. However, in some researches, the answer extraction part is combined with the answer selection part. Some researches even omit the answer extraction part and return the final answer as a sentence.

In this research, we focus on factoid questions; questions that can be answered with facts expressed in a few words. Factoid questions and their answer are semantically similar. According to this, we can consider the answer-selection task on factoid questions as a similarity measurement problem. Similarity measurement problem computes the similarity between a pair of sentences or paragraphs semantically and syntactically. This problem is considered as a supervised classification problem. In other words, we can train a classifier to predict the similarity between a pair of sentences. The answer-selection task can be expressed as: assume $q = q_1, q_2, \ldots, q_n$ is a set of questions, for each question $q_i$, there is a candidate set of answers $(s_{i1}, y_{i1}), (s_{i2}, y_{i2}), \ldots, (s_{im}, y_{im})$ where $s_{ij}$ refers to the $j_{th}$ candidate answer for $q_i$, $y_{ij}$ also refers to the correctness of the answer, as if $y_{ij} = 1$, the answer is correct and if $y_{ij} = 0$, the answer is incorrect. If a training dataset exists that includes such information, we can train a classifier that can find the most relevant answer to factoid questions semantically and syntactically.

An instance of a question with three candidate answers is shown in Table 1.

Until now, several methods have been proposed to undertake both answer-selection and similarity measurement tasks. These methods are divided into two general categories: (1) rule-based, (2) machine-learning. The rule-based methods measure the similarity between a pair of sentences based on linguistic rules, while the machine-learning methods use machine learning algorithms to learn linguistic rules automatically. In the machine-learning methods, feature engineering was initially utilized, but deep learning methods omit the need to feature engineering in recent years.

In this research, we demonstrate that using a robust language model such as BERT improves the performance of the answer selection task than the ELMo language model. We also indicate Expected Answer Types improve the performance of language models to find the most relevant answer. Finally, we illustrate that using some classifiers such as RNNs and CNNs has better performance than the BERT base classifier. We propose a model which uses the BERT model to find the most relevant answer to a factoid question from candidate answers. The empirical results demonstrate the superiority of our proposed model, which achieves state-of-the-art performance for TrecQA raw, TrecQA clean, and WikiQA datasets.

In this research, we endeavor to solve the following three research questions. Our experiments also aim to solve these questions:

- Can the BAS (BERT Answer Selection) model outperform the baseline models?
Table 1. A Factoid Question With Three Candidate Answers.

<table>
<thead>
<tr>
<th>Q</th>
<th>Who is the telephone inventor?</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a\textsubscript{1}</td>
<td>The first telephone was invented by Alexander Graham Bell.</td>
<td>( y_{11} = 1 )</td>
<td></td>
</tr>
<tr>
<td>a\textsubscript{2}</td>
<td>In 1875, Alexander Graham Bell succeeded in presenting the first telephone to human society.</td>
<td>( y_{12} = 1 )</td>
<td></td>
</tr>
<tr>
<td>a\textsubscript{3}</td>
<td>The first telephone was invented in 1875.</td>
<td>( y_{13} = 0 )</td>
<td></td>
</tr>
</tbody>
</table>

- Does the preprocessing have a significant effect on the performance of the BAS model?
- How do different classifiers affect the performance of the BAS model?

The contribution of this research paper includes:

- We propose the BAS model which ranks the candidate answers using language models, semantically and syntactically.
- The preprocessing part shows the importance of using EAT (Expected Answer Type) entities to find the most relevant answers.
- Some classifiers are utilized to capture the meaning of input sentences better than ordinary neural networks.
- The MAP and MRR measures of the BAS model show that it performs better than state-of-the-art models.

In Section 2, related works will be explained. In Section 3, the proposed model will be described in detail. In Section 4, the baseline models, datasets, and implementation details will be presented. In Section 5, the proposed model will be evaluated, and the results of the experiments will be discussed. Finally, the paper will be concluded in Section 6.

2 Related Works

This section consists of two parts: in the first part, we discuss the related works of the answer-selection task, and in the second part, the BERT language model will be briefly examined.

2.1 Answer Selection

The researches of the answer-selection task are divided into six parts: (1) Feature Engineering, (2) Siamese-based models, (3) Attention-based models, (4) Compare-Aggregate-based models, (5) Language Model-based models, and (6) Special models.

2.1.1 Feature Engineering

The feature engineering-based methods used the overlap between a question and its candidate answer sentence; that is, the most relevant answer was selected based on the common words between the question and candidate answer. These researches used bag-of-words and bag-of-grams methods. The question and the candidate answers presented in Table 1 indicate that using these methods is not sensible. The weakness of these methods was the lack of using semantic and linguistic features of questions and candidate answers [11]. In this regard, some studies used lexical resources such as WordNet [12] to overcome the semantic problem, but these researches failed to remove language constraints because some words were not mentioned in these lexical resources [13]. Some researches used syntactic and semantic structures of questions and candidate answers using dependency tree method [14]. Finally, Severyn et al. [15] presented a framework that performed feature engineering automatically and attempted to eliminate feature engineering problems to some extent. This framework can be considered one of the first attempts to eliminate feature engineering.

2.1.2 Siamese-Based Models

The proposed Siamese-based models follow the structure of the Siamese network [16]. In these models, the question and candidate answers are independently processed, and a vector representation for each of them is generated.

Yu et al. [17] presented the first model which used the deep neural network to overcome the answer-selection task. This model selects the most relevant answer from candidate answers using a convolutional neural network and logistic regression. Feng et al. [18] used the model presented by Yu et al. They employed various hidden layers, convolution operations, pooling, and activation functions. They also produce several independent models by combining convolution neural networks and fully connected networks. In this regard, He et al. [19] combined several models and produced a single model. The ranking method of previous models was a pointwise ranking, but Rao et al. [20] showed that using the pairwise ranking improves the model’s performance. In this research, they presented a framework that converted a pair of pointwise models into a pairwise model. In this regard, the model presented by He et al. [19] was given as a pointwise model to the Rao et al. model, which enhanced the performance of
the model. Madabushi et al. [17] provided a preprocessing operation rather than enhancing previous models. In their research, the named entities in the candidate answers, equivalent to the EAT, are replaced with a special token, making it easier for models to find the most relevant answer. This preprocessing was applied to the model presented by Rao et al. [20]. The problem was to replace all the tokens with a unique token. In this regard, Kamath et al. [21] presented a new model with recurrent neural networks. Instead of replacing all named entities with a unique token, they replaced each type of named entity with a special token.

2.1.3 Attention-Based Models

The attention-based models use context-sensitive interactions between the question and candidate answers. The attention mechanism is utilized in these models. The attention mechanism was first used in machine translation researches, but later in other fields of natural language processing such as question answering and the answer-selection task was also used [22].

Yang et al. [23] presented one of the first models which used the attention mechanism for the answer-selection task. In this research, the attention mechanism [22] was used to overcome the answer-selection task. The first attention mechanism was presented only for recurrent neural networks, but He et al. [24] could provide a model for answer-selection tasks which used the attention mechanism in convolutional neural networks. This research showed that combining the attention mechanism with convolutional neural networks is more efficient than the combination of the attention mechanism with recurrent neural networks. Mozafari et al. [11] showed that using feature vectors, convolutional neural networks, and pairwise ranking algorithms can provide a more robust model.

2.1.4 Compare-Aggregate-Based Models

The compare-aggregate models focus on context-interaction between smaller units such as words or tokens instead of sentences. These models compute interaction between smaller units of sentences such as words to capture more information. Then, they aggregate the information obtained from the computation between words and present a vector representation for the question and candidate answer [25].

He et al. [26] presented one of the first models to overcome the answer-selection task using compare-aggregate methods. They compared the word vectors of the question and candidate answer and produced the vector representations of each of the input sentences by aggregating the word vector’s values. Wang et al. [27] used the idea of He et al. [26] and presented a general compare-aggregate framework for the answer-selection task. Wang et al. [28] developed this framework and showed that if the question and candidate answer are matched in two directions, and instead of word-by-word matching, each word is matched with all the components of the other sentence, a more robust model is presented. Bian et al. [29] used the dynamic-clip technique rather than a simple attention mechanism in the compare-aggregate framework and showed that this modification eliminates ineffective information and provides a more robust vector representation. Shen et al. [30] introduced an inter-weight layer and tried to set a weight to each word. Tran et al. [31] introduced a new recurrent neural network which understood input text content more than previous models.

2.1.5 Language Model-Based Models

Instead of overcoming the answer-selection task from scratch, the language model-based models use pre-trained language models that understand languages semantically and syntactically. These models used the pre-trained language models to overcome the answer-selection task in a similar way proposed by Howard et al. [32].

Yoon et al. [33] developed a model which used language models for the answer-selection task. This model used the ELMo language model [33] along with techniques such as Latent-Clustering and demonstrated that the combination of these components produced a robust model. Laskar et al. [34] demonstrated that heavier language models produce a high-quality representation vector for the question and candidate answers. Nevertheless, Mozafari et al. [35] showed that heavier language models do not guarantee better performance for the answer selection task. Shonibare [36] demonstrated that using pairwise ranking and triplet ranking can be improved the performance of answer selection-task.

2.1.6 Special Models

Some models were not in line with earlier models and tried to provide an independent model. These researches create a new path in the answer selection field. However, they did not get much attention.

Shen et al. [37] developed the KABLSTM model, which utilizes knowledge graphs. They developed a context-knowledge interactive learning architecture, which used interactive information from input sentences and knowledge graphs. Yang et al. [38] presented the RE2 model which attempted to provide a lightweight model with satisfactory performance. The model’s name stands for Residual vectors, Embedding vectors, and Encoded vectors. Han et al. [39] also
showed that using candidate answer passages along with question and candidate answer sentences generates high-quality representation for input sentences. In Table 2 the related works for various datasets are shown.

2.2 BERT Language Model

In recent years, one of the issues that has achieved much attention is developing models that attempt to comprehend languages [33]. In these researches, they present some models which learn the syntactic and semantic rules of languages [8]. In other words, this model learns a language and can generate new texts with correct syntax and semantic rules. One of the novel language models which can overcome all other language models is the BERT model [8]. This model has taken advantage of Transformers [40], which is now widely used in the natural language processing community. The BERT model will be described in more detail below.

2.2.1 Transformer

One of the novel machine translation architectures is the encoder-decoder architecture [11]. This architecture has been used as one of the most widely used architectures in machine translations. Based on this architecture, the Transformer was introduced [40] in which a self-attention technique was used instead of using a recurrent neural network in the encoder and decoder. The method used by the Transformer went beyond machine translations and was employed in various natural language processing tasks. One of these tasks is language models such as the BERT language model, which uses the transformer encoder component to implement the language model. Figure 2 shows the transformer encoder architecture.

The first step of the Transformer encoder is Self Attention. In this step, three vectors are generated for each input vector X named Query, Key, and Value. The learned matrices \( W_{Qi} \in \mathbb{R}^{[X] \times |Q_i|} \), \( W_{Ki} \in \mathbb{R}^{[X] \times |K_i|} \), and \( W_{Vi} \in \mathbb{R}^{[X] \times |V_i|} \) are employed to produce Query, Key, and Value vectors for ith self-attention respectively:

\[
Q_i = X \times W_{Qi} \\
K_i = X \times W_{Ki} \\
V_i = X \times W_{Vi}
\]

The output vector of \( i \)th self-attention which is generated as follows:

\[
Z_i = \sigma ( Q_i \times K_i^T ) \times V_i / \sqrt{|K_i|} 
\]

According to Figure 2 there are \(|S|\) self-attention in the Transformer encoder. By concatenating the output vectors of self-attentions, the \( Z_{1..|S|} \) vector is generated. This vector is multiplied by the learned matrix \( W_O \in \mathbb{R}^{[Z_{1..|S|}] \times |H|} \) and \( Z \) vector is produced:

\[
Z = Z_{1..|S|} \times W_O 
\]

Finally, the \( Z \) vector is transferred to a fully connected layer and a new vector \( X_{new} \) is produced. \( W_F \in \mathbb{R}^{[H] \times [X]} \) is a matrix that is equivalent to the hidden layer parameters, and \( b_F \in \mathbb{R}^{[X]} \) is a vector that is equivalent to the bias:

\[
X_{new} = Z \times W_F + b_F
\]

2.2.2 BERT Model

The BERT language model consists of several transformer encoders stacked together. Two general types are defined based on the number of stacked encoders (L), the hidden layer size (H), and the number of self-attention heads (A). These two general types include the BERT-base and the BERT-large. The characteristics of these models are shown below:

- **BERT-base**: L:12, H: 768, A: 12, Training parameters: 110 M
- **BERT-large**: L:24, H: 1024, A: 16, Training parameters: 340 M

2.2.3 Fine-Tuning

Fine-tuning is to train models already trained for a particular task in order to be used for another task. We used fine-tuning, when we need to use captured knowledge achieved for a task for another task. As shown in Figure 3 fine-tuning has also been performed for different tasks in the BERT. These tasks include...
Table 2. Related Works According to Their Characteristics.

<table>
<thead>
<tr>
<th>Reference</th>
<th>Architecture</th>
<th>MAP TrecQA RAW</th>
<th>MRR TrecQA RAW</th>
<th>MAP TrecQA Clean</th>
<th>MRR TrecQA Clean</th>
<th>MAP WikiQA</th>
<th>MRR WikiQA</th>
</tr>
</thead>
<tbody>
<tr>
<td>[14]</td>
<td>Feature Engineering</td>
<td>0.631</td>
<td>0.748</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>[15]</td>
<td>Feature Engineering</td>
<td>0.678</td>
<td>0.736</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>[17]</td>
<td>Siamese</td>
<td>0.711</td>
<td>0.785</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>[18]</td>
<td>Siamese</td>
<td>0.711</td>
<td>0.800</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>[19]</td>
<td>Siamese</td>
<td>0.762</td>
<td>0.830</td>
<td>0.777</td>
<td>0.836</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>[20]</td>
<td>Siamese</td>
<td>0.780</td>
<td>0.8340</td>
<td>0.801</td>
<td>0.877</td>
<td>0.709</td>
<td>0.723</td>
</tr>
<tr>
<td>[7]</td>
<td>Siamese</td>
<td>0.836</td>
<td>0.862</td>
<td>0.864</td>
<td>0.903</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>[21]</td>
<td>Siamese</td>
<td>0.850</td>
<td>0.892</td>
<td>-</td>
<td>-</td>
<td>0.689</td>
<td>0.709</td>
</tr>
<tr>
<td>[23]</td>
<td>Attention</td>
<td>0.750</td>
<td>0.811</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>[11]</td>
<td>Attention</td>
<td>0.806</td>
<td>0.852</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>[26]</td>
<td>Compare Aggregate</td>
<td>0.758</td>
<td>0.821</td>
<td>-</td>
<td>-</td>
<td>0.709</td>
<td>0.723</td>
</tr>
<tr>
<td>[27]</td>
<td>Compare Aggregate</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.743</td>
<td>0.754</td>
</tr>
<tr>
<td>[28]</td>
<td>Compare Aggregate</td>
<td>-</td>
<td>-</td>
<td>0.801</td>
<td>0.877</td>
<td>0.743</td>
<td>0.755</td>
</tr>
<tr>
<td>[29]</td>
<td>Compare Aggregate</td>
<td>-</td>
<td>-</td>
<td>0.821</td>
<td>0.899</td>
<td>0.754</td>
<td>0.764</td>
</tr>
<tr>
<td>[30]</td>
<td>Compare Aggregate</td>
<td>-</td>
<td>-</td>
<td>0.822</td>
<td>0.889</td>
<td>0.733</td>
<td>0.750</td>
</tr>
<tr>
<td>[31]</td>
<td>Compare Aggregate</td>
<td>-</td>
<td>-</td>
<td>0.829</td>
<td>0.875</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>[6]</td>
<td>Language Model</td>
<td>-</td>
<td>-</td>
<td>0.868</td>
<td>0.928</td>
<td>0.764</td>
<td>0.784</td>
</tr>
<tr>
<td>[35]</td>
<td>Language Model</td>
<td>0.858</td>
<td>0.894</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>[34]</td>
<td>Language Model</td>
<td>0.891</td>
<td>0.925</td>
<td>0.888</td>
<td>0.953</td>
<td>0.829</td>
<td>0.843</td>
</tr>
<tr>
<td>[36]</td>
<td>Language Model</td>
<td>-</td>
<td>-</td>
<td>0.752</td>
<td>0.835</td>
<td>0.795</td>
<td>0.804</td>
</tr>
<tr>
<td>[37]</td>
<td>Special</td>
<td>0.792</td>
<td>0.844</td>
<td>0.803</td>
<td>0.884</td>
<td>0.732</td>
<td>0.749</td>
</tr>
<tr>
<td>[38]</td>
<td>Special</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.745</td>
<td>0.761</td>
</tr>
<tr>
<td>[39]</td>
<td>Special</td>
<td>0.824</td>
<td>0.863</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Sentence Pair Classification Tasks, Single Sentence Classification Tasks, Question Answering Tasks, and Single Sentence Tagging Tasks. This paper demonstrates that the BERT language model has a high comprehension of language [8].

3 Proposed Model Architecture

In this paper, we present the BERT Answer Selection (BAS) model which consists of three components: (1) preprocessing, (2) language model, and (3) classifier. The preprocessing component receives a question sentence and a candidate answer sentence as inputs. In the preprocessing component, the question is first given to the Expected Answer Type (EAT) Detector subcomponent. This subcomponent detects the answer type of the question and passes it to the Highlighter subcomponent. The Highlighter subcomponent highlights the words are equal to the detected EAT. The question and the replaced candidate answer are then passed to the language model component. This component uses the BERT language model [8]. The BERT model processes the inputs and generates a
vector representation for each token as output vectors. These output vector representations are passed as input to the classifier component to compute the relevance between the question and the replaced candidate answer. In this component, we employ some classification methods containing FCN classifier, BOW classifier, CNN classifier, and RNN classifier. We will explain each section in detail below. Figure 3 shows the BAS model architecture.

3.1 Preprocessing

The exact answer to a factoid question is a word that directly appeared in the answer sentence. For example, the answer to the question Who is the telephone inventor? is a sentence referring to Alexander Graham Bell. For example, the sentences The first telephone was invented by Alexander Graham Bell and In 1875, Alexander Graham Bell succeeded in presenting the first telephone to human society are both correct candidate answer sentences to this question. However, the exact answer is a person’s name. In other words, the exact answer to the question is Alexander Graham Bell. To better understanding, The first telephone was invented in 1875 is not a correct candidate answer sentence because Alexander Graham Bell is not mentioned in this sentence. More generally, the relevancy probability of a candidate answer containing named entities whose type is EAT, is more than other candidate answers 7. Earlier question answering systems process questions and candidate answers without any preprocessing. There is no guarantee that the system automatically detects the answer type in these systems and selects candidate answers containing EAT words. We process each candidate answer separately, and if the candidate answer includes EAT word, replace it with a special token. In this regard, the system learns that assigns more likelihood to the candidate answers containing the special token. We need two subcomponents to implement this component: (1) Expected Answer Type Detector, and (2) Highlighter. Each of these components will be explained below.

3.1.1 Expected Answer Type Detector

This subcomponent detects the answer type of questions. To perform this, we utilize the coarse-level of the EAT API output [7]. For example, In Who is the telephone inventor? sentence, the answer type of the question is (HUM, ind) that coarse-level answer (HUM) is kept and the fine-level answer (ind) is discarded.

3.1.2 Highlighter

This subcomponent replaces EAT words of candidate answers with a special token. To perform this, named entities of the candidate answers are detected using the Spacy NER tool. The detected named entities are then replaced with a special token (SPECIAL_TOKEN) if their type equals to EAT. The mapping between the named entity type detected by the Spacy NER tool and the output of the EAT detector is presented in Table 3.

The Table 3 describe preprocessing steps on Who is the telephone inventor? and The first telephone was invented by Alexander Graham Bell. sentences.

3.2 Language Model

The question and the processed candidate answer are passed to the language model component. In this component, the question and the candidate answer transforms into an appropriate template for the BERT model [8]. In this research, we use the BERT-base language model. The input of this model should be as follows:

\[ \text{BERT Input}(\text{Sentence}_1, \text{Sentence}_2) = \quad [\text{CLS}] \text{Sentence}_1 [\text{SEP}] \text{Sentence}_2 [\text{SEP}] \]

For example, for the question Who is the telephone inventor? and the candidate answer The first telephone was invented by Alexander Graham Bell., the input will be the following:
Figure 4. BERT Answer Selection Model Architecture. the Green Sections Are Trainable and the Orange Section Is Non-Trainable.

Table 3. Mapping Between the Named Entity Type and the Output of the Expected Answer Type Component [21].

<table>
<thead>
<tr>
<th>EAT</th>
<th>Spacy annotated tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>HUM</td>
<td>PERSON, ORG, NORP</td>
</tr>
<tr>
<td>LOC</td>
<td>LOC, GPE</td>
</tr>
<tr>
<td>ENTY</td>
<td>PRODUCT, EVENT, LANGUAGE, WORK OF ART, LAW, FAC</td>
</tr>
<tr>
<td>NUM</td>
<td>DATE, TIME, PERCENT, MONEY, QUANTITY, ORDINAL, CARDINAL</td>
</tr>
</tbody>
</table>

Table 4. An Example of the Preprocessing Component.

<table>
<thead>
<tr>
<th>Component</th>
<th>Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>EAT Detector</td>
<td>Who is telephone inventor? EAT Detector → (Who is telephone inventor?, HUM)</td>
</tr>
<tr>
<td>Highlighter</td>
<td>The first telephone was invented by Alexander Graham Bell. Spacy NER tool → The first [PRODUCT] was invented by [PERSON]. Replacing Spacy annotated tag with EAT, The first ENTY was invented by HUM. Replacing EAT detector output with special token, The first telephone was invented by SPECIAL_TOKEN.</td>
</tr>
</tbody>
</table>

[CLS] Who is the telephone inventor? [SEP] The first telephone was invented by SPECIAL_TOKEN [SEP]

The [CLS] token in the BERT model is utilized for classification tasks. In this research, we use the [CLS] token output and ignore the [SEP] tokens. By passing the question and the candidate answer to the language model, a new vector is generated for each input token, which captures the meaning of the token. In other words, the BERT model replaces the semantic vector of each token which independently captures the meaning of the token, with a vector that captures the meaning of the token according to its position in the sentence. The BERT model can be illustrated as follows:

\[
(E_{[CLS]}, E_1, \ldots, E_N, E_{[SEP]}, E'_1, \ldots, E'_N) = \text{BERT}([CLS], T_1, \ldots, T_N, [SEP], T'_1, \ldots, T'_N)
\]

3.3 Classifier

In this component, in addition to the BERT classification method, other methods will be implemented. Each of these methods will be explained below.
3.3.1 BERT-Base-Baseline (BB-Baseline)

In this method, the BERT classification method \cite{8} is employed. That is, the output of the [CLS] token, a vector of length 768, is passed as input to a fully connected neural network with a hidden layer of length 1024. The output layer of the fully connected neural network consists of two elements that the first indicates the correctness of the answer candidate, and the latter indicates the incorrectness of the answer candidate. Algorithm 1 presents the pseudo-code of this method. \(W_{h1} \in \mathbb{R}^{1024 \times 768}\) is a matrix that is equivalent to the hidden layer parameters, and \(b_{h1} \in \mathbb{R}^{1024}\) is a vector that is equivalent to the bias for the hidden layer. \(W_{h2} \in \mathbb{R}^{2 \times 1024}\) is a matrix that is equivalent to the output layer parameters, and \(b_{h2} \in \mathbb{R}^{2}\) is a vector that is equivalent to the bias for the output layer. Relu and Softmax activation functions are also used. Figure 11 illustrates the architecture of this method.

Algorithm 1 BB-Baseline Model Pseudo-Code.

\begin{enumerate}
\item \textbf{Require:} \(q, n\)
\item \(\text{Question} \Leftarrow \text{EAT} - \text{Detector}(q)\)
\item \(\text{Answer} \Leftarrow \text{Highlighter}(a)\)
\item \(\text{Input} \Leftarrow \text{BERT}\_\text{Input}(\text{Question}, \text{Answer})\)
\item \((E_{[CLS]}, E_1, \ldots, E_N, E_{[SEP]}; E'_1, \ldots, E'_N, E_{[SEP]}) \Leftarrow \text{BERT}(\text{Input}\_\text{TOKENS})\)
\item \(\text{CNN} \_1 \Leftarrow \text{relu}(W_{h1} E_{[CLS]} + b_{h1})\)
\item \(f(q, a) \Leftarrow \text{softmax}(W_{h2} \text{CNN} \_1 + b_{h2})\)
\item \(\text{return } f(q, a)\)
\end{enumerate}

In this method, only the output of the [CLS] token is used, and the other output vectors are ignored.

3.3.2 BERT-Base-BOW (BB-BOW)

In this method, in addition to the output vector of the [CLS] token, the output vectors of the question and candidate answer tokens are also utilized for classification. That is, the token vectors of each sentence are summed, and a new vector of length 768 is presented for each sentence. As a result, there will be three vectors of length 768: the output vectors of the question tokens, the candidate answer tokens, and the [CLS] token, respectively. A vector of length 2304 is produced by concatenating these vectors and is passed as input to a fully neural network connected with a hidden layer of length 1024. Algorithm 2 presents the pseudo-code of this method. In this method, \(W_{h1} \in \mathbb{R}^{1024 \times 2304}, b_{h1} \in \mathbb{R}^{1024}, W_{h2} \in \mathbb{R}^{2 \times 1024}, b_{h2} \in \mathbb{R}^{2}\). The Concat function concatenates the input vectors and produces a matrix. Figure 12 illustrates the architecture of this method.

Algorithm 2 BB-BOW Model Pseudo-Code.

\begin{enumerate}
\item \textbf{Require:} \(q, n\)
\item \(\text{Question} \Leftarrow \text{EAT} - \text{Detector}(q)\)
\item \(\text{Answer} \Leftarrow \text{Highlighter}(a)\)
\item \(\text{Input} \Leftarrow \text{BERT}\_\text{Input}(\text{Question}, \text{Answer})\)
\item \((E_{[CLS]}, E_1, \ldots, E_N, E_{[SEP]}; E'_1, \ldots, E'_N, E_{[SEP]}) \Leftarrow \text{BERT}(\text{Input}\_\text{TOKENS})\)
\item \(E\_1 \Leftarrow \text{concat}(E_1, \ldots, E_N)\)
\item \(E'\_1 \Leftarrow \text{concat}(E'_1, \ldots, E'_N)\)
\item \(E\_1 \_N' \Leftarrow \text{concat}(E\_1, \ldots, E_N)\)
\item \(\text{BOW} \_1 \Leftarrow \sum_{i=1}^{N} E\_1 \_i\)
\item \(E\_2 \Leftarrow \sum_{i=1}^{N} E'\_1 \_i\)
\item \(I\_L \Leftarrow \text{concat}(E_{[CLS]}, \text{BOW} \_1, \text{BOW} \_2)\)
\item \(I\_L \Leftarrow \text{relu}(W_{h1} I\_L + b_{h1})\)
\item \(f(q, a) \Leftarrow \text{softmax}(W_{h2} I\_L + b_{h2})\)
\item \(\text{return } f(q, a)\)
\end{enumerate}

window size is 3, and accordingly, the padding value is 2. The number of filters is 200, and MaxPooling is used for the pooling operation. By passing the sentence tokens to the CNN, A vector of length 200 is generated for each input sentence. These vectors are concatenated to the output vector of the [CLS] token, and a vector of 1168 lengths is produced. This vector is passed to a fully connected neural network whose hidden layer size is 1024. Then, a classification operation is performed. In this method, \(W_{h1} \in \mathbb{R}^{1024 \times 1168}, b_{h1} \in \mathbb{R}^{1024}, W_{h2} \in \mathbb{R}^{2 \times 1024}, b_{h2} \in \mathbb{R}^{2}\). Algorithm 3 is a pseudo-code of this method. In this pseudo-code, the CNN function refers to the convolutional neural network, and the MaxPool function also performs maximum pooling. Figure 13 illustrates the architecture of this method.

3.3.3 BERT-Base-CNN (BB-CNN)

This method also uses the output vector of the question and the candidate answer tokens. However, the convolutional neural network is used. In the CNN, the

3.3.4 BERT-Base-RNN (BB-RNN)

In this method, instead of using a CNN, we use an RNN. The network is a two stacked RNN whose hidden layer size is 768. For each input sentence, a vector of length 768 is produced, and a vector of length 2304 is produced by concatenating these vectors. This vector is passed to a fully connected neural network whose hidden layer size is 1024. Then, a classification opera-
tion is performed. In this method, $W_h \in \mathbb{R}^{1024 \times 2304}$, $b_h \in \mathbb{R}^{1024}$, $W_a \in \mathbb{R}^{2 \times 1024}$, $b_a \in \mathbb{R}^2$. Algorithm 4 presents a pseudo-code of this method. In this figure, the RNN function refers to the recurrent neural network. Figure 14 illustrates the architecture of this method.

### Algorithm 4 BB-RNN Model Pseudo-Code.

**Require:** $q, n$

1: Question $\leftarrow$ EAT - Detector($q$)
2: Answer $\leftarrow$ Highlighter($a$)
3: Input $\leftarrow$ BERT([Input, Question, Answer])
4: $(E_{CLS}, E_1, \ldots, E_N, E_{SEP}) \leftarrow$ BERT([Input.TOKENS])
5: $E_1N \leftarrow \text{concat}(E_1, \ldots, E_N)$
6: $E_1N' \leftarrow \text{concat}(E_1', \ldots, E_N')$
7: $RNN_i \leftarrow RNN(E_1N[i])$
8: $RNN_2 \leftarrow RNN(E_1N'[i])$
9: $H_L \leftarrow \text{concat}(E_{CLS}, RNN_1, RNN_2)$
10: $H_L \leftarrow \text{relu}(W_hH_L + b_h)$
11: $f(q, a) \leftarrow \text{softmax}(W_aH_L + b_a)$
12: return $f(q, a)$

### 3.4 Training Algorithm

To train the proposed model, the training parameters must be tuned so that the model enables us to find the best answer to user’s questions. The training algorithm is shown in Algorithm 5. The Q refers to the pool of questions, A refers to the pool of candidate answers, and L refers to the labels of the candidate answers. BAS Model refers to one of the proposed models, such as BB-Baseline, BB-BOW, BB-CNN, or BB-RNN. Cross_entropy is a loss function. Optimizer function also endeavors to tune the training parameters in order to minimize the loss value.

### Algorithm 5 Training Algorithm.

**Require:** $Q, A, L$

1: for epoch $\in \{1, 2, 3, 4\}$ do
2: for batch $\in \{Q, A, L\}$ do
3: for $(q, a, l) \in$ batch do
4: prediction $\leftarrow$ BAS_Model($q, a$)
5: loss$_\text{val} \leftarrow \text{Cross_entropy(prediction, l)}$
6: optimize($W_{BAS\_Model}, \text{loss}_\text{val}$)
7: end for
8: end for
9: end for

### 4 Experiments

In this section, we briefly describe the baseline models and compare their results with the proposed models. We then explain the datasets we evaluate the proposed models with. Finally, we provide evaluation metrics and implementation details.

#### 4.1 Baseline Models

To prove the superiority of the proposed models, the models should be compared with some competitive baseline models. In this regard, we compare the proposed models with the two competitive baseline models with the best results. The baseline models and our proposed models are summarized in Table 5.

#### 4.2 Dataset

We employ three datasets to evaluate the BAS model, including TrecQA Raw, TrecQA Clean, and WikiQA. Each of these datasets will be explained in more detail below.

##### 4.2.1 TrecQA Raw

The TrecQA Raw dataset is one of the most commonly used datasets in the answer-selection task built by Yao et al. [14] from Trec Question Answering Tracks. Trec Question Answering Track 8-12 data is used to produce training data, and Trec Question Answering Track 13 data is used for validation data and test data. In this dataset, training data consist of 1229 questions and 53417 pairs, evaluation data consist of 82 questions, and 1148 pairs, and test data consist of 100 questions and 1517 pairs.

##### 4.2.2 TrecQA Clean

The TrecQA Clean dataset is made from the TrecQA Raw dataset. In this dataset, questions with no correct answers or only one correct/incorrect answer are removed from the validation and test data. Training data such as TrecQA Raw consists of 1229 questions and 53417 pairs. However, the validation data and test data are different from the TrecQA Raw dataset. Validation data consist of 65 questions, 1117 pairs, and test data consisting of 68 questions and 1142 pairs.

##### 4.2.3 WikiQA

The WikiQA dataset consists of Bing search engine logs. Candidate answers to each question are extracted from Wikipedia pages. This dataset also eliminates questions that do not have the correct candidate answers. Training data consists of 873 questions and 8672 pairs, validation data consist of 126 questions and 1130 pairs, and test data consist of 243 questions and 2351 pairs.

The characteristics of these datasets are presented in Table 6.
Table 5. Summarization of Baseline Models and the Proposed Model.

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[21]</td>
<td>A Bi-LSTM model performs a preprocessing algorithm on the input sentences. In this preprocessing, the named entities which are equivalent to the answer type announced by the question processing part, are replaced with a special token.</td>
</tr>
<tr>
<td>[6]</td>
<td>A model which used language models for the answer-selection task. This model used the ELMo language model along with techniques such as Latent-Clustering and demonstrated that the combination of these components produced a robust model.</td>
</tr>
<tr>
<td>BB-BOW</td>
<td>A model which uses the output vector of the [CLS] token, the output vectors of questions, and answers tokens of the BERT language model to find the best answer. That is, the token vectors of each sentence are summed, and a new vector is presented for each sentence.</td>
</tr>
<tr>
<td>BB-CNN</td>
<td>A model which uses the output vector of the [CLS] token, the output vectors of questions and answers tokens of the BERT language model to find the best answer. That is, the token vectors of each sentence are transferred to a CNN, and a new vector is presented for each sentence.</td>
</tr>
<tr>
<td>BB-RNN</td>
<td>A model which uses the output vector of the [CLS] token, the output vectors of questions and answers tokens of the BERT language model to find the best answer. That is, the token vectors of each sentence are transferred to an RNN, and a new vector is presented for each sentence.</td>
</tr>
</tbody>
</table>

Table 6. Details of the TrecQA Raw, TrecQA Clean, and WikiQA Datasets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Set</th>
<th>Number of Questions</th>
<th>Number of Pairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>TrecQA Raw</td>
<td>Train</td>
<td>1229</td>
<td>53417</td>
</tr>
<tr>
<td></td>
<td>Validation</td>
<td>82</td>
<td>1148</td>
</tr>
<tr>
<td></td>
<td>Test</td>
<td>100</td>
<td>1517</td>
</tr>
<tr>
<td>TrecQA Clean</td>
<td>Train</td>
<td>1229</td>
<td>53417</td>
</tr>
<tr>
<td></td>
<td>Validation</td>
<td>65</td>
<td>1117</td>
</tr>
<tr>
<td></td>
<td>Test</td>
<td>68</td>
<td>1142</td>
</tr>
<tr>
<td>WikiQA</td>
<td>Train</td>
<td>873</td>
<td>8672</td>
</tr>
<tr>
<td></td>
<td>Validation</td>
<td>126</td>
<td>1130</td>
</tr>
<tr>
<td></td>
<td>Test</td>
<td>243</td>
<td>2351</td>
</tr>
</tbody>
</table>

4.3 Evaluation Metrics

MAP and MRR metrics are used in the answer-selection task to evaluate models. These metrics show the rating quality of candidate answers. The MRR metric only considers the rank of the first relevant answer, but the MAP measure considers the order of all relevant answers [2]. These metrics are shown below.

\[
MAP(Q) = \frac{1}{|Q|} \sum_{j=1}^{m_j} \frac{1}{m_j} \sum_{k=1}^{m_j} \text{Precision}(R_{jk}) \tag{10}
\]

\[
MRR(Q) = \frac{1}{|Q|} \sum_{j=1}^{m_j} r_j \tag{11}
\]

In these equations, \(Q\) is the set of questions, \(m_j\) is the number of relevant answers to \(q_j\), \(R_{jk}\) is a list of candidate answers that contains top \(k\) relevant answers, a precision function is a function that measures the ratio of the number of relevant answers to the total candidate answers, \(r_j\) is the inverse of the first relevant answer rank for \(q_j\).

4.4 Implementation Details

We implement the BAS model with PyTorch library in Python 3.6 programming language on the Colab platform. The model is trained on NVIDIA Tesla K80. We use BERT wordpiece tokenizer to tokenize input sentences. The batch size is equal to 32. The dropout is set to 0.2. Gelu function is used for activation function in the BERT language model and ReLU for fully connected layer activation function.
To train the proposed models, we set the learning rate to 0.0001. The model is trained for 4 epochs. AdamW optimizer and WarmupLinearSchedule scheduler [8] are utilized for training.

As shown in Figure 4, the Language model and the Classifier components are trainable and the Preprocessing component is non-trainable. The total number of training parameters of the Language model component is equal to 110M. In the BB-Baseline, the total number of training parameters of the fully connected layer is about $768 \times 1024 + 1024 \times 2 = 789k$. Hence the total number of training parameters in the BB-Baseline is $110000k + 789k = 110789k$. In the BB-BOW, the total number of training parameters of the fully connected layer is about $3 \times 768 \times 1024 + 1024 \times 2 = 2361k$. Hence the total number of training parameters in the BB-BOW is $110000k + 2361k = 112361k$. In the BB-CNN, the number of parameters in the convolution layer is about $2 \times 768 \times 2 = 3k$, and the total number of training parameters of the fully connected layer is about $(2 \times 200 + 768) \times 1024 + 1024 \times 2 = 1200k$. Hence the total number of training parameters in the BB-CNN is $110000k + 1200k + 3k \approx 111203k$. In the BB-RNN, the number of parameters in the recurrent layer is about $2 \times 768 = 1k$, and the total number of training parameters of the fully connected layer is about $3 \times 768 \times 1024 + 1024 \times 2 = 2361k$. Hence the total number of training parameters in the BB-RNN is $110000k + 2361k + 1k \approx 112362k$.

5 Results and Discussion

In this section, we explain the experimental results of the BAS model in detail. In other words, we respond to the research questions. In this regard, in Section 5.1 we answer whether the BAS model can outperform the baseline models. Section 5.2 answers whether the preprocessing has a significant effect on the performance. Section 5.3 answers how different classifiers affect the BAS model performance.

5.1 Model Performance

In this section, the results of the BB-BOW, BB-CNN, and BB-RNN models are compared with the baseline models.

The BAS model is compared with baselines in Table 4. The results show that the idea of using language models improves the performance of answer-selection models. The MAP and MRR metrics are increased for the TrecQA Raw, TrecQA Clean, and WikiQA datasets. This proves idea of using language modeling is significantly practical. For the TrecQA Raw dataset, the best results belong to the BB-RNN model. In this model, the MAP and MRR metrics are improved by 2.2% and 0.7%, respectively. For the TrecQA Clean dataset, the best results also belong to the BB-RNN. As shown in Table 7, the MAP and MRR metrics are improved by 4.3% and 3.1%, respectively. For the WikiQA data, the best results belong to the BB-BOW model. The results show that the MAP and MRR metrics are improved by 5.3% and 5.1%, respectively. These results are shown in Figure 5 and Figure 6.

These results demonstrate that using a robust language model such as the BERT language model has a significant impact on the performance of the answer-selection model. This shows that just using a language model is not enough, and each language model can lead to different results. The effect of preprocessing on the language model-based answer-selection model is still unclear. In the next section, we will examine the lack of preprocessing.

5.2 Lack of Preprocessing

In this section, the preprocessing component is removed from the BAS model and the results of the modified model are presented.

Table 8 shows the results of the alteration. As the results demonstrate, this alteration has a negative effect on model performance. The performance of the BB-BOW model has reduced by removing the preprocessing component. However, the performance of the BB-CNN model and the BB-RNN model have fewer changes than the BB-BOW model. The reason
Table 7. Evaluation of the Proposed Model.

<table>
<thead>
<tr>
<th>Architecture</th>
<th>MAP</th>
<th>MRR</th>
<th>MAP</th>
<th>MRR</th>
<th>MAP</th>
<th>MRR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TrecQA Raw</td>
<td>TrecQA Clean</td>
<td>TrecQA Raw</td>
<td>TrecQA Clean</td>
<td>WikiQA</td>
<td>WikiQA</td>
</tr>
<tr>
<td>[21]</td>
<td>0.850</td>
<td>0.892</td>
<td>-</td>
<td>-</td>
<td>0.689</td>
<td>0.709</td>
</tr>
<tr>
<td>[6]</td>
<td>-</td>
<td>-</td>
<td>0.868</td>
<td>0.928</td>
<td>0.764</td>
<td>0.784</td>
</tr>
<tr>
<td>BB-BOW</td>
<td>0.871</td>
<td>0.898</td>
<td>0.909</td>
<td>0.946</td>
<td>0.817</td>
<td>0.835</td>
</tr>
<tr>
<td>BB-CNN</td>
<td>0.863</td>
<td>0.893</td>
<td>0.909</td>
<td>0.938</td>
<td>0.790</td>
<td>0.805</td>
</tr>
<tr>
<td>BB-RNN</td>
<td>0.872</td>
<td>0.899</td>
<td>0.915</td>
<td>0.959</td>
<td>0.784</td>
<td>0.801</td>
</tr>
</tbody>
</table>

Table 8. Evaluation of the Effect of Lack of the Preprocessing (Pp) on the BAS Model.

<table>
<thead>
<tr>
<th>Architecture</th>
<th>MAP</th>
<th>MRR</th>
<th>MAP</th>
<th>MRR</th>
<th>MAP</th>
<th>MRR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TrecQA Raw</td>
<td>TrecQA Clean</td>
<td>TrecQA Raw</td>
<td>TrecQA Clean</td>
<td>WikiQA</td>
<td>WikiQA</td>
</tr>
<tr>
<td>BB-BOW(without PP)</td>
<td>0.862</td>
<td>0.880</td>
<td>0.889</td>
<td>0.924</td>
<td>0.793</td>
<td>0.819</td>
</tr>
<tr>
<td>BB-CNN(without PP)</td>
<td>0.861</td>
<td>0.890</td>
<td>0.903</td>
<td>0.938</td>
<td>0.792</td>
<td>0.810</td>
</tr>
<tr>
<td>BB-RNN(without PP)</td>
<td>0.868</td>
<td>0.894</td>
<td>0.912</td>
<td>0.948</td>
<td>0.786</td>
<td>0.803</td>
</tr>
</tbody>
</table>

Figure 7. Effect of Lack of the Preprocessing on Different Datasets in Terms of MAP.

Figure 8. Effect of Lack of the Preprocessing on Different Datasets in Terms of MRR.

that lack of the preprocessing has a low effect is the complexity of classifiers employed in these models. It has also reduced the BAS model performance for the TrecQA Raw and TrecQA Clean datasets. In contrast, the performance of the BB-CNN and BB-RNN models for the WikiQA dataset has improved slightly. The reason is the sparsity of training data in the WikiQA dataset because the model has not been able to properly learn the effect of the special token on finding the correct answer. These results are shown in Figure 7 and Figure 8.

These results show that the preprocessing has a positive impact on the performance of the language model-based answer-selection models. Given that its impact in some cases is low and even harmful, it is still useful. As mentioned, this may be due to the complexity of the classifiers. But this question needs to be considered separately and the impact of complex classifiers is examined.
to the user. The model consists of three components. The preprocessing component replaces the EAT tokens in the candidate answers with a special token. The language model component receives the question and the replaced candidate answer and generates a vector representation for each of them using the BERT language model. Finally, in the classifier component, the relevance matching is calculated using several classifiers. To evaluate our model, we performed several experiments. The experiments were performed on the TrecQA Raw, TrecQA Clean, and WikiQA datasets.

In the first experiment, the model performance was evaluated. In this experiment, the BAS model was compared with the two baseline models, and it was shown that the BB-BOW model was state-of-the-art for the TrecQA Raw and TrecQA Clean datasets. The BB-BOW model was also state-of-the-art for the WikiQA dataset. The experiment results showed that the language model comprehends the input sentences better than ordinary neural networks and produces robust representations.

In the second experiment, the effect of the preprocessing component was evaluated. In this experiment, the preprocessing component was removed and shown to significantly affect the BB-BOW model than the BB-CNN and BB-RNN models. The results of this experiment showed that the lack of preprocessing leads to a reduction in the model performance.

In the third experiment, the impact of classifiers such as BOW, CNN, and RNN was evaluated. In this experiment, the model’s performance with the typical BERT classifier was compared with mentioned classifiers in the paper, and it was shown that the BOW, CNN, and RNN classifiers performed better than the typical BERT classifiers. This experiment showed that using the output of all tokens instead of [CLS] token leads to a better comprehension of the input sentences.

In conclusion, we have shown that using strong language models eliminates the need to use knowledge bases and external resources. In other words, if a robust language model such as BERT is employed, the need for additional parts will be eliminated. The reason is the excellent comprehension of language models from languages, making it easier for the model to identify the relevant answers. The results prove the idea of using the language models. This idea can also be applied to other natural language processing tasks.

As future work, we would like to employ the language models derived from the BERT language model. Some models have been trained on more data and has provided a more efficient model than the BERT model, which better comprehends the

5.3 The Classifiers Impact

In this section, the results of BB-BOW, BB-CNN and BB-RNN models are compared with the BB-Baseline model.

The results are shown in Table 9. These results show that all the classifiers employed instead of the typical BERT classifier improve the performance of the answer-selection model. Using output vectors of all tokens instead of [CLS] token causes the BB-BOW, BB-CNN and BB-RNN models have better performance than the BB-Baseline model. BB-BOW uses the output vector of the other tokens as well as the output vector of the [CLS] token in order to capture more information about the input sentences. In addition to using the output of all tokens, BB-CNN uses the convolutional neural network to overcome the words order problem. In addition to preserving the order of words, BB-RNN also uses recurrent neural network memory to store sentences’ information. These results are shown in Figure 9 and Figure 10.

These results show that using different classifiers instead of the typical classifier of language models can improve the performance of answer-selection models.

6 Conclusions

In this research, we present the BAS model, which stands for BERT Answer Selection. This model aims to extract the answer of the user’s questions from the candidate answers pool and provide it as a final answer.
Table 9. Evaluation of the Impact of the Classifiers on the BAS Model.

<table>
<thead>
<tr>
<th>Architecture</th>
<th>MAP TrecQA RAW</th>
<th>MRR TrecQA RAW</th>
<th>MAP TrecQA Clean</th>
<th>MRR TrecQA Clean</th>
<th>MAP WikiQA</th>
<th>MRR WikiQA</th>
</tr>
</thead>
<tbody>
<tr>
<td>BB-Baseline</td>
<td>0.869</td>
<td>0.866</td>
<td>0.908</td>
<td>0.942</td>
<td>0.789</td>
<td>0.810</td>
</tr>
<tr>
<td>BB-BOW</td>
<td>0.871</td>
<td>0.898</td>
<td>0.909</td>
<td>0.946</td>
<td>0.817</td>
<td>0.835</td>
</tr>
<tr>
<td>BB-CNN</td>
<td>0.863</td>
<td>0.893</td>
<td>0.909</td>
<td>0.938</td>
<td>0.790</td>
<td>0.805</td>
</tr>
<tr>
<td>BB-RNN</td>
<td>0.872</td>
<td>0.899</td>
<td>0.915</td>
<td>0.959</td>
<td>0.784</td>
<td>0.801</td>
</tr>
</tbody>
</table>

language. Using these models, we can produce more robust representations of input sentences. Instead, some models offer a lite BERT model. The reduction of the training parameters in these models has resulted in a reduction in the performance of the BERT model. Instead, it allows us to employ more complex models alongside the language model. Also, we can use more powerful classifiers rather than the classifiers presented in this research.

References

[13] Z. Tu. An experimental analysis of multi-


[33] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner,


7 Appendix

This appendix includes models’ architecture presented for answer selection-task. In the following, we demonstrate the architectures of BB-Baseline, BB-BOW, BB-CNN, and BB-RNN models.
Jamshid Mozafari is a research assistant in Natural Language Processing at the BIGDATA lab of the University of Isfahan. He has received his B.Sc. and M.Sc. degrees in computer engineering from the University of Kurdistan and University of Isfahan in 2016 and 2019, respectively. His interests include Question Answering, Information Retrieval, and Machine Reading Comprehension. He is a member of the BIGDATA Lab at the University of Isfahan.

Afsaneh Fatemi received her B.Sc. from the Isfahan University of Technology, Isfahan, Iran, in 1995, and her M.Sc. and Ph.D. from the University of Isfahan, Isfahan, Iran in 2001 and 2012, all in Computer Engineering (Software). Dr. Fatemi joined the Faculty of Computer Engineering at the University of Isfahan in 2013, as an assistant professor. Her main research interests lie in the fields of Complex Networks, Multi-Agent Systems, and Emergency Response Systems’ Modeling. She as been collaborating with the MDSE research group since 2014.

Mohammad Ali Nematbakhsh is a full professor in the software engineering department at the University of Isfahan. He received his BSc in electrical engineering from Louisiana Tech University, USA, in 1981, and his MSc and Ph.D. degrees in electrical and computer engineering from the University of Arizona, USA, in 1983 and 1987, respectively. He worked for Micro Advanced Co. and Toshiba Corporation for many years before joining the University of Isfahan. He has published more than 150 peer-reviewed research papers, several U.S. registered patents, and two database books that are widely used in universities. His main research interests include intelligent Web and big data technology.