Neural Computing and Applications (2020) 32:2909–2928
https://doi.org/10.1007/s00521-020-04725-w

ORIGINAL ARTICLE

A deep learning analysis on question classification task using Word2vec representations

Seyhmus Yilmaz[1] · Sinan Toklu[1]

Received: 21 February 2019 / Accepted: 7 January 2020 / Published online: 21 January 2020

Abstract
Question classification is a primary essential study for automatic question answering implementations. Linguistic features take a significant role to develop an accurate question classifier. Recently, deep learning systems have achieved remarkable success in various text-mining problems such as sentiment analysis, document classification, spam filtering, document summarization, and web mining. In this study, we explain our study on investigating some deep learning architectures for a question classification task in a highly inflectional language Turkish that is an agglutinative language where word structure is produced by adding suffixes (morphemes) to root word. As a non-Indo-European language, languages like Turkish have some unique features, which make it challenging for natural language processing. For instance, Turkish has no grammatical gender and noun classes. In this study, user questions in Turkish are used to train and test the deep learning architectures. In addition to this, the details of the deep learning architectures are compared in terms of test and 10-cross fold validation accuracy. We use two major deep learning models in our paper: long short-term memory (LSTM), Convolutional Neural Networks (CNN), and we also implemented the combination of CNN-LSTM, CNN-SVM structures and a number of various those architectures by changing vector sizes and the embedding types. As well as this, we have built word embeddings using the Word2vec method with a CBOW and skip gram models with different vector sizes on a large corpus composed of user questions. Our another investigation is the effect of using different Word2vec pre-trained word embeddings on these deep learning architectures. Experiment results show that the use of different Word2vec models has a significant impact on the accuracy rate on different deep learning models. Additionally, there is no Turkish question dataset labeled and so another contribution in this study is that we introduce new Turkish question dataset which is translated from UIUC English question dataset. By using these techniques, we have reached an accuracy of 94% on the question dataset.

Keywords Deep learning · Question classification · SVM · Word embedding · Word2vec

1 Introduction

There is an increasingly growing amount of data on the Internet such as the size and variety of online text documents. It results in the consumers inconvenient with the answers returned by programs, which just provide ranked lists of texts that individuals have to consume time manually browsing through. In most cases on the natural language process, what a customer desires is the accurate answers to the questions asked by individuals. The aim of a question answering (QA) implementation is to reply straightforwardly natural language queries asked by people. QA systems are popular study area that uses NLP with information retrieval.

On the other hand, the area of question classification or question categorization is to identify the category sort of questions in question answering (QA) implementation, which is asked in NLP.A significant part in QA (question answering) systems [1–3] or other dialog implementations is to identify the questions to the probable category of an answer [4]. For instance, the question of What country is famous for chocolate ought to be classified into the kind of

✉ Seyhmus Yilmaz
seyhmusyilmaz@duzce.edu.tr

Sinan Toklu
sinantoklu@duzce.edu.tr

[1] Department of Computer Engineering, Faculty of Engineering, Düzce University, Konuralp Campus, 81620 Düzce, Turkey

location (country). Such data narrows down the search space to classify the accurate answer string. Additionally, such data may propose different methods to investigate and confirm a probable answer. For instance, the classification of the question "Who is the prime minister of Belgium" to a type of "human (person)" question should use the search method specific for human (person) type.

Document categorization is one of the related issues to question categorization [5]. Even though document categorization has been given a large amount of scientific concentration in recent times, question categorization particularly in Turkish language now is a novel academic problem. While a number of academic publications have attempted to classify documents for Turkish language, to the best of our knowledge, we have not come across any important academic papers related to question categorization in Turkish. The most significant distinction between document categorization and question categorization is that the document length is much longer than a question length thus each word and character in questions could be useful. As a result of this, it is harder to take features from a single question than a large document [5]. The most important difference between classifying questions and classifying documents is that the length of a question is much smaller than a document. Therefore, in a question, each word and character could be important and it may be harder to subtract features from a one document.

In question classification tasks, there are three techniques: machine learning, rule-based and hybrid techniques [6, 7].In our study, some deep learning techniques such as deep learning (CNN [8], LSTM [9]), SVM [10] and their combinations are used to classify user questions based on Word2vec methods both skip gram and Continues Bag of Grams. Our major contributions in this study are: Most papers related to question classification focus on English language and they have not studied an agglutinative language where the structure of words is generated by putting suffixes (morphemes) to the words root. Turkish language has been proven challenging for NLP. Most of the difficulties stem from the complicated morphology of Turkish and how morphology interacts with syntax [11]. As a non-Indo-European language, languages like Turkish have some distinctive features, which make it difficult for natural language processing. For instance, Turkish language does not have grammatical gender and noun classes [12]. In order to differentiate various age, courtesy or familiarity toward addressee, social distance, levels of politeness, this language is based upon second person pronouns. In general, it is hard to capture these nuances by natural language processing methods, which have been commonly used for Indo European languages like German and English. The real meaning of a noun can be changed by the extensive usage of affixes [12]. Although a number of studies about text classification for Turkish language based on Word2vec methods have been done [13], in our knowledge, we have not come across any important study related to text classification in Turkish based on deep learning using Word2vec skip gram or continues bag of grams.

For linguistic and grammatical reasons, natural languages have numerous words, which derive from the same morphological class. Especially, in Turkish language, there is a huge amount of derived words because of the language structure [14]. Turkish utilizes the derivative affixes and the inflectional to derive new words in which they naturally might contain a number of hundred structures, and a number of million structures can be created from every verbalroot [15].

For that reason, it is very likely to derive words that mean nearly a sentence in English language. Numerous derivational and inflectional suffixes can be taken from Turkish words when used in context in a sentence for example;

gör + ebil + iyor + du → → (s)he was able to see (it)
okul + da + idi + ler → → they were in the school.
gönder + ebil + ecek + se + n→ → If you will be able to send (it)

For that reason, in most cases lemmatization procedure is very important for gaining the uninflected word forms to apply IR (information retrieval) and other document process technique to struggle Turkish. Lemmatization process generally is used to improve the performance of the systems in information retrieval. Decreasing inflectional forms and occasionally derivationally connected form of words to root forms are the main aim of lemmatization. It provides a link among surface form of connected words and dictionary forms. In Turkish, due to the language structure of Turkish for above-mentioned reasons, there is no an effective lemmatization tool in comparison with English language. This is another difficulty with Turkish language when studying text processing (for more details about the difficulty of Turkish see [16]).

Another problem with Turkish is that it can lack resources needed when deciding to study on textual data. At the beginning of this study, we were unable to find any Turkish question dataset and then we decided to translate an English question dataset to Turkish in order to assess the performance of the proposed methods in its best way. After this study, we will plan to share this Turkish question dataset for researchers who wish to study in this area.

Previously, bag of words was a modern technique to extract features from questions [6] but there are some big disadvantages in this technique. Bag of word techniques cannot capture semantics of the word. For instance: a training question containing words 'car' and 'automobile' are frequently used in the same context [17]. As a result,

misclassification would be possible. On the other hand, the vectors related to these words are orthogonal in bag of words system. This makes our problem more serious while making a model for questions. For instance, "Powerful" and "strong" and "Paris" are equal distant in BoW [17].

In order to tackle this challenge, we require one mathematical representation of words. In this mathematical representation, we assigned every word $\boldsymbol{x}$ to a vector $\boldsymbol{f}(\boldsymbol{x})$ such that if $\boldsymbol{y}$ and $\boldsymbol{x}$ have syntactic and semantic similarity then $\boldsymbol{f}(\boldsymbol{y})$ and $\boldsymbol{f}(\boldsymbol{x})$ will become nearby vectors. A new representation of words with above-mentioned feature is invented by Mikolov et al. called distributed representations of words or Word2vec [18], which is used in our study in feature extraction step. The idea behind in word representation technique is that words which have semantic or syntactic relation are seen in the same contexts with high likelihood [19]. Consequently, if word1 and word2 occur in the same context, the vectors of these two words ought to be a little bit nearer to each other.

In Word2vec, the representation of the words in a vector space aids learning algorithms to accomplish better results in NLP systems by classifying related words. Distributed representations of words calculated utilizing neural networks are very remarkable since the computed vectors obviously code numerous linguistic regularities and patterns [18]. Somewhat unexpectedly, most of such patterns can be shown as linear translations. For instance, the outcome of a vector calculation vec("king")—vec("man") + vec("women") is closer to vec("queen") than to any other word vectors [18].

Our main contributions are as follows:

(1) Most papers related to question classification focus on English language and they have not studied an agglutinative language where the structure of words is generated by putting suffixes (morphemes) to the words root. Turkish language has been proven challenging for NLP. As a non-Indo-European language, languages like Turkish have some distinctive features, which make it difficult for natural language processing.
(2) Another contribution the effect of using different Word2vec pre-trained word embeddings on different deep learning architectures. First technique shown in our study utilizes Word2vec algorithms that are Continues Bag of Grams and skip gram to cluster words in the corpus and transform all words into vectors in the space. The other technique uses every feature vector as a combination of vectors of the words of the questions. To extract word vectors, the Word2vec technique is used. After that, CNN, LSTM, CNN-LSTM and CNN-SVM combinations are used for classification. By applying those four different techniques, an average correctness of 92.77% using CNN, 90.86% using LSTM, 91.8% using CNN-LSTM and 92.07% using CNN-SVM over the Turkish question dataset was achieved.
(3) As well as this, there was no Turkish question dataset labeled and so another contribution in this study is that we introduce a new Turkish question dataset which is translated from UIUC English question dataset [20].

1.1 Deep learning for text mining

Recently, the employ of a novel method known as deep learning has attracted the attention of developers and researchers, since deep learning algorithms have acquired extraordinary performance for a variety of natural language processing applications. Deep learning algorithms are the type of machine learning algorithms consisting of various layers of perceptron which model the human being brain [21]. In other words, it is an architecture that includes numerous layers of nonlinear data processing and a technique to learn the representation of features at consecutive layers. Varied deep learning architectures have been used in different tasks, for example, long short-term memory (LSTM), deep neural networks (DNN), deep restricted Boltzmann machine (RBM), convolution neural networks (CNN), etc. In addition, deep learning architecture has obtained remarkable results in natural language processing for various problems such as sentimental analysis and text classification. From now on, we will illustrate some of the related works that use deep learning.

In [22], the authors use long short-term memory (LSTM) to predict the sentiment of Roman Urdu language tweets. In addition to LSTM model, they also use other classifiers such as random forest and Naive Bayes. Their LSTM deep learning model with word embedding outperforms other models on Sentiment Analysis in Roman Urdu language. The authors of [23] introduce the method to identify fuzziness in law textual data with a deep neural network model in Thai language. In their paper, the definition of the fuzziness is an imprecise meaning in law text documents, which can be vague when read by a system. They generated a labeled corpus from four Thai Law codes specifically (1) The Civil Procedure Code, (2) Commercial Code and The Civil, (3) The Criminal Code and The Criminal Procedure Code. To classify the fuzziness, three situations are produced. First is a verdict that needs the production of evidence and second is a verdict depending upon a judge's view and last one is a verdict, which shows other units. In addition to deep neural network model, the authors use some machine learning algorithms such as SVM (support vector machine, random forest, decision

tree). According to the experimental results, deep neural network considerably has superior performance with accuracy of 97.54% on all the dataset. In [24], the authors amalgamate sentiment prediction and morphological evaluation in Punjabi language, which is an Indian language, using deep learning. To accomplish this, they use sentimental analysis of farmer suicide cases written in Punjabi language on Punjabi online sites. 275 suicide cases are taken in Punjab. For classification, they use Deep Neural Network and Morphological Punjabi text classification. They achieve the accuracy of 95.45%.

An ensemble architecture of shallow and deep neural network methods are applied for Vietnamese sentimental analysis in [25]. In this paper, the authors made an experiment of three different dataset. In [26], the authors built a chatbot using the seq 2seq model integrating a deep learning architecture of attention mechanism in Vietnamese language but they use the limited dataset for Vietnamese language. This chatbot are able to response to the users, but messages produced by the chatbot should be enhanced to obtain a meaningful dialogue by enlarging the dataset. The authors of [27] use a method based upon deep learning algorithms to categorize a document as composed by a specific writer in Russian language. In addition to this, they make a comparison between dissimilar techniques of vector representations of the textual data, for example, character n-grams, label encoding. They test the experiments on some corpora in Russian, which gathered some common social media websites. Lastly, they compare different deep learning algorithms like long short-term memory (LSTM), CNN (Convolutional Neural Network), RNN (recurrent neural network) and the combination some architecture. In [28], a novel framework for managing negative messages on social media websites has been introduced. In order to get superior classification and training performance on social-based mentions, the authors made considerable alteration in the combination method of deep learning architecture layers. In addition to this, for retraining the pre-trained embedding word vectors for better reflect sentimental terms; they introduced the resultant sentimentally embedded word vectors. The authors of [29] employ an ensemble architecture which combines long short-term memory (LSTM) and Convolutional Neural Network (CNN) architectures in order to perform the sentiment of Arabic tweet analysis. In this model, they did not utilize any feature extraction technique and any complicated methods to obtain particular features. They only use a pre-trained word vector representation, which is pre-trained on Twitter corpus.

2 Related works

In classic question answering systems, there are three separate steps [30]:

1. Question processing: The purpose of this step is to understand questions asked by users [31], for which logical operations are applied for the representation and classification of the questions. In other words, this step classifies the questions asked by users and this step also is called as question classification step.
2. Document extraction and processing: This step selects a set of related documents and extracts a set of paragraphs, which depends upon the focus of the question. The answer is in terms of these paragraphs.
3. Answer processing: This step is aim to select the response based upon the related fragments of the documents. This requires a pre-processing of the data so as to pair an answer with the question asked. In Fig. 1, the common architecture for a question answering system is shown [32].

In order to resolve the question classification problem, there are numerous various techniques. Many of these techniques can be separated into three groups: rule-based, machine learning techniques, and hybrid techniques [6, 7]. In order to classify questions in rule-based techniques [33], the system tries to pair questions with manually written rules. But deciding the precise rules is spending enormous time and effort to understand a variety question types. For highly inflectional languages such as Turkish, it is very difficult to find all probable type of questions. The main cause why this kind of classification methods is uncommon is that the general accuracy is not even close to the approaches that uses machine learning [6]. In [34], authors discuss that machine learning approaches are better than manual methods. In contrast to manual approaches, machine learning approaches provide a reasonably easier way to categorize questions. In this way, it can be learnt from the data, and therefore, this type of implementations easily can be tailored to a new system.

Finally, the hybrid approaches are novel and not widespread; there are limited papers which are using this technique. We will mention a little bit about some studies. The authors of the [5] have proposed a hybrid method for a Persian closed domain question classification system. The authors have formed taxonomy by themselves and prepared a database that include 9500 questions with the assist of a few scholars. They achieve the satisfactory performance according to high number of question classes with 80.5%.

Despite rule-based methods, machine learning methods are able to automatically build an accurate classification implementation using various features of questions [5]. In

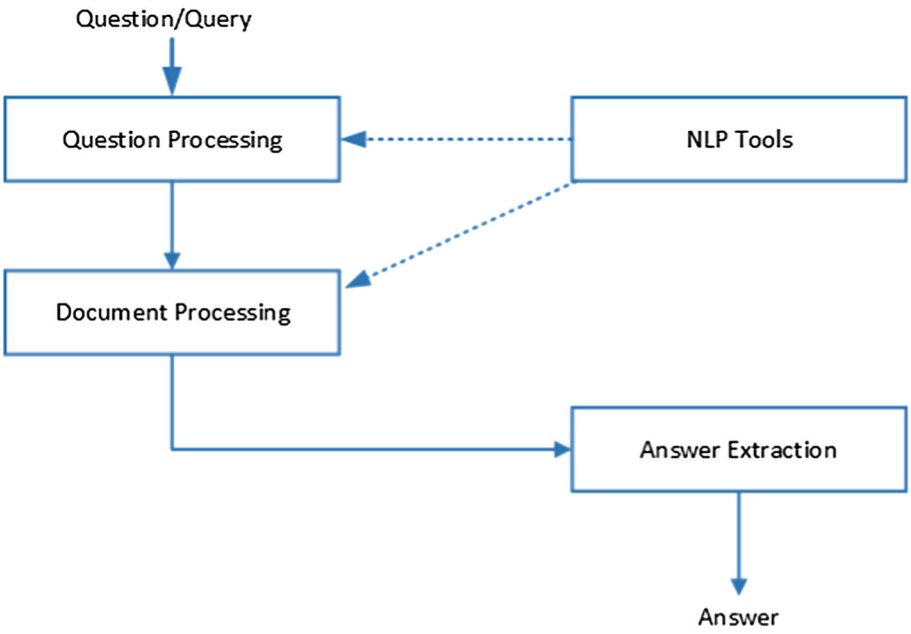

Fig. 1 The general architecture of NLQA system

addition, there are various machine learning approaches, for example, SVM, Naive Base, decision trees, K-nearest neighbors and deep learning classifiers that are utilized for question classification. However, SVM (Support Vector Machine) is the main machine learning approach utilized for question classification [30]. The authors of [30] are employed Support Vector Machine and dimension reduction with bag of n-grams feature vector. In order to achieve their goals, the authors select to employ as few linguistic features as possible.

In some machine learning approaches, questions are reformulated as a tree. In [35], the authors used a tree kernel with a SVM to classify questions and succeeded 80.2% statistics. But they did not use semantic and syntactic features during their experiments. Moreover, the authors of the [36] implemented a kernel function called Hierarchical Directed Acyclic Graph (HDAG) that straightforwardly accepts structured natural language information, for example, some levels of chunks and their relatives. The authors of [37] proposed a hierarchical method using the SNoW learning architecture to classify questions. In that study, a two-phase classification process is employed. In the initial phase, the five most likely coarse-grained question categories are shown. In the next phase, the question is categorized into one of the child categories of the five coarse-grained question categories with result of 84.2%.

In [4], the authors proposed two approaches to gain augment semantic features of defined headwords based upon WordNet. By using Maximum Entropy (ME) and linear Support Vector Machine (SVM) and methods, they reach the accuracy of 89.2% and 89.0%, respectively. Mollaei in [38] classify sentences into two levels of coarse-grained and fine-grained categories based upon the category of the answer to every question. Then, they extract setting and features sliding window on the Conditional Random Fields architecture and trained CRF question classifier. The aim of this paper is to categorize Persian questions, and they gain the dataset employed for the study by their own effort. Most of the question dataset are obtained from the junior high and primary school documents, and the rest is taken from commonly posed questions in some Internet sites. The satisfactory statistics has been achieved according to a large number of question categories with an accuracy of 79.8%. Mishra et al. combine semantic, syntactic and lexical features which get better result of classification. Additionally, they adopt three diverse methods: (NN) nearest neighbors, SVM (support vector machines) and NB(Naive Bayes) based on bag of n-grams and/or bag of words. The authors also show that when taking SVM classifier and combining the semantic, syntactic and lexical feature, this recovers the results of classification.

Loniin [39] uses features roughly the same as what the authors of [4] introduced. Even though the improvement is that the authors employed a dimension reduction method close to PCA (principal component analysis) called LSA (latent semantic analysis) to decrease the space of feature dimension to a much smaller one. In their study, BPNN (back-propagation neural networks) and support vector machines are utilized. Their statistics show that back-propagation neural networks get better results than SVM. Some clustering algorithms are used by Razzaghnoori et al. [6] in order to cluster words in the vocabulary and then

Table 1 Outline of related studies

Language	Dataset	Feature extraction method	Classification technique	Accuracy (%)	References
English	UIUC question dataset	Bag of n-grams	SVM	87.4	[35]
English	UIUC question dataset	Bag of n-grams	DT	84.2	[35]
English	UIUC question dataset	Bag of n-grams	NN	79.8	[35]
English	UIUC question dataset	Bag of n-grams	NB	83.2	[35]
English	UIUC question dataset	To generate more complicated features, named entities, chunks, head chunks, Words, POS tags, semantically associated words and over these basic features some operators are used	SNoW	91	[34]
English	Penn treebank using an additional treebank with 1153 words	Lexical feature, POS tags	–		[33]
English	UIUC question dataset	*N*-grams, named entities	SVM	82.0	[30]
English	UIUC question dataset	*N*-grams	SVM	80.2	[30]
English	NTCIR-QAC1	Semantic information, named entities, words	SVM using HDAG kernel	88.0	[36]
English	UIUC question dataset	WordNet for quoted and target strings, Lexical and syntactic info: chunks, named entity tags, language model, POS tags	ME (maximum entropy model)	86.0	[40]
English	UIUC question dataset	Words, named entity, semantic information	–	91	[41]
English	UIUC question dataset	WordNet semantic features for n-grams, word shape, headword, wh-words, headword	SVM and ME	89	[4]
Persian	Almost junior high school and primary	POS tags, *N*-gram, position of tokens, Question informer, words, Question Words	CRF	85.3	[38]
Persian	QURANIC Question	*N*-gram, Verse Finder, special word detection, POS tags, Lemma, length of question, normalized word	SVM with defined rules	75.9	[5]
English	UIUC question dataset	Headwords, word-shapes, related words, hypernyms, Bigrams, wh-words		93.8	[39]
Persian	UTQD.2016	Word2vec, tf-idf	MLP, SVM, LSTM, RNN	85	[6]

they converted every question into a vector space. After that for classification, MLP and SVM are used. By performing such techniques, they achieve an average accuracy of 72% by Support Vector Machine and an accuracy of 72.46% by Multi-Layered Perception on 3 different databases. Additionally, they prepare UTQD-2016 (University of Tehran Question Dataset 2016). Questions in this dataset are taken from several type of jeopardy game shown by the official Iran's TV. In their third technique, each question is converted to a matrix where every row shows Word2vec representation of a word. After they use a LSTM [6] model to classify the questions and they also achieve an average accuracy of 81.77% on three question databases. Table 1 illustrates the outline of the related studies in question classification.

In addition, MT (machine translation), which translates the data in source language into target language or vice versa [42], is one of the other ways to accomplish question answering tasks for especially under-sourced languages [43]. The authors of [44] proposed some baseline frameworks for cross-lingual OpenQA with two machine translation-oriented approaches that make the translation of training data and test data, respectively. In the translation test environment, they used Google Translate to translate from Ukrainian, Polish and Tamil to English and they make use of their own translator for French, Portuguese, German, Chinese and Russian. In [45], a novel method called XLDA (cross-lingual data augmentation) is introduced to improve the performance of natural languages processing systems including question answering (QA). In this method, the authors replace a piece of the NLP input text with its translation in another language. As a result, they make improvements to all languages in the XNLI dataset by up to 4.8%. The system achieved the state-of-the-art performance for three languages including the low-resource languages such as Urdu. The authors of [46]

proposed a version in Indonesian question analyzer to improve the English monolingual question answering architecture into Indonesian-English Cross Language question answering. Better results are revealed by using a question analyzer on the sourced language side than utilizing the question translation on the target language side. It ought to be observed that the question analyzer is similarly appropriate for monolingual question answering in Indonesian question answering. The authors of [47] examine the relation between automatic and manual translation evaluation metrics. To do this, they begin with a standard QA dataset and generated manual and automatic translations. A question set with five different varieties of translation results are generated by the authors. Firstly, they translate the questions into Japanese manually and then generated translations of the Japanese dataset into English by five different approaches. For machine translation, they use Google, Yahoo, Moses and Travator Translate. In [48], the authors generate a question–answer pairs with the questions and answers both being in Hindi and English. For classifying an input question into the categories depending on the expected answer, they built a deep neural architecture. While testing, they translated Hindi answers to English and produced a gold answer set by merging the real English answer and the translated English answer for each question. Their method reached an accuracy of 80.30% and 90.12% for finer and coarse categories, respectively. In [49], the authors concentrate on Questing Answering systems and give an outline of the current state of the area for cross-lingual (CLQA) and multilingual (MLQA) sub-tasks. Furthermore, they include an initial effort to analyze the result of a basic deep learning architecture in several language setting. The authors of the [50] generate a feature for every combination of translation direction and technique, and train an architecture that learns optimum feature weights. On a big forum dataset containing of messages in Chinese, Arabic and English, their new learn-to-translate method is better than a robust baseline, which translates all text into English and then trains a classifier based only upon English (translated or original) text.

3 Methodology and results

In this part, the feature extraction techniques will be explained in depth. These feature extraction techniques are very important to classify the nature of the questions. After that, classification techniques and the classifiers will be investigated. Furthermore, we will demonstrate the approaches used in this part. In Fig. 2, the procedure of converting words to vectors and classifying the questions to related classes are illustrated in our proposed approaches. After, we will use question classification algorithm to classify questions utilizing Word2vec methods both skip gram and Continues Bag of Grams. Figure 2 shows the general architecture of this study.

3.1 Question database

There is a lack of Turkish question datasets in comparison with English. In this study, we use a Turkish question dataset, which is translated by us from an English Question Dataset. To improve the quality of translation of this

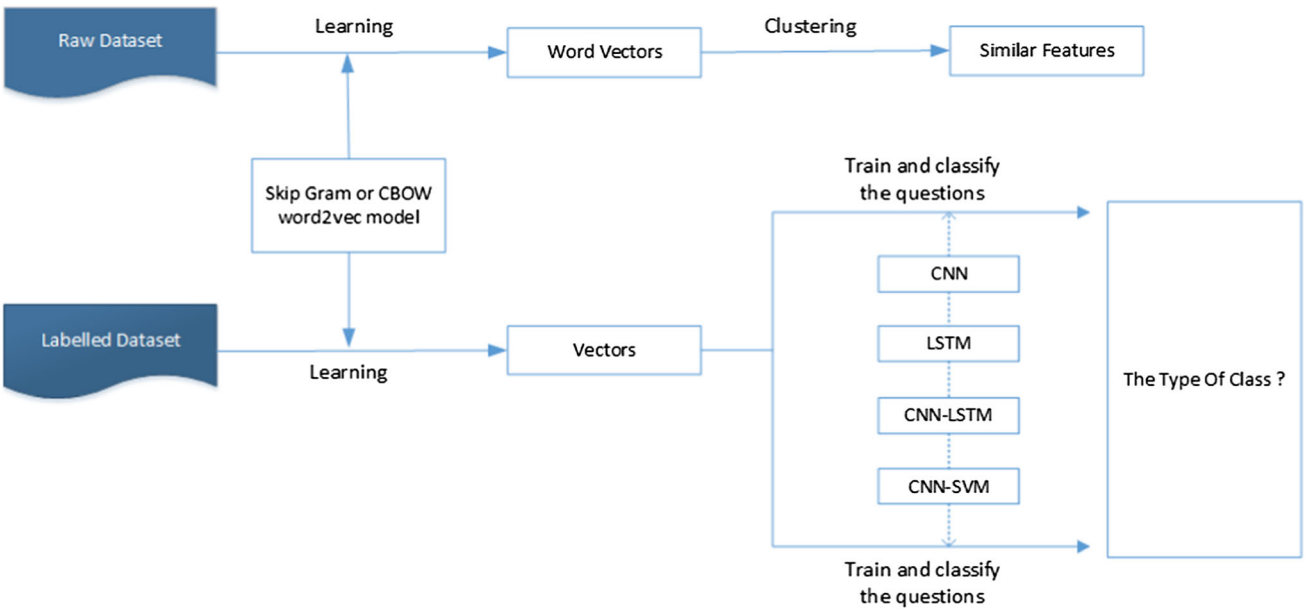

Fig. 2 The general architecture of this study

Table 2 Distribution of question categories in the UIUC Question dataset

Category	# Train	# Test	Category	# Train	# Test
ABBREVIATION	86	9	Animal	112	16
abb	16	1	Creative	207	0
exp	70	8	Other	217	12
DESCRIPTION	1162	94	HUMAN	1223	65
Reason	191	6	Description	25	3
Description	274	7	Group	47	6
Manner	276	2	Individual	189	55
Definition	421	123	Title	962	1
ENTITY	1250	94	LOCATION	835	81
Currency	4	6	Mountain	21	3
Religion	4	0	State	66	7
Letter	9	0	City	129	18
Instrument	10	1	Country	155	3
Symbol	11	0	Other	464	50
Plant	13	5	NUMERIC	896	113
Body	16	2	Order	6	0
Lang	16	2	Temp	8	5
Word	26	0	Code	9	0
Vehicle	27	4	Speed	9	6
Technique	38	1	Weight	11	4
Color	40	10	Size	13	0
Substance	41	15	Period	27	8
Product	42	4	Distance	34	16
Event	56	2	Other	52	12
Sport	62	1	Money	71	3
Term	93	7	Percent	75	3
Dis.med.	103	2	Date	218	47
Food	103	4	Count	363	9

database, we have received assistance from a professional company [51]. This database uses Li and Roth's (2002) taxonomy. They introduced a two-layered classification, which is broadly utilized for question classification. This dataset consists of 6 coarse categories and 50 fine-grained categories that is indicated as 'Coarse: fine, such as "LOCATION:city". All main classes and sub-classes of this question database [11] are shown in Table 2. In this database, there are 5500 questions in training data and 500 questions in test data in total. We built our Turkish dataset from this English dataset for our experiments, and we are planning to make this dataset available publicly for research purpose.

3.2 Word2vec

The Word2vec model was introduced in Mikolov [18, 52] and is illustrated in Fig. 3. Word2vec is a shallow, two-layer neural networks which is trained to reform linguistic contexts of words. This algorithm produces word vectors from an enormous amount of text of words as input by observing the contextual data the input words appear in [53]. In the Word2vec vector space, the dimensionality of these vectors becomes usually a few hundred dimensions. In the text, every distinctive word is assigned to a related vector in the Word2vec space [54]. In the Word2vec space, the vectors of the words are located such that if words will occur in similar contexts, and thus, the Word2vec algorithm will discover that such words ought to be positioned in near proximity to one another in the word space. It is a mainly computationally well-organized predictive system for learning word embeddings from raw corpus.

Wored2vec has two different approaches:

1. Skip gram
2. CBOW (Continuous Bag of Words)

Those two methods are algorithmically close to each other [55]. In Continuous Bag of Words (CBOW) architecture, the algorithm predicts center words(target) based on the neighboring words. It is a statistical result that Continuous Bag of Words smoothes on numerous the

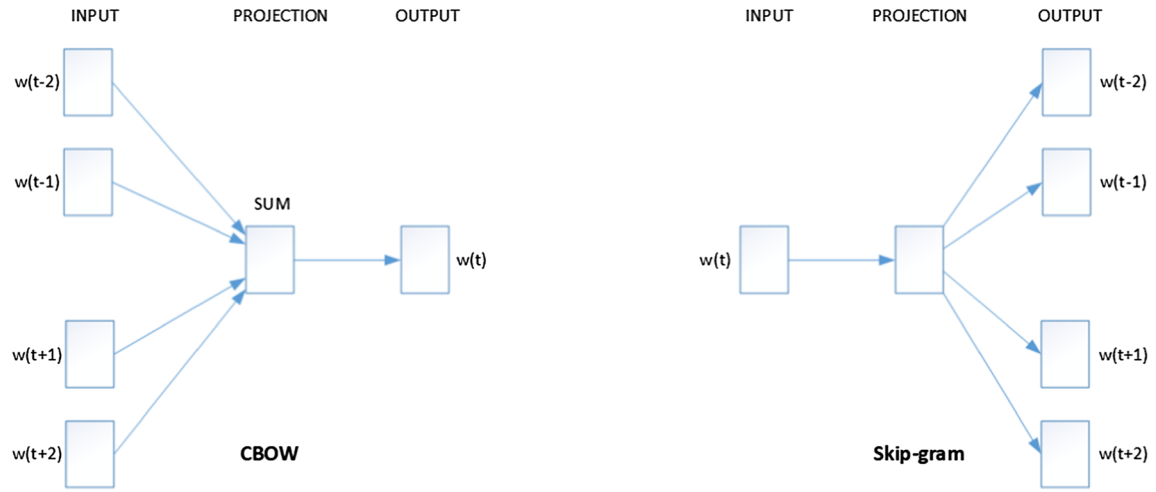

Fig. 3 The skip gram and CBOW model

distributional corpus, which treats a whole context as one observation. That becomes a positive thing for small corpus.

The skip gram algorithm is illustrated in Fig. 3. Skip gram is very similar to CBOW, except that it exchanges the output and input. This is the inverse of Continuous Bag of Words. It predicts all surrounding words ("context") from one input word. Essentially, the training aim of the skip gram algorithm is to discover word vectors that are useful at finding close words in the related contexts [56]. In skip gram model, nearby context words are predicted from the center word (opposite of Continuous Bag of Words). When the corpus is bigger, skip gram model is more effective, because in skip gram model every context-center pair is treated as a new observation. The Continuous Bag of Words and skip gram architecture can be seen in Fig. 3.

3.3 Training Word2vec embedding model

As part of this study, we train Word2vec models both skip gram and CBOW with different parameters on Wikipedia corpora. We train our Word2vec model for word representations on a corpus: a Turkish Wikipedia. We prefer the Wikipedia as the dataset since it is the biggest encyclopedia which is open, multilingual on the Internet; its documents are organized by topics clearly. Thus, Wikipedia corpora are very appropriate for the proposed Word2vec model. While training on Wikipedia corpora, we remove words with less than 5 words because these words have a smaller amount data and are generally meaningless for training the Word2vec in our study (for example, some have stop words, emoticons). By using these Wikipedia corpora, we have built our skip gram and CBOW model with different vector length 100, 200, 300 and 400. The Wikipedia used in our study is:

Tr-Wiki: a Turkish Wikipedia snapshot numbered 20180120.

In order to implement our Word2vec model, we use Gensim [57] to produce a set of word embeddings by setting the dimensionality D, the context window size W, the total of negative samples ns and the skip gram sg. The default parameter value for the context window size is chosen as $W = \{5\}$. For this context size of the window, four different dimensionality sizes $D = \{100, 200, 300, 400\}$ are used to investigate both of the high and low dimensions for the Word2vec vectors. Consequently, totally 8 different Word2vec models in total are produced by changing W and sg for Wikipedia corpora. For the rest values, we chose the default parameters. We set the negative sampling to 5, batch words to 10,000, minimum count of words to 5 and iteration to 5. Furthermore, we investigate the impact of the dimensionality vector on Wikipedia corpus. About 1 million Turkish articles are seen in Turkish Wikipedia. After eliminating words with a frequency less than 5, more than 200 thousand Turkish words are gathered in the corpus. The main parameters used in this study are as follows.

3.4 Visualization of word embeddings

The authors of [58] introduced an embedding method to visualize syntactic and semantic analogies and they made experiments to show whether the resulting visualizations obtain the noticeable architecture of the word embeddings produced with word Word2vec.PCA (principal component analysis), semantic axis, Cosine distance histogram are utilized for the visualization methods [59, 60]. In this study, principal component analysis is used to show the syntactic and semantic relations of the words in two-

dimensional space because most papers use PCA to visualize their data.

As I mentioned above, one of the main parameters of Word2vec is "the dimension of embedding layer". The dimension of embedding layers how many dimensional spaces the words are represented in. For example, if the dimension of embedding layer is 100, our words are calculated in 100-dimensional space. The embedding layer has very high dimension, and we should use a technique such as PCA to decrease the dimension for visualization. After that, the PCA technique can then be used to visualize the words represented in this 100-dimensional space in two-dimensional space. The models we will visualize have many thousands of word vectors and most words are overlapped on the vector space, and therefore, it is impossible to see all word vectors in a single plot, which is illustrated in Fig. 4.

After training, we visualize one of the learned Word2vec embeddings as an example, which is shown in Fig. 4. In this way, it keeps related words close together on the graph, while maximizing the distance between unrelated words [61]. In order to classify the questions based on the above-mentioned features, we will train four classifiers, using following models.

1. CNN (Convolutional Neural Networks): In NLP models, Convolutional Neural Network has been used convolving a filter with a fixed long representation of words as input into the architecture [62], and final classification is gained throughout a sequence of nonlinear function mappings and matrix multiplications [63]. The semantics of local form of textual data such as a phrase can be learned automatically by such architectures, while refraining from having to take memory over the whole sequence of words for a given sequence [62]. In our study, for the question

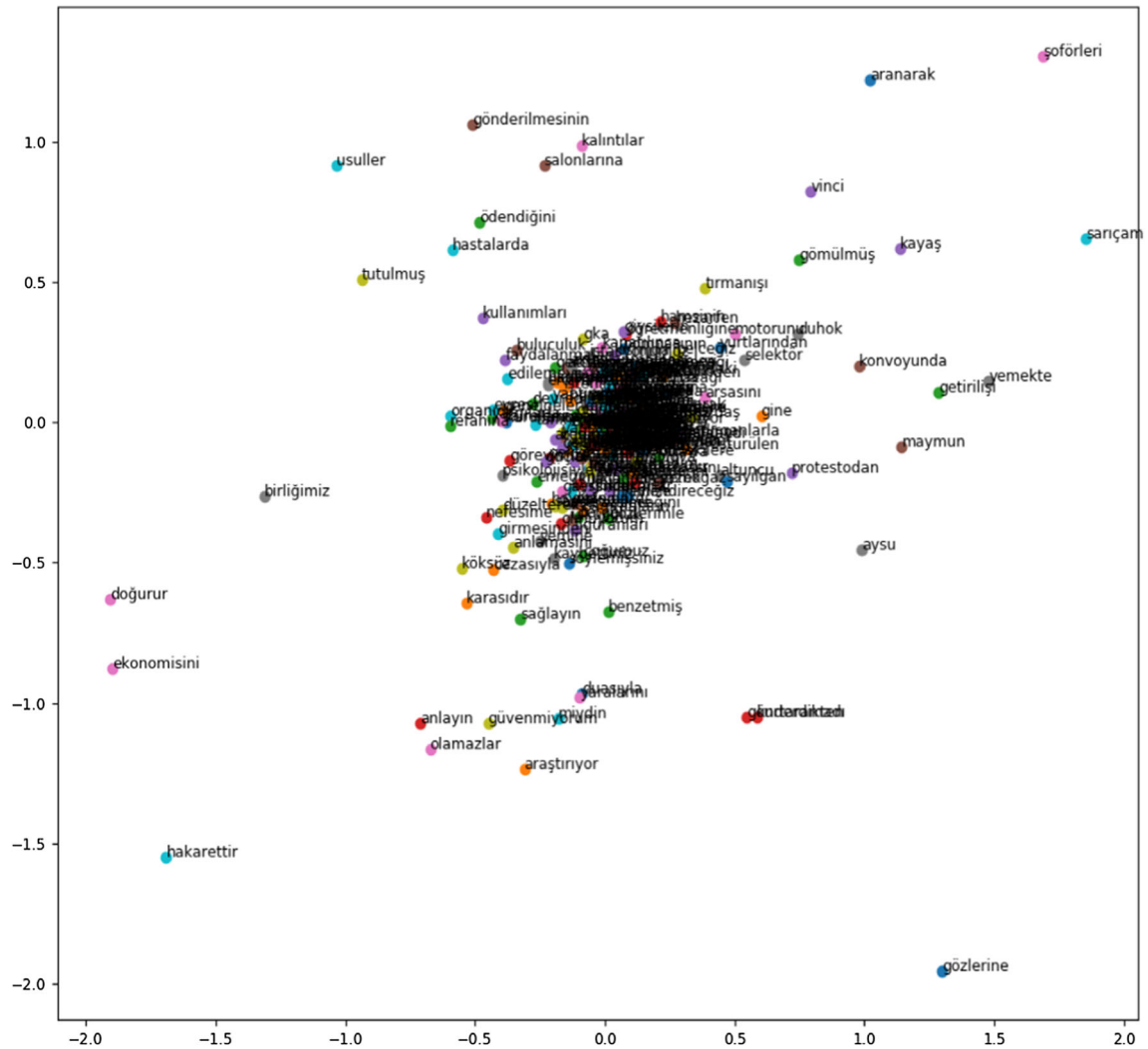

Fig. 4 2D results for our one of the Word2vec models with PCA

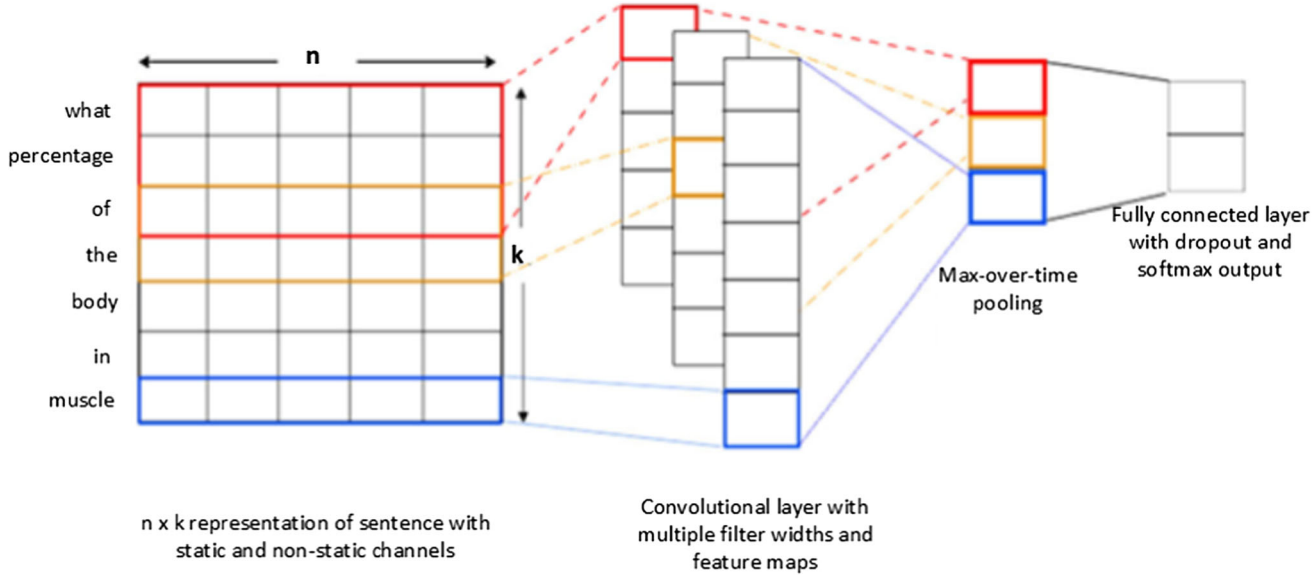

Fig. 5 Convolutional Neural Networks (CNN) for sentence classification model architecture for an example

classification performance of a Convolutional Neural Network, the efficient word vector representations are significant factors. Word2vec methods [52] are able to produce vectors which allow the architecture to be successful on the task of relating the words to their context in a given window [64].

In order to classify questions, we used a Convolutional Neural Networks (CNN) model explained by [8, 12, 65] and highlighted in Fig. 5. To make our system appropriate for this study, we changed some parameters, which is explained as follows. The CNN model has three layers: the convolutional layer, max-pooling layer, the dropout layer, the fully connected layer. In order to construct a classifier utilizing a CNN, every question is shown as a word embedding matrix. Given a question containing n words v_1, v_2, $v_{3,\ldots,}v_n$, every word with its d-dimensional pre-trained word embedding matrix are replaced, and stacked row-wise to generate an instance matrix $V_i \in \mathbb{R}^{n\times d}$

Now, a description of each layer will be explained in detail.

In a question, define a vector $V_i \in \mathbb{R}^d$, where V_i represent a d-dimensional embedding vector for the ith word.

In here, the questions are padded to make all questions the same length of words as the maximum length of the question, because all questions must be the equivalent length of words in a CNN model. Every question j is shown with a two-dimensional $n \times k$ matrix $\boldsymbol{c}_j = [v_1, v_2 \ldots, v_n]$, the definition of k denotes the dimensionality of the **Vi** embedding and n represents the longest count of words. In order to generate a map feature q for r words, a convolutional process on the $i{:}i + r$ sub-matrix is applied by a filter **T** with region size $r \times k$. Mapping feature q_i is produced via a convolutional layer by using a nonlinear function F.

$$q_i = \mathrm{F}(\mathbf{T} \cdot \boldsymbol{c}_{i:i+r} + b), \tag{1}$$

where F is a nonlinear function, for example, RELU (the Rectified Linear Unit), T represents a filter $\in \mathbb{R}^{rk}$, bias term is represented by b and $\boldsymbol{c}_{i:i+r}$ represents sub-matrix produced from r words. Thus, a question is mapped by the Convolutional Neural Network with n words to $n - r + 1$ features $q = [q_1, q_2, q_3, \ldots, q_{n\text{-}r+1}]$. In here, every$q$shows the higher word representations in the filter.

In our model, the maximum pooling layer is utilized to take the feature q with the maximum value to choose the most significant mapping features generated via a filter. To gain multiple representations, we can apply filters with dissimilar region sizes and convolutional functions for every question. The output of max-pooling layer for entire filters is then transferred to the fully connected layer with six probabilistic output units, which is one of the database class labels. The dropout layer is set as 0.5. Dropout is frequently applied at this layer as a means of regularization [66]. Hyper and training parameters used in this method are RELU. This extracts the maximum value oi from each feature map i (1-maximum pooling). Specifically, we choose ReLU(the Rectified Linear Unit) because RELU is broadly employed in modern deep neural network architecture like CNN. Compared to traditional activation functions such as sigmoid and tanh, ReLU offers a few benefits: (1) it is simple and fast to perform, (2) it mitigates the vanishing gradient problem and (3) it induces sparseness [67].

In our CNN model, we also use filter windows of 5, 4, 3 with 400, 300, 200 and 100-feature map search l2

Table 3 The main parameters used in the Word2vec models

Parameters	Value
Sentences	None
Windows size	5
Dimensionality	400, 300, 200, 100
max_vocab_size	None
sg({0,1}	1 for skip gram otherwise CBOW
hs ({0,1}	0
Negative	5
hashfxn	
Iter	5
Sample	0.001
Batch_words	10,000
Seed	1
min_alpha	0.0001
min_count	5
cbow_mean	1
null_word	0
Workers	4
trim_rule	None
sorted_vocab	1
Alpha	0.025

constraint of 3. The results of CNN model using Word2vec is illustrated in Table 3.

In addition to this, accuracy is utilized for evaluation metric. The count of correctly classified questions (the count of accurately classified samples) divided by the entire of tested questions (the number of tested samples) defines the accuracy [68]. In question classification, the formula of the accuracy is shown as follows.

$$\text{Accuracy} = \frac{\text{the count of correct classified questions}}{\text{the count of test questions}} \quad (2)$$

Especially in this study, accuracy is coarse because coarse groups are used for classification. In our study, we used both two Word2vec word embeddings CBOW and skip gram as feature extraction techniques. In addition to this, we also compare the results in terms of 10-cross fold validation accuracy.

2. LSTM (long short-term memory): LSTM is an updated variation of recurrent neural networks (RNN) and it plays a vital role for computers to understand text documents.

BPTT (The backpropagation through time algorithm) and RTRL (the Real-Time Recurrent Learning algorithm) expand the ordinary backpropagation algorithm to fit the RNN model, but Recurrent Neural Network has the disadvantages of remembering long past data. Through RTRL and BPTT, error signals incline to disappear after steps of flowing, incapable of changing weights [9]. Consequently, long-term dependencies are difficult to learn [9] (for more comprehensive analysis in (1)–(3) [69]). LSTM is able to hold data over much longer periods than (10–12) time steps [70], which is the boundary of BPTT and RTRL systems. It uses prior data to the current task and are implemented to allow machines to learn to overcome struggles whose input and output are sequence with diverse length.

Specifically, we use LSTM to classify questions using Word2vec representations. As shown in Fig. 6, LSTM provides a gating structure, which includes four parts: an output gate o_t, a forget gate f_t, a memory cell C_t and an input gate i_t. All three gates take the data from the inputs at present time step and the outputs at prior time step for the classic LSTM. All symbols are demonstrated in Table 5 where the present time step and the prior time step are defined with the symbols $t-1$ and t correspondingly. C acts like a conveyor belt and the data goes through the belt. The previous data are deleted, and fresh data are put via f and i. The output for present time step with o is generated by the system. The most important part of the long short-term memory architecture is C. This retains the data for long period. Therefore, long-term dependencies are supported by this mechanism. The mathematical equations summary the above-mentioned process as follows.

$$f_t = \varphi\left(W_t * [h_{t-1}, x_t] + b_f\right) \quad (3)$$

$$i_t = \varphi(W_i * [h_{t-1}, x_t] + b_i) \quad (4)$$

$$o_t = \varphi(W_o * [h_{t-1}, x_t] + b_o) \quad (5)$$

$$C_t^{'} = \tanh(W_c * [h_{t-1}, x_t] + b_c) \quad (6)$$

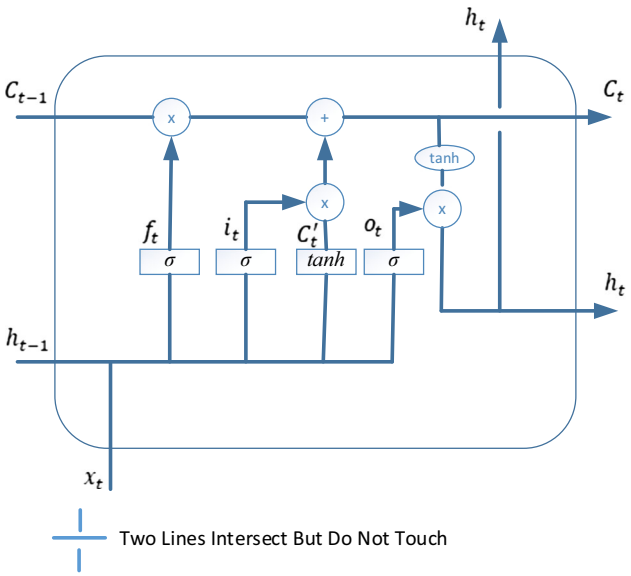

Fig. 6 The LSTM architecture

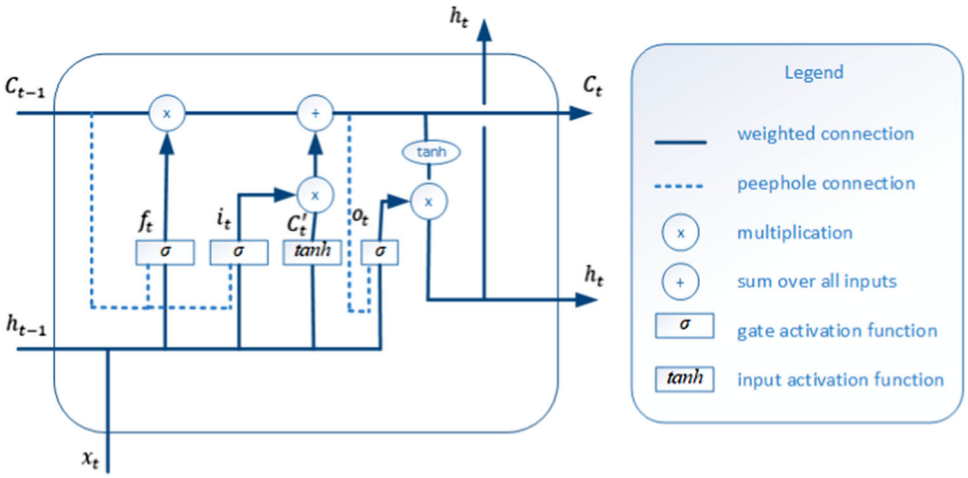

Fig. 7 LSTM structure with peepholes

$$C_t = f_t * C_{t-1} + i_t * C_t^{'} \tag{7}$$

$$h_t = o_t * \tanh(C_t) \tag{8}$$

We have explained the standard LSTM structure so far. But there are various LSTM structures which support the development of learning long-term dependencies. All LSTMs are not the same as the standard one. One well-known LSTM structure that inserts "peephole connections" is used in our method. In this model, it allows the gate layers look at the cell state. Figure 7 shows LSTM structure with peepholes.

In [71], the authors propose a sigmoid layer named the "forget gate layer" in order to remove some of previous data from the memory cell and make more the space of memory for new arriving data. In [69], the cell status is being monitoring by gate layers via "peephole connections". According to [72], such connections improve performance on time-retaining mechanisms in which the network has to learn to compute exact intervals among events.

In this section, we use a variation of LSTM, which includes the "peephole connections" to the mechanism of Fig. 6 (illustrated in Fig. 7) to let the memory cell C_{t-1} directly control the gates as shown the equations:

$$f_t = \varphi\left(W_f \times [C_{t-1}, h_{t-1}, x_t] + b_f\right) \tag{9}$$

$$i_t = \varphi(W_i \times [C_{t-1}, h_{t-1}, x_t] + b_i) \tag{10}$$

$$C_t^{'} = \tanh(W_c * [h_{t-1}, x_t] + b_c) \tag{11}$$

$$C_t = f_t * C_{t-1} + i_t * C_t^{'} \tag{12}$$

In these equations, the transition matrix for the input x_t is W, the memory cell C_{t-1}, component-wise multiplication is shown as x, the hidden state vector h_{t-1} and φ indicates the sigmoid function.

The output gate o_t controls the present hidden state value h_t, which applies to the system result of a nonlinearity to the memory cell contents:

$$o_t = \varphi(W_o \times [C_t, h_{t-1}, x_t] + b_o) \tag{13}$$

$$h_t = o_t * \tanh(C_t) \tag{14}$$

At the following phase, the present time step of the hidden state h_t is utilized for the acquisition of h_{t+1}. In other words, the word sequence is recursively processed by long short-term memory by calculating their internal

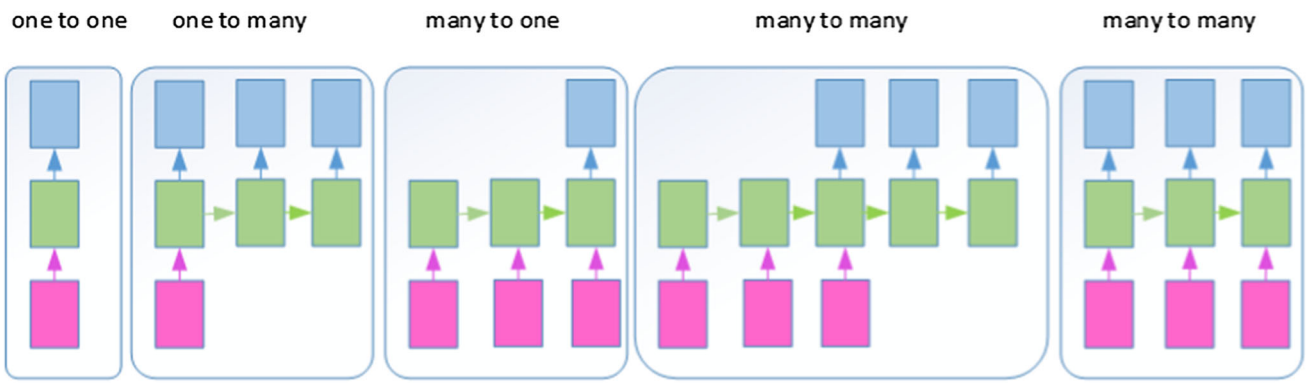

Fig. 8 Types of architecture mapping in LSTM

Fig. 9 The details of many to one structure

hidden state h_t at every time phase. The hidden activations of the final time phase can be taken into consideration as the semantic representation of the entire sequence and utilized as input for classification layer.

Furthermore, there are various types of architecture mapping in LSTM [73]. These are one-to-one, one-to-many, many-to-one and many-to-many mappings as represented in the Fig. 8.

A many-to-one structure produces one output value after taking multiple input values. In this structure, the input is in the form of a sequence, and thus, the hidden states functionally depend on both the input at that time step and the prior hidden state, for example, sequence input (for example, tweets where an input tweet is categorized as expressing negative or positive sentiment). Because of that, many-to-one model is selected in the case of question classification. In many-to-one architecture, there is only one output. For example, our input is, "Avrupa'nın en büyük ülkesi hangisidir?" ("What is the biggest country in Europe?"). In this question classification example, the input is a sentence where the input is a question (sequence of words) and the output is a probability showing that the input question is in a country. Table 6 illustrates the results of LSTM architecture with Word2vec.

The details of the many to one structure can be seen from Fig. 9, every rectangle presents a vector and arrows are functions, for example, matrix multiply. The output vectors are shown at the top and the input vectors are at the bottom, and vectors in the middle take the state of RNN [74].

3. CNN-LSTM model: The basic architecture of the proposed model is illustrated in Fig. 10, which is adapted from [65, 71, 75], and it summaries the combination of the two deep neural network models: CNN and LSTM.

In this method, a softer form in which the maximum operation is executed over a smaller region of the feature maps is used rather than max pooling over time. With this method, temporal data are better preserved and a sequence is generated rather than a single value by max-pool layer [71]. In the following layer, the data are then fed into a LSTM cell with many to one structure and a fully connected layer with softmax output. Table 6 illustrates the results of CNN-LSTM model using Word2vec.

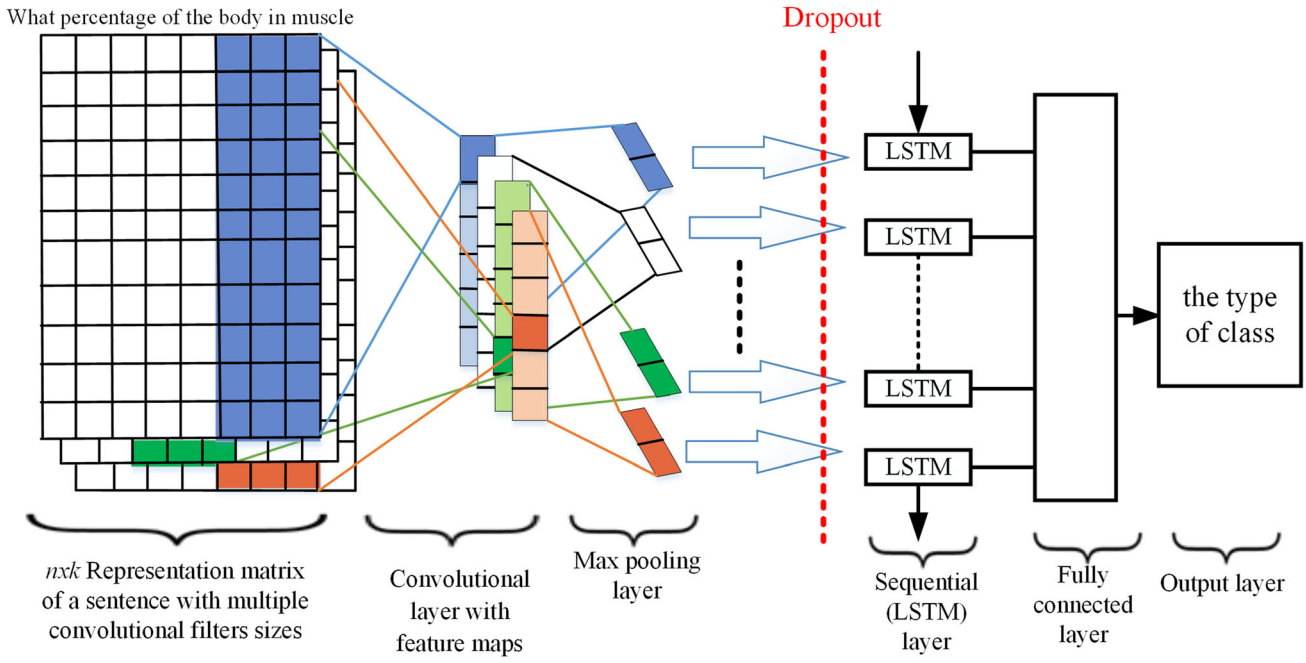

Fig. 10 CNN-LSTM structure with an example

3.4.1 Input layer

In this architecture, this is the initial part and it shows each question as a row of vectors. Every vector denotes a token based-upon word level. Every word in the question will be shown into a specific vector with one of the fixed sizes of 400, 300, 200 or 100 based upon the Word2vec model used. Every question j is shown with a two-dimensional $n \times k$ matrix c_j =[$v_1, v_2 \ldots, v_n$], the definition of k denotes the dimensionality of the v_i embedding and n denotes the longest count of words. The questions are padded using $<$ Pad $>$ in order to make all questions the equivalent length of words as the highest length of the question since all questions must be the equivalent length of words in this layer.

3.4.2 Convolutional layer with feature maps

Every input has a sequence of vectors, and this layer scans it with a fixed length of filter. In this method, the filter sizes of 3,4 and 5 are utilized to take the features of words. The filters shift or stride merely single row and single column matrix. Every filter takes various features in a question with the Rectifier Linear Unit function (ReLU) in order to represent them in the feature map.

3.4.3 Max-pooling layer

This layer down-samples and reduces the features in the feature map after the Convolutional layer. The maximum function or operation is the most usually employed method at the max-pooling layer. For that reason, we selected the maximum operation in this study. In order to decrease the computation in the advanced layers and take the most significant feature, the maximum value is selected. After that, the system applies the dropout method with the dropout value 0.5 to decrease overfitting.

3.4.4 LSTM layer

The aim of choosing the LSTM is to take the sequential information by using the prior information. In LSTM layer, the output vectors from the dropout layer are taken as inputs. In this layer, all cell inputs are the output from the dropout layer and the layer contains a set number of units and cells. In this layer, the last output of the layer has the equivalent number of units.

3.4.5 Fully connected layer

LSTM layer produces a single matrix from its outputs by combining and merging them, and after that, this layer passes it to a fully connected layer. In order to classify questions, the network converts the array into an output using the fully connected layer and softmax.

4. CNN-SVM: SVM (Support Vector Machine) is a supervised machine learning algorithm, which can be applied for both regression and classification problems [10]. The aim of supervised machine learning is to generalize and learn an input output mapping. In question classification problems, a set of questions are used as input and their relevant class is used for output.

In the previous deep architectures, we use the last layer as a fully connected layer with softmax output, which is equal to a Linear Classifier. The prior layers perform as a feature extractor in this condition. In this way, we follow the authors of [71]. In this study, it is shown that that this feature extractor can extract valuable features that present points that can be divided satisfactorily through a simple Linear Classifier. The general framework of CNN-SVM architecture is shown in Fig. 11. In addition to that, those features can be helpful for other type of Linear Classifier. In this method, Linear SVM is used as the last layer. In here, the typical cross-entropy loss function.

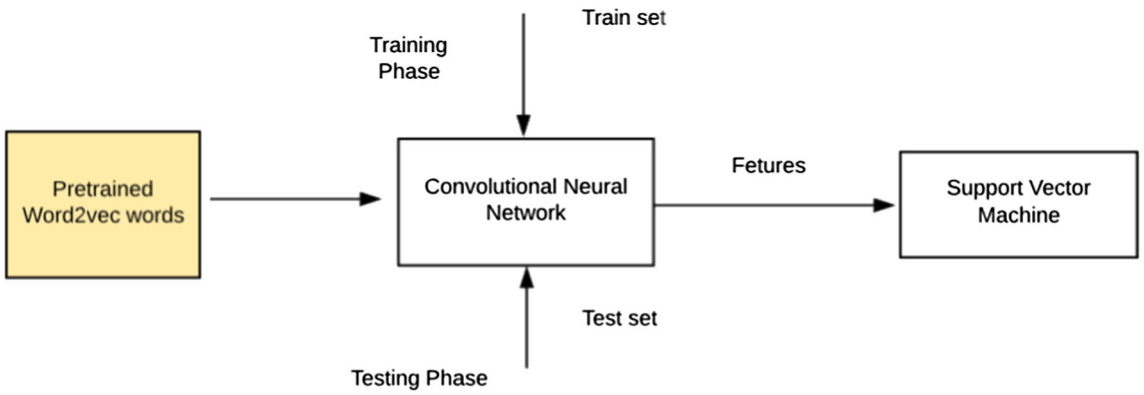

Fig. 11 The general framework of CNN-SVM architecture

$$L = CE(target, softmax(W\emptyset(input))) \quad (15)$$

is replaced with Linear SVM's loss function. It is the extracted features from the previous layers. Where CE is *CrossEntropy*, and L is Loss. After replacing SVM as the final layer, loss function is as follows:

$$L = \sum_{i}^{n}\sum_{j}^{6} R(-t_j W_{j\phi i} + 1) + C\left|W_j\right|^2 \quad (16)$$

where R is RELU(Rectifier Linear Unit function); t is the target that can be -1 or 1. This loss function shows the loss when one vs rest scheme is utilized, since our question classification data are using a multiclass dataset. For Linear SVM with a soft margin, the loss function is now.

$$L = \sum_{i}^{n}\sum_{j}^{6} R(-t_j W_{j\phi i} + 1 - \xi_{ij}) + R\left(-\xi_{ij}\right) + C\left|W_j\right|^2 \quad (17)$$

In Table 7, the results of CNN-SVM model using Word2vec are illustrated.

4 Conclusions

In this article, we apply some deep learning methods on question dataset based upon Word2vec embedding vectors both skip gram and CBOW, which can effectively capture the semantic and syntactic relations among words. To do this, firstly the Word2vec algorithms calculate word vectors of vocabulary words. The algorithms initialize word vectors with random vectors. Then, the algorithms try to raise the cosine similarity among all words and their context, which can be defined based upon the system. Therefore, by providing big amount of text Wikipedia corpus for these algorithms, they will be able to allocate word vectors in the vector space such that their closeness is proportional to relevant to their corresponding words.

Table 4 The results of CNN model using Word2vec

Number of feature Vectors	The type of Word2vec model	Accuracy (Test)	Accuracy (10-cross fold validation)
100	CBOW	91.8	86.5
100	Skip gram	92.8	89.3
200	CBOW	92.4	86.7
200	Skip gram	92.6	89.5
300	CBOW	92.4	87.2
300	Skip gram	94	89.6
400	CBOW	92.4	86.9
400	Skip gram	93.8	89.5

Table 5 The symbols used in LSTM architecture

Symbol	Description
C	CEC (The memory cell)
C'	The fresh candidate value
z, r, f, i, o	update, reset, forget, input, and output gates
φ	Sigmoid function
h	Output
b_C, b_z, fb_f, b_o, b_i	Offset values
$tanh$	Hyperbolic tangent function
x	Input
W_c	The weight vector for input
W_z	The weight vector for the update gate
W_o, W_i, W_f	The weight vectors of the output, input, forget gates separately
$\otimes$, $\times$	Component-wise multiplication
*	Product of two scalars or product of a scalar and a vector
$1-$	All elements of vectors are subtracted from 1
$\oplus$	Element-wise sum of two vectors

Table 6 The Results of LSTM using Word2vec model

Number of feature vectors	The type of Word2vec model	Accuracy (test)	Accuracy (10-cross fold validation)
100	CBOW	91	87.8
100	Skip gram	90.8	86.9
200	CBOW	90.6	87.6
200	Skip gram	90.3	87.7
300	CBOW	90.2	88.2
300	Skip gram	91.4	88.4
400	CBOW	91.6	88.5
400	Skip gram	91	88.5

Table 7 The Results of CNN-LSTM architecture

Number of feature vectors	The type of Word2vec model	Accuracy (test)	Accuracy (10-cross fold validation)
100	CBOW	90.6	86.9
100	Skip gram	91	88.6
200	CBOW	92	89.5
200	Skip gram	92.8	88.9
300	CBOW	91.2	87.7
300	Skip gram	92.6	89
400	CBOW	91.4	87.4
400	Skip gram	92.8	89.6

Table 8 The Results of the CNN-SVM model

Number of feature vectors	The type of Word2vec model	Accuracy (test)	Accuracy (10-cross fold validation)
100	CBOW	91.8	83.4
100	Skip gram	91.8	86.8
200	CBOW	91.2	81.1
200	Skip gram	92.8	87.1
300	CBOW	90.8	84
300	Skip gram	93	89.5
400	CBOW	92	84.6
400	Skip gram	93.2	89.4

These methods used for classification are based on Word2vec both skip gram and CBOW and deep learning techniques together. This study focuses on agglutinative language and to the best of our knowledge; this is for the first time that question classification is studied in an agglutinative language in depth. As a result, we have reached satisfactory statistics according to the experimental results of classes. Prior studies on question classification concentrate on different tasks for instance the named entity or similar class categorizing [76] and integrating a rule-based technique using an HMM-based sequence classification technique [77] their answer might not generalize to question classification tasks in an agglutinative language.

On the other hand, similar works other languages on question classification studies [6, 71] have not investigated the impact of the Word2vec variations both CBOW and skip gram and some parameters such as feature vector size on question classification performance. The experimental results in this study illustrates that those above-mentioned

factors can definitely influence the classification performance on question classification systems.

Generally, in our study, we investigated CNN, LSTM, CNN-LSTM and CNN-SVM when using Word2vec methods both CBOW and skip gram. While using two different types of the Word2vec method, the CNN, CNN-LSTM and CNN-SVM model using skip gram is able to perform significantly better results in terms of accuracy on question classification dataset in comparison to using CBOW (Tables 4, 7, 8). In contrast to CNN, CNN-LSTM and CNN-SVM, using CBOW generally achieve better results on LSTM structure. In CNN-LSTM model, using skip gram is better than using CBOW in most cases. In addition to that, we have experienced the best result in CNN model, an accuracy of 94%, when using skip gram with 300 feature vectors. In addition to this, we experience that using the right form of dataset can potentially contain more vocabulary for the classification database. Therefore, the relation between corpus and the classification dataset give superior question-level representations.

Lastly, when compared to a similar study on [71] performed on the same dataset, which the authors have reached an accuracy of 95.4% with LSTM in English language; our results were low compared to this study conducted in English. The most important reason for this is the language structure of Turkish as we mentioned earlier in Sect. 1. Consequently, there is an absence of effective lemmatization tools for Turkish Language compared to English language.

To conclude our study, we recommend here one line for the future work that we think may be motivating to explore. For the future, a hybrid feature extraction technique based on more than one word embedding method together can be used to increase the accuracy of the question classification system. By using this hybrid method, the system will be able to take the advantages of all embedding methods used.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Blooma MJ, Goh DHL, Chua AYK, Ling Z (2008) Applying question classification to Yahoo! Answers. In: Applications of digital information and Web Technologies, ICADIWT 2008. First international conference on the IEEE, pp 229–234
2. Silva J, Luísa C, Mendes AC, Andreas W (2011) From symbolic to sub-symbolic information in question classification. Artif Intell Rev 35(2):137–154
3. Mishra M, Mishra VK, Sharma HR (2013) Question classification using semantic, syntactic and lexical features. Int J Web Semant Technol 4(3):39
4. Zhiheng H, Marcus T, Zengchang Q (2008) Question classification using head words and their hypernyms. In: Proceedings of the conference on empirical methods in natural language processing. Association for Computational Linguistics, pp 927–936
5. Ehsan S, Mojgan F (2014) A hybrid approach for question classification in Persian automatic question answering systems 2014. In: 4th international e conference on computer and knowledge engineering (ICCKE). IEEE, pp 279–284(2014)
6. Razzaghnoori M, Sajedi H, Jazani IK (2018) Question classification in Persian using word vectors and frequencies. Cogn Syst Res 47:16–27
7. Hao T, Xie W, Xu F (2015) A WordNet expansion-based approach for question targets identification and classification. In: Chinese computational linguistics and natural language processing based on naturally annotated Big Data. Springer, Cham, pp 333–344
8. Kim Y (2014) Convolutional neural networks for sentence classification. arxiv preprint: arxiv:1408.5882
9. Hu F, Li L, Zhang ZL (2017) Emphasizing essential words for sentiment classification based on recurrent neural networks. J Comput Sci Technol 32(4):785–795. https://doi.org/10.1007/s11390-017-1759-2
10. https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/
11. Le-Hong P, Phan XH, Nguyen TD (2015) Using dependency analysis to improve question classification. In Knowledge and Systems Engineering (pp. 653-665). Springer, Cham
12. Bilić P, Primorac J, Valtýsson B (eds) (2018) Technologies of labour and the politics of contradiction. Springer, Cham, p 85
13. Şahin G (2017) Turkish document classification based on Word2Vec and SVM classifier. In: Signal processing and communications applications conference (SIU), 2017 25th, IEEE, pp 1–4
14. http://user.ceng.metu.edu.tr/∼mturhan/dfa/node14.html
15. Ozturkmenoglu O, Alpkocak A (2012) Comparison of different lemmatization approaches for information retrieval on Turkish text collection. In: 2012 International symposium on innovations in intelligent systems and applications. IEEE
16. Oflazer K (2014) Turkish and its challenges for language processing. Lang Resour Eval 48(4):639–653
17. Wang C (2016) What are the limitations of the Bag-of-Words model? [Online]. Available: https://www.quora.com/What-are-the-limitations-of-the-Bag-of-Words-model
18. Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013) Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems, pp 3111–3119
19. Bollegala D, Maehara T, Kawarabayashi K (2015) Embedding semantic relations into word representations. In: Twenty-fourth international joint conference on artificial intelligence
20. http://cogcomp.cs.illinois.edu/Data/QA/QC/
21. Abdi A et al (2019) Deep learning-based sentiment classification of evaluative text based on Multi-feature fusion. Inf Process Manag 56(4):1245–1259
22. Ghulam H et al (2019) Deep learning-based sentiment analysis for Roman Urdu text. Procedia Comput Sci 147:131–135
23. Sangkeettrakarn C, Haruechaiyasak C, Theeramunkong T (2019) Fuzziness detection in Thai law texts using deep learning. In: 2019 10th International conference of information and communication technology for embedded systems (IC-ICTES). IEEE
24. Singh J et al (2018) Morphological evaluation and sentiment analysis of Punjabi text using deep learning classification. J King Saud Univ-Comput Inf Sci (2018)

25. Nguyen H-Q, Nguyen Q-U (2018) An ensemble of shallow and deep learning algorithms for Vietnamese Sentiment Analysis. In: 2018 5th NAFOSTED conference on information and computer science (NICS). IEEE
26. Nguyen T, Shcherbakov M (2018) A neural network based Vietnamese Chatbot. In: 2018 International conference on system modeling & advancement in research trends (SMART). IEEE
27. Dmitrin YV (dmitrinyuri@gmail.com), Botov DS Comparison of deep neural network architectures for authorship attribution of Russian Social Media Texts
28. Vo K et al (2019) Handling negative mentions on social media channels using deep learning. J Inf Telecommun 1–23
29. Heikal Maha, Torki Marwan, El-Makky Nagwa (2018) Sentiment analysis of Arabic Tweets using deep learning. Procedia Comput Sci 142:114–122
30. Hacioglu K, Ward W (2003) Question classification with support vector machines and error correcting codes. In: Proceedings of HLT-NAACL, Association for Computational Linguistics, Morristown, USA, vol 2, pp 28–30
31. Loni B (2011) A survey of state-of-the-art methods on question classification
32. Athira PM, Sreeja M, Reghuraj PC (2013) Architecture of an ontology-based domain-specific natural language question answering system. Int J Web Semant Technol 4(4):31
33. Hermjakob U (2001) Parsing and question classification for question answering. In: Proceedings of the workshop on open-domain question answering. Association for Computational Linguistics, vol 12, pp 1–6
34. Close LW (2002) Question classification using language modeling. Center of Intelligent Information Retrieval (CIIR). Technical report
35. Dell Z, Wee SL (2003) Question classification using support vector machines. In: Proceedings of the 26th annual international ACM SIGIR conference on research and development in information retrieval. ACM, pp 26–32
36. Suzuki J, Taira H, Sasaki Y, Maeda E (2003) Question classification using HDAG kernel. In: Proceedings of the ACL 2003 workshop on multilingual summarization and question answering. Association for Computational Linguistics, vol 12, pp 61–68
37. Li X, Roth D (2002) Learning question classifiers. In: Proceeding of the 19th international conference on computational linguistics. Association for Computational Linguistics, Morristown, USA, vol 1, pp 1–7
38. Mollaei A, Rahati-Quchani S, Estaji A (2012) Question classification in Persian language based on conditional random fields. In: 2012 2nd international conference on computer and knowledge engineering (ICCKE). IEEE, pp 295–300
39. Loni BK, Seyedeh H, Wiggers P (2011) Latent semantic analysis for question classification with neural networks. In: 2011 IEEE workshop on automatic speech recognition and understanding (ASRU). IEEE, pp 437–442
40. Blunsom P, Kocik K, Curran J (2006) Question classification with log-linear models. In: Proceedings of the 29th annual international ACM SIGIR conference on research and development in information retrieval. ACM, pp 615–616
41. Santosh K, Ray Shailendra S, Joshi BP (2010) A semantic approach for question classification using WordNet and Wikipedia. Pattern Recognit Lett 31(13):1935–1943
42. Lee C-H, Lee H-Y (2019) Cross-lingual transfer learning for question answering. arXiv preprint arXiv:1907.06042
43. https://ahcweb01.naist.jp/DigRevURL/index.html
44. Liu, Jiahua, et al. "XQA: A Cross-lingual Open-domain Question Answering Dataset." Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. 2019
45. Singh J et al (2019) "XLDA: cross-lingual data augmentation for natural language inference and question answering. arXiv preprint arXiv:1905.11471
46. Faruqi MI, Purwarianti A (2011) An Indonesian question analyzer to enhance the performance of Indonesian-English CLQA. In: Proceedings of the 2011 international conference on electrical engineering and informatics. IEEE, pp 1–6
47. Sugiyama K, Mizukami M, Neubig G, Yoshino K, Sakti S, Toda T, Nakamura S (2015) An investigation of machine translation evaluation metrics in cross-lingual question answering. In: Proceedings of the tenth workshop on statistical machine translation, pp 442–449
48. Gupta D, Kumari S, Ekbal A, Bhattacharyya P (2018 MMQA: a multi-domain multi-lingual question-answering framework for English and Hindi. In: Proceedings of the eleventh international conference on language resources and evaluation (LREC-2018)
49. Loginova E, Varanasi S, Neumann G (2018) Towards multilingual neural question answering. In: European conference on advances in databases and information systems. Springer, Cham, pp 274–285
50. Ture F, Boschee E (2016) Learning to translate for multilingual question answering. In: Proceedings of the 2016 conference on empirical methods in natural language processing, pp 573–584
51. http://www.benila.com.tr/index.php
52. Mikolov, Tomas et al (2013) Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781
53. https://github.com/akoksal/turkish-word2Vec
54. https://israelg99.github.io/2017-03-23-Word2Vec-Explained/
55. https://medium.com/@mubuyuk51/word2vec-nedir-t%C3%BCrk%C3%A7e-f0cfab20d3ae
56. Mandelbaum A, Shalev A (2016) Word embeddings and their use in sentence classification tasks. arXiv preprint arXiv:1610.08229
57. https://radimrehurek.com/gensim/models/word2vec.html
58. Liu S, Bremer PT, Thiagarajan JJ, Srikumar V, Wang B, Livnat Y, Pascucci V (2018) Visual exploration of semantic relationships in neural word embeddings. IEEE Trans Vis Comput Graph 24(1):553–562
59. Chen Z et al (2018) Evaluating semantic relations in neural word embeddings with biomedical and general domain knowledge bases. BMC Med Inf Decis Mak 18(2):65
60. https://medium.com/@patrickhk/practice-ntlk-word2vec-pca-wordcloud-jieba-on-harry-potter-series-and-chinese-content-ca6f845b3293
61. https://www.tensorflow.org/tutorials/representation/word2vec
62. Zhang Y, Wallace B (2015) A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. arXiv preprint arXiv:1510.03820
63. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5300751/
64. Banerjee I et al (2019) Comparative effectiveness of convolutional neural network (CNN) and recurrent neural network (RNN) architectures for radiology text report classification. Artif Intell Med 97:79–88
65. Yang X, Macdonald C, Ounis I (2018) Using word embeddings in twitter election classification. Inf Retr J 21(2–3):183–207
66. Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–1958
67. https://arxiv.org/pdf/1707.08214.pdf
68. Liu Y et al (2018) Feature extraction based on information gain and sequential pattern for English question classification. IET Softw 12(6):520–526
69. Hochreiter S (1998) The vanishing gradient problem during learning recurrent neural nets and problem solutions. Int J Uncertain Fuzziness Knowl Syst 6(2):107–116

70. Zhou H et al (2016) Exploiting syntactic and semantics information for chemical–disease relation extraction. Database2016
71. https://github.com/thtrieu/qclass_dl/blob/master/ProjectDescription.pdf, https://github.com/thtrieu/qclass_dl/blob/master/ProjectPresentation.pdf
72. Gers FA, Schmidhuber J (2000) Recurrent nets that time and count. In: Proceedings of the IEEE-INNS-ENNS international joint conference on neural networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium, vol 3, pp 189–194
73. http://karpathy.github.io/2015/05/21/rnneffectiveness/
74. http://www.jackdermody.net/brightwire/article/Sequence_to_Sequence_with_LSTM
75. Alayba AM, Palade V, England M, Iqbal R (2018) A combined CNN and LSTM model for arabic sentiment analysis. In: International cross-domain conference for machine learning and knowledge extraction. Springer, Cham, pp 179–191
76. Derici C, Celik K, Kutbay E, Aydın Y, Güngör T, Özgür A, Kartal G (2015) Question analysis for a closed domain question answering system. In: International conference on intelligent text processing and computational linguistics. Springer, Cham, pp 468–482
77. Dönmez İ, Adalı E (2017) Turkish question answering application with course-grained semantic matrix representation of sentences. In: Computer science and engineering (UBMK), 2017 international conference on IEEE, pp 6–11