


Article

A Robust Morpheme Sequence and Convolutional Neural Network-Based Uyghur and Kazakh Short Text Classification

Sardar Parhat, Mijit Ablimit and Askar Hamdulla * 

Information Science and Engineering Institute, Xinjiang University, Xinjiang 830046, China; sardar312@163.com (S.P.); mijit@xju.edu.cn (M.A.)

* Correspondence: askar@xju.edu.cn

Received: 26 October 2019; Accepted: 4 December 2019; Published: 6 December 2019



Abstract: In this paper, based on the multilingual morphological analyzer, we researched the similar low-resource languages, Uyghur and Kazakh, short text classification. Generally, the online linguistic resources of these languages are noisy. So a preprocessing is necessary and can significantly improve the accuracy. Uyghur and Kazakh are the languages with derivational morphology, in which words are coined by stems concatenated with suffixes. Usually, terms are used as the representation of text content while excluding functional parts as stop words in these languages. By extracting stems we can collect necessary terms and exclude stop words. Morpheme segmentation tool can split text into morphemes with 95% high reliability. After preparing both word- and morpheme-based training text corpora, we apply convolutional neural network (CNN) as a feature selection and text classification algorithm to perform text classification tasks. Experimental results show that the morpheme-based approach outperformed the word-based approach. Word embedding technique is frequently used in text representation both in the framework of neural networks and as a value expression, and can map language units into a sequential vector space based on context, and it is a natural way to extract and predict out-of-vocabulary (OOV) from context information. Multilingual morphological analysis has provided a convenient way for processing tasks of low resource languages like Uyghur and Kazakh.

Keywords: Uyghur and Kazakh; text classification; CNN; morphology

1. Introduction

Uyghur and Kazakh are a kind of morphologically rich agglutinative languages, in which words are formed by a root (stem) followed by suffixes, therefore, the vocabulary size of these languages is huge. There are around 10 million Uyghur people and 1.6 million Kazakh people living in northwestern China. Officially, Arabic scripts are used for both languages, while Latin scripts are also widely used on the Internet, especially on mobile communications and social networks. The grammar and lexical structure of Uyghur and Kazakh are basically the same. Words in these languages are naturally separated in sentences, and are relatively long as the powerful suffixes can extend the stems (roots) semantically and syntactically, as shown in the example below:

Uyghur Arabic script form.

مۇسابىقىدە مۇسابىقىنىڭ ئاخىرقى مۇسابىقى نۇمۇرىنى ئېلىپ، تاللانما مۇسابىقىدىن غەلبىلىك ئۆتتى.

Uyghur Latin script form.

musabiqidA musabiqiniN vaHirqi musabiqA numurini velip, tallanma musabiqidin GALbilik vOtti.

Kazakh Arabic script form.

جارتا جارتىنىڭ سوڭغى جارس نومىرىن ئالىپ، تەڭداۋ جارىستان جەڭسپەن ئوتتى.

Kazakh Latin script form.

jaresta jaresneN soNGe jares nomeren alep, taNdaw jarestan jENespEn votte.

As the stems are the notionally independent word particles with a practical meaning, and affixes provide grammatical functions in Uyghur and Kazakh languages, morpheme segmentation can enable us to separate stems and remove syntactic suffixes as stop words, and reduce noise and feature dimension of Uyghur and Kazakh texts in classification task. The form of the sentences in the example given above are becoming the form shown below after morpheme segmentation:

Uyghur morpheme segmentation. **musabiqA** + dA **musabiqA** + niN vaHir + qi **musabiqA** numur + i + ni val + ip, talla + an + ma **musabiqA** + din GALbA + lik vOt + ty.

Kazakh morpheme segmentation. **jares** + ta **jares** + neN soNGe **jares** nomer + en al + ep, taNdaw **jares** + tan jENespEn vot + te.

There are 10 words in the above Uyghur and Kazakh sentences, and the stems (bold parts) of four words are /musabiqA/ (match) and /jares/ (match), respectively. After morpheme segmentation and stem extraction on the above sentences, only one stem was extracted from four words, so the number of features will be greatly reduced, as shown in Table 1.

Table 1. Uyghur and Kazakh word variants.

Stem	Variants	Suffix
(match)	(in the match) Uyghur: musabiqidA = musabiqA + dA Kazakh: jaresta = jares + ta	dA ta
Uyghur: musabiqA	(of the match) Uyghur: musabiqiniN = musabiqA + niN Kazakh: jaresneN = jares + neN	niN neN
Kazakh: jares	(from the match) Uyghur: musabiqidin Kazakh: jarestan = jares + tan	din tan

The Uyghur and Kazakh Arabic letters corresponding to the Latin letters are shown in Table 2. Here, La, Uy, and Ka in table headings stand for Latin, Uyghur, and Kazakh, respectively.

Both Uyghur and Kazakh languages transcribe speech as they pronounce which cause the personalized spelling of words especially less frequent words and terms. The main problems in natural language processing (NLP) tasks for Uyghur and Kazakh languages are the scarceness in resources and derivative morphology in language structure. Data collected from the Internet are noisy and uncertain in terms of coding and spelling [1]. Generally, Internet data for these low resource languages suffered from high uncertainty of writing forms on these languages, due to the deep influence of the major languages, Chinese and English [2]. This influence is greatly aggravated by the rapid development of information technology, which triggers a broad spectrum of cross-lingual and cross-cultural interaction, leading to unceasing coining of new words and new concepts. Most of these new items are borrowed from Chinese and English, and the integration is in forms that are full of noise caused by the different spelling habits [3]. Dialects and uncertainty in terms of spelling and coding pose a big challenge for reliability of extracting and classifying short and noisy text data.

Previous works [4] on stem extraction for Uyghur and Kazakh texts are mostly based on simple suffix-based stemming methods and some simple hand-crafted rules, which suffer from ambiguity, particularly on the short texts. Sentence or longer context-based reliable stem extraction methods can extract stems and terms accurately in noisy texts for Uyghur and Kazakh texts on a sentence level, and lead to the ambiguity reduction in a noisy text environment.

Table 2. Correspondence of Arabic and Latin letters.

No.	La	Uy	Ka	No.	La	Uy	Ka	No.	La	Uy	Ka
1	y	ي	ي	13	m	م	م	25	H	خ	خ
2	a	ا	ا	14	s	س	س	26	U	ؤ	
3	l	ل	ل	15	b	ب	ب	27	h	ھ	ھ
4	G	غ	ع	16	d	د	د	28	g	گ	گ
5	u	ؤ	ؤ	17	A	ه	ا	29	f	ف	ف
6	z	ز	ز	18	v	ئ	ئ	30	w	ؤ	ؤ
7	k	ك	ك	19	r	ر	ر	31	O	ؤ	ؤ
8	x	ش	ش	20	n	ن	ن	32	J	ژ	
9	i	ى	ى	21	N	ئ	ئ	33	j	ج	ج
10	t	ت	ت	22	c	چ	چ	34	E		ه
11	o	و	و	23	e	ې	ى	35	B		ؤ
12	p	پ	پ	24	q	ق	ق	36	C		ؤ

Text classification approaches based on convolutional neural networks (CNN) are extensively studied on major languages such as English and Chinese. P. Wang et al. [5] proposed semantic clustering and a CNN-based short text classification method. Reference [5] used a fast clustering approach to find semantic cliques by clustering word embeddings, and used the semantic units which meet the preset threshold to build semantic matrices, then put them into CNN. In this experiment, more than 97% of classification accuracy was obtained on TREC (Text REtrieval Conference) questions data set when using the GloVe (Global Vectors for Word Representation) model to pretrain word embeddings. R. Johnson and T. Zhang [6] proposed a text classification method based on CNN with a multiconvolution layer. CNN was used directly to the text data with high dimension and learned the local features of small text areas for classification of texts, and the bag of word (BoW) conversion operation was made in convolution layer in [6], and an excellent result with 9.33% error rate was obtained in topic classification experiment on RCV1 (Reuters Corpus Volume 1) data set with 103 topics. M. Zhu and X.D. Yang [7] proposed a Chinese text classification method based on CNN. They added attention mechanism after pooling layer to make improvements on the CNN model proposed by Kim [8], and achieved 95.15% of classification accuracy on Sohu news data with nine classes.

Some works on Uyghur and Kazakh text classification have been reported in [4,9,10]. Tuerxun et al. [4] used KNN (K-Nearest Neighbor) as a classifier on Uyghur text to conduct text classification, and used the TFIDF (Term Frequency-Inverse Document Frequency) algorithm to calculate the feature weight in this paper. Imam et al. [9] used the TextRank algorithm to select the features to make a sentiment classification experiment on Uyghur text, and SVM (Support Vector Machine) was used as a classifier in this experiment. Yergesh et al. [10] made a sentiment classification experiment on Kazakh text based on linguistic rules of Kazakh. The text classification methods used by the researchers mentioned above are the traditional classification framework in which the machine learning process is shallow and do not consider the context relationship between the words in the text or just based on the simple rules, so they are problematic for noisy text.

Automatic text classification (ATC) is a guided learning process, which classifies a large number of unstructured text data into specific categories according to the given classification system and the content of text information [11–13], which is widely used in sentiment classification [14], spam filtering [15], and Web searching [16]. CNN uses relatively little preprocessing compared to traditional

classification models. This is because the network can learn the filters that were hand-crafted in a traditional classification framework. CNN can be used to learn features as well as to classify data.

Text feature representation methods used frequently are BoW [17], TFIDF [18], and LDA (Latent Dirichlet Allocation) [19]. In this paper, we propose a subword and stem vector-based Uyghur and Kazakh short text classification method. We used a word (stem) embedding method for extraction of text features, and used a TFIDF algorithm to weight the feature vectors to better represent the Uyghur and Kazakh texts, and then used CNN as a feature selection and text classification algorithm to obtain a Uyghur and Kazakh text classification model, and conducted a classification experiment on a corpora collected from the Internet.

2. Proposed Uyghur and Kazakh Text Representation and Classification Method

The proposed text classification system in this paper consists of two steps. Step one is the preprocessing of Uyghur and Kazakh text data, which includes the establishment of the text corpus, morpheme segmentation, and extracting the stems. Step two is the classification process, which includes the extraction of features and classification of texts.

2.1. Term Representation

Along with the rise of neural networks, a large number of neural network models suitable for natural language have been proposed [20,21]. Bengio et al. [20] proposed a neural network-based language model construction method in 2003. Mikolov et al. [21] proposed word2vec algorithm in 2013, and by describing the representation of a word through text context information, got low-dimensional dense vectors, which can represent semantic relationships between words.

2.1.1. Word Vector

Word (stem) embedding [22] is a technique to map words or word units to vectors of real values, and the similarity of any two stems can quickly be determined by calculating the distance between their stem vectors. Word2vec tool enabled us to obtain stem vectors conveniently. There are two submodels in the word2vec framework, which are the CBOW (Continuous Bag Of Words) model [23] and the skip-gram model [24], respectively.

CBOW is a model to predict the probability of occurrence $p(s_t | s_{t-c}, s_{(t-c)-1}, \dots, s_{t-1}, s_{t+1}, s_{t+2}, \dots, s_{t+c})$ of a stem, s_t , given the context stems $s_{t-c}, s_{(t-c)-1}, \dots, s_{t-1}, s_{t+1}, s_{t+2}, \dots, s_{t+c}$. In this model, a stem is represented by c stems before and after that stem, c is the size of the preselected window, the output is the stem vector for this feature stem s_t , as shown in Figure 1. We used the CBOW method for training of stem vectors.

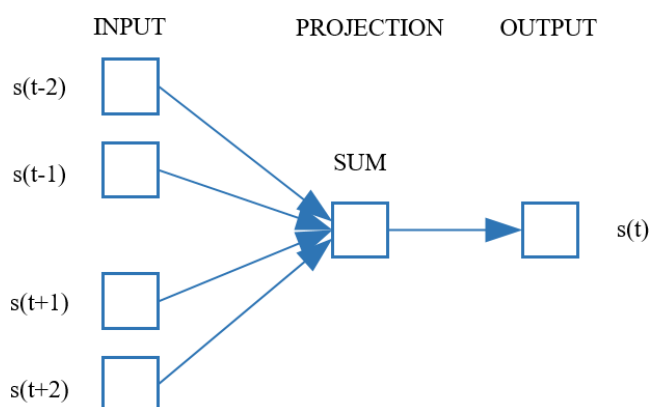


Figure 1. CBOW (Continuous Bag Of Words) model.

The idea of the skip-gram model is exactly the opposite of the CBOW model, in that it predicts the occurrence probability $p(s_{t-c}, s_{(t-c)-1}, \dots, s_{t-1}, s_{t+1}, s_{t+2}, \dots, s_{t+c} | s_t)$ of context stems $s_{t-c}, s_{(t-c)-1}, \dots, s_{t-1}, s_{t+1}, s_{t+2}, \dots, s_{t+c}$, given the particular stem, s_t , as shown in Figure 2.

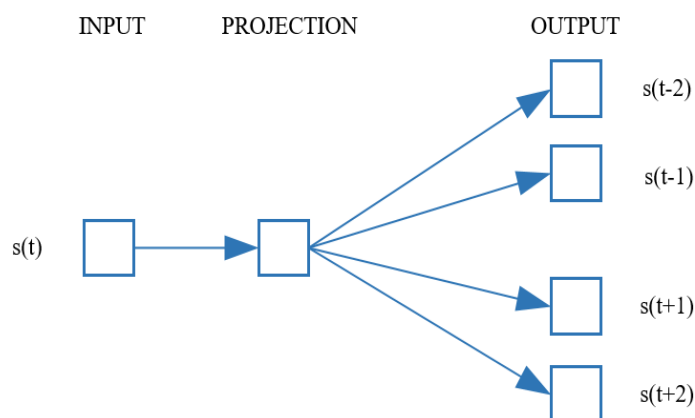


Figure 2. Skip-gram model.

The stem vector obtained by word2vec training can determine the degree of semantic similarity between stems by the cosine distance. The larger the calculated cosine value, the closer the semantics are; otherwise, the opposite is true, as shown in Table 3.

Table 3. Stem vector semantic similarity.

Uyghur Word Stem Muzika (Music) Related Stems		Kazakh Word Stem Vaqxa (Coin) Related Stems	
stem	cosine distance	stem	cosine distance
vusul(dance)	0.8138	qarez(loan)	0.9206
sAnvAt(art)	0.7970	pul(money)	0.8783
naHxa(song)	0.7742	bankE(bank)	0.8704
gitar(guitar)	0.7413	somasen(amount)	0.8694
klassik(classic)	0.7281	qaytarew(repayment)	0.8664

From Table 3 we can see that the Uyghur stem muzika (music) and Kazakh stem vaqxa (coin) are input, respectively, and the five stems which have the closest semantics to the two input stems are obtained by calculating the cosine distance between stem vectors.

2.1.2. Weighting Term Vector by TFIDF

For set D with M texts, where $D_i (i = 1, 2, \dots, M)$, the stem vector is obtained by the CBOW model. For each stem in a text, apply $TF - IDF$ algorithm to get the $tfidf(t, D)$ value, which refers to the weight value of stem t in text $D_i (i = 1, 2, \dots, M)$.

The stem vector of each stem is weighted by $tfidf$ value to represent a text, as shown in Equation (1):

$$vec(D_i) = \sum_{t \in D_i} w_t \cdot tfidf(t, D_i) \tag{1}$$

Here, $vec(D_i)$ refers to the stem vector of each text D_i , w_t represents N -dimensional stem vector of stem t , and $tfidf(t, D_i)$ represents the $TF - IDF$ weight value of stem t in text D_i .

2.2. CNN Model Architecture

CNN is a deep learning model, which was proposed by Lecun et al. [25], and Kim [8] first used CNN in a text classification task. CNN can automatically extract and learn the feature of the sentence on the basis of the stem vector, thus reducing the dependence on the manual selection of features and optimizing the effect of feature selection. The architecture of the CNN model proposed in this paper is shown in Figure 3. Our CNN model consisted of four different layers: Input layer, two convolution layers, two pooling layers, and a fully connected layer, respectively.

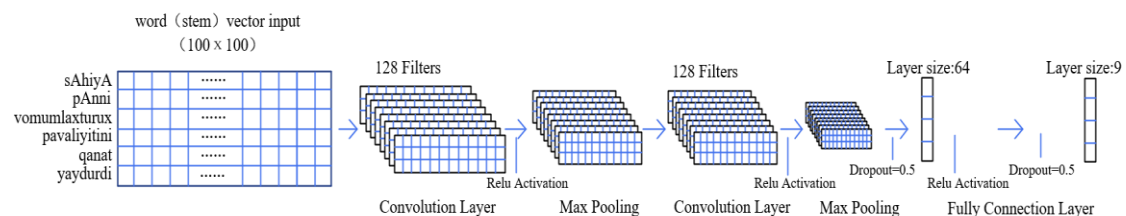


Figure 3. Convolutional neural network (CNN) model architecture proposed in this paper.

(1) Input layer. The first layer of CNN is the input layer, in which the input parameters are the stem vectors that we obtained after pretraining the texts.

(2) Convolution layers, which are the core components of CNN. These layers convolve feature maps of a previous layer in the network with convolution kernels to generate new features.

(3) Pooling layer. The input of this layer is the feature matrix generated in the convolution layer. The function of the pooling layer is to sample the feature map which is generated by the convolution layer.

(4) Fully connected layer, which is the last layer in CNN, connects all the features and output values to the classifier.

For the text set $D_i (i = 1, 2, \dots, M)$, we got text vectors $vec(D_i)$ after training the texts by using the CBOW algorithm, then we weighted them by using the $tfidf$ value of each stem. Then we let all the text vectors obtained by the previous steps have the necessary input matrix for CNN processing by modifying them. The input text of CNN can be expressed as shown in Equation (2). Here, $T_{1:m}$ represents all the input texts, and \oplus is the concatenation operator.

$$T_{1:m} = vec(D_1) \oplus vec(D_2) \oplus \dots \oplus vec(D_m) \quad (2)$$

2.3. Robust Uyghur and Kazakh Morpheme Segmentation

For improving the minority language NLP tasks, we developed a compact tool [2]. This tool can segment the sequences of words in Uyghur and Kazakh texts into the sequences of morphemes.

Based on aligned word and morpheme parallel training data, this program will automatically learn the various surface forms and acoustic rules from training data. When the morphemes were merged to a word, the phonemes on the boundaries changed their surface forms according to the phonetic harmony rules. Morphemes will harmonize each other, and appeal to each other's pronunciation. When the pronunciation is precisely represented, the phonetic harmony can be clearly observed in the text. A segmentation program will export all possible segmentation forms for each candidate word. An independent statistical model can be incorporated to select the best result from N-best results. This toolkit provided a reliable basis for morpheme segmentation and stem extraction and greatly improved the short text classification task of Uyghur and Kazakh.

We used this tool to train a statistical model using word-morpheme parallel training corpus, which included 10,025 Uyghur sentences and 5000 Kazakh sentences, and we selected 80% of them as the training corpus. The rest was used as the testing corpus to carry out morpheme segmentation and stem extraction experiments, as shown in Figure 4. The highest accuracy of stem extraction on two data sets obtained were 97.66% and 95.87%, respectively, which were the percentage of exact match

of all morphemes obtained by automatic segmentation with morpheme of manual segmentation, as shown in Table 4.

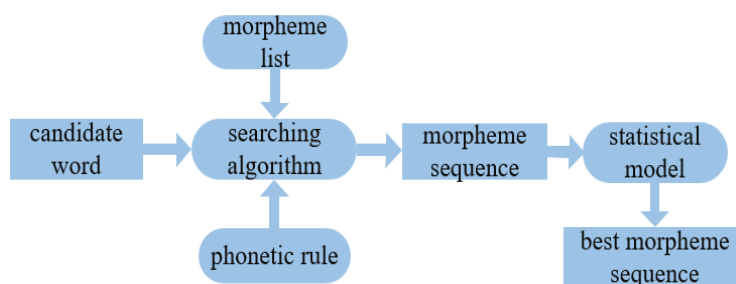


Figure 4. Morpheme segmentation flow chart.

Table 4. Results of rule-based morpheme segmentation and stem extraction.

Corpus Size (word/k)	Corpus	Accuracy (%)	Morpheme Coverage (%)	Word Coverage (%)
1.91	Uyghur	93.47	82.30	38.93
	Kazakh	89.09	77.31	30.15
7.49	Uyghur	96.59	91.88	60.94
	Kazakh	93.89	86.91	46.94
32.74	Uyghur	97.45	95.96	71.43
	Kazakh	95.06	94.19	65.72
64.51	Uyghur	97.52	96.11	74.35
	Kazakh	95.87	96.21	73.87
129.4	Uyghur	97.66	98.44	86.85
	Kazakh	-	-	-

Table 5 shows some ambiguous examples which can only be disambiguated by morphological analysis within a longer context, like a sentence, and intraword-based stemming methods are unable to extract stems reliably.

Table 5. Ambiguous stems (Uyghur).

Variants(English)	Variants	Suffix
vAl(people)/val(get)	vAl/val → el + iN	iN
person’s name/lucky	qUt → qUtlUq, qUt + IUq	IUq
fire/grass	Vot → vot + tAk, vot + laq	tAk, laq

Usually, the number of words in texts are not the same. Therefore, we use padding to modify the text length to let all the texts have the same word length, which can produce the required input matrix for CNN. We made statistics about the number of words in each raw text in our corpus, and found that the number of words in the experimental text corpus tend to range from 40 to 120 words, and most of the texts had about 100 words. Hence, we select 100 as a standard word length of the text corpus for CNN. We filled in the texts with less than 100 words with zero. For the corresponding morpheme sequence texts, after extracting the stems, the first 100 stems were selected for each text as the input of CNN. Similarly, the texts with less than 100 stems were filled with zero to get the text matrix required for CNN input.

3. Experimental Results and Analysis

At present, the research on the classification of Uyghur and Kazakh text is at its initial stage. There are no publicly available standard and open source text corpus. Therefore, we had to build Uyghur and Kazakh text corpus for our experiment by crawling the Internet.

3.1. Experimental Setup

We collected our text corpora by using Web crawler technique and downloaded from official Uyghur and Kazakh Web sites with Arabic script such as www.uyghur.people.com.cn and www.Kazakh.ts.cn. Our corpus included nine categories of Uyghur texts: Law, finance, sports, culture, health, tourism, education, science, and entertainment. Each category contained 900 texts, for a total of 8100 texts. Our corpus also included eight categories of Kazakh texts: Law, finance, sports, culture, tourism, education, science, and entertainment. Each category contained 900 texts, for a total of 7200 texts. (The names of the eight categories in the Uyghur and Kazakh corpora are the same, but the text content in the same-named category in the two corpora was not the same.) We used 75% of them as a training set, used 10% as a validation set, and used the rest as a test set. We used the pytorch to implement the CNN framework on a Linux CentOS operating system with GPU (Graphics Processing Unit) support. The resulting loss and accuracy rates were recorded.

After normalizing all the texts with various coding in the original corpus into standard Roman code, we segmented them to morpheme sequence and extracted stems by using morpheme segmentation tool. The stem-affix based subword extraction method gave a very good result in the reduction of feature space, in that the morpheme vocabulary reduced greatly to less than 31% of the word vocabulary, as shown in Tables 6 and 7. We can see that the accumulation of morphemes was also only one-third of the accumulation of words when the number of classes and volume of corpus were increasing.

Table 6. Reduction in feature space dimension by stem extraction (Uyghur).

Corpus	# of Class	Word Vocabulary	Morpheme Vocabulary	Morpheme-Word Vocabulary Ratio (%)
Uyghur	5	55,165	18,148	32.8
	7	67,924	21,474	31.6
	9	79,762	24,643	30.8

Table 7. Reduction in feature space dimension by stem extraction (Kazakh).

Corpus	# of Class	Word Vocabulary	Morpheme Vocabulary	Morpheme-Word Vocabulary Ratio (%)
Kazakh	4	49,018	15,238	31.1
	6	60,826	18,439	30.3
	8	72,152	21,715	30.1

After a robust morpheme segmentation and stem extraction, the stem vector of all corpora was trained using the CBOW model based on the hierarchical softmax algorithm. During training, the training window size was set to 5, and the learning rate was 0.025. After obtaining stem vectors, the TFIDF algorithm was used to weight them. In order to confirm the optimal dimension of stem embedding in this classification task, we experimented with the different stem embedding dimensions range from 50 to 300 in steps of 50. The effect of different stem embedding dimensions on the classification accuracy is shown in Figure 5.

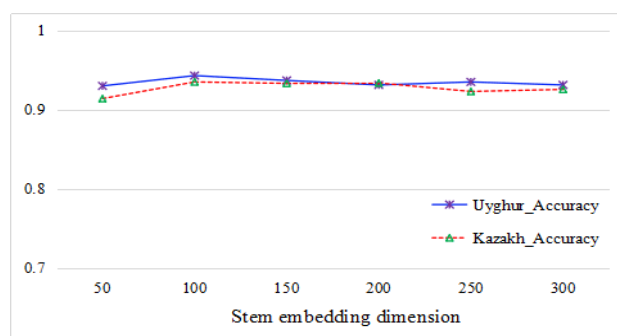


Figure 5. Effect of stem embedding dimension on accuracy.

From Figure 5 we can see that the classification accuracy was the highest when the vector dimension was around 100, and the increase in the stem vector dimension hardly influenced the accuracy, so we selected 100 as a dimension of stem vectors in our experiments.

3.2. Evaluation Indicator

Accuracy, precision, recall, and F1-Measure are frequently used evaluation methods for text classifier performance. Precision and recall evaluate the quality of feature extraction items, and F1-Measure is the method which is combined by precision and recall. The calculations for them are shown below, where

accuracy = number of texts classified accurately/total number of texts

precision = number of texts classified accurately into class C_k /total number of texts actually categorized into class C_k

recall = number of texts accurately categorized into class C_k /total number of texts in category C_k

$F1 = 2 \times \text{precision} \times \text{recall} / (\text{precision} + \text{recall})$.

In this paper, we use accuracy and macro-F1 value to evaluate the performance of a proposed text classification approach. Macro-F1 is a global F1 indicator, in that the F1 value is calculated for each class respectively, and the arithmetic mean of them is then used as the global index.

3.3. Test Results and Analysis

3.3.1. Comparison of Accuracy on Different Word Units

We segmented the texts into morpheme sequences and selected 100 stem units, and selected the top 100 word units from the texts without segmentation as the input for CNN separately. We used word2vec algorithm to extract 100×100 word and stem vectors from all texts, and used the TFIDF algorithm to weight them, then input them into CNN, and conducted classification experiments based on word and morpheme units, respectively. We compared the word-based results with the morpheme-based results.

In this paper, several CNN model structures including 2, 4, and 6 convolution layers were tested. From these experiments, we found that for the task of text classification proposed in this paper the best CNN model structure consisted of two groups of convolution layers, each of which was followed by a max pooling layer. Through repeated experiments, we determined that 128 convolutional filters with a filter size of 5×100 on each convolution layer had the best classification effect. After the second max pooling layer, a dropout function was used to avoid overfitting problems and the dropout value was set to 0.5. Then, a full connection layer with a length of 64 was added, followed by a second dropout function, then followed by a dense layer with a size of 9 for Uyghur texts and 8 for Kazakh texts to represent the number of categories with a softmax function determining the output, as shown in Figure 3.

CNN obtained weights by iterative calculations and got the ideal parameters after several times of iterations. In this experiment, 150 times of iterations were performed. The experimental results are shown in Figures 6 and 7.

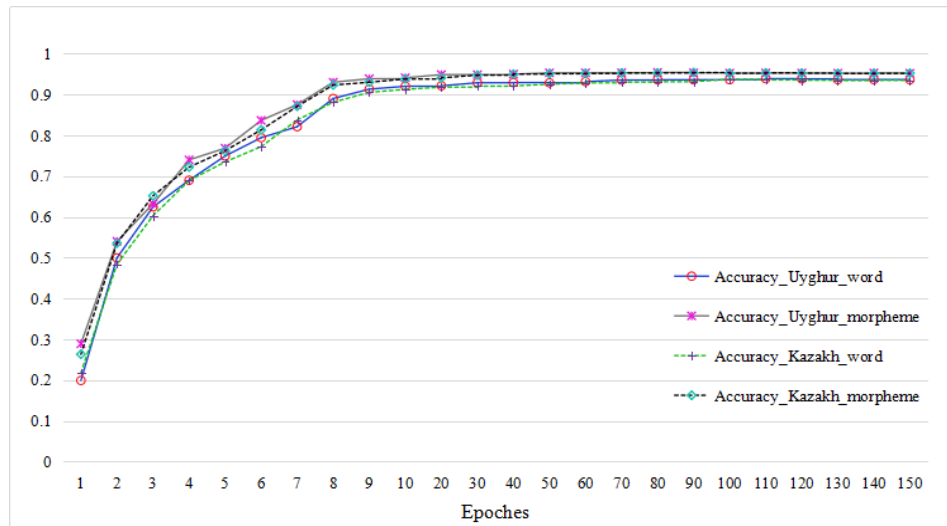


Figure 6. Accuracy based on different word units.

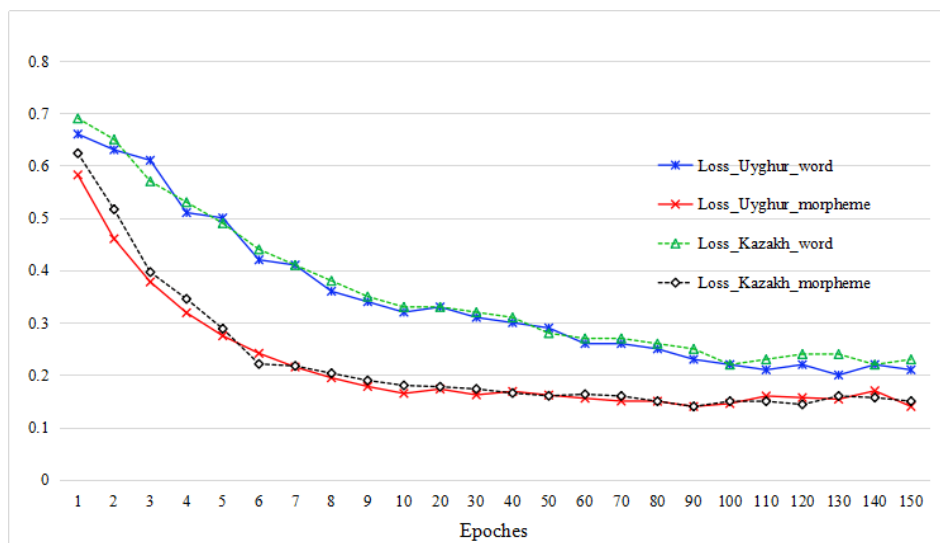


Figure 7. Loss based on different word units.

As can be seen from Figures 6 and 7, the classification accuracy based on both word and morpheme units on two data sets increased with the rise of iteration time, but the morpheme-based accuracy on both data sets was bigger than that of word-based accuracy in all the ranges of iteration times. After 9 times of iterations the accuracy of both word- and morpheme-based approaches exceeded 90%, and reached 91.33% and 90.55% on Uyghur and Kazakh data sets with word units, and to 93.88% and 93.02% on Uyghur and Kazakh data sets with morpheme units, respectively. Around 80–90 times of iterations, the accuracy based on morpheme units on two data sets reached to the highest point of 95.43% and 95.39%, and then tended to converge around 95.3% and 95.2%, respectively, which were much bigger than that of the highest values with the word-based approach, which accounted for 93.90% and 93.71% on two data sets, respectively. The loss on two data sets decreased with the rise of iteration times, but the speed of decrease was much faster on morpheme units than that of word units. The loss

rate for both data sets on morpheme units converged around 14%, which was much lower than that of word units with around 22% of loss rate. These results prove that the morpheme unit is a much better option than word units in text classification task of agglutinative languages like Uyghur and Kazakh.

3.3.2. Comparison of Proposed Classification Method with Other Methods

In order to prove the superiority of the CNN model in text classification tasks, we used several commonly used machine learning models, namely KNN, RF (Random Forest), NB (Naive Bayes), LR (Logistic Regression), and SVM, to perform classification experiments on our corpus. We used w2vec_TFIDF fusion features to represent morpheme sequence text. For traditional classifiers we used chi-2 feature selection method to reduce original feature space dimensions, and selected from 100 to 2000 feature dimensions to perform classification experiments. Comparative results are shown in Table 8.

Table 8. Classification results based on different classification methods.

Methods	Corpus	Accuracy (%)	Macro-F1
KNN	Uyghur	86.12	0.86
	Kazakh	85.49	0.85
RF	Uyghur	84.56	0.83
	Kazakh	84.69	0.83
NB	Uyghur	92.47	0.91
	Kazakh	92.07	0.90
LR	Uyghur	88.23	0.87
	Kazakh	87.69	0.85
SVM	Uyghur	93.81	0.92
	Kazakh	93.64	0.91
CNN	Uyghur	95.43	0.97
	Kazakh	95.39	0.97

As can be seen from Table 8, among the traditional machine learning models, the SVM achieved the best performance with an accuracy rate of 93.81% and 93.64% on two data sets, respectively. This is related to the training goal of the SVM model pursuing structural risk minimization, which reduces the requirement of data size and data distribution. Therefore, the best performance among traditional methods was obtained under the small sample conditions in this paper. The CNN model outperformed all the other traditional machine learning methods on both data sets in terms of accuracy and F1 score, and the CNN model-based accuracy rate was still better than the best results with SVM for 1.62% and 1.75% on two data sets, respectively. These results verify that CNN is superior to other traditional models in text classification tasks.

3.3.3. Effects of Different Text Representation Methods on Accuracy

In order to verify the performance of word2vec_TFIDF fusion feature representation in text classification tasks, we performed experiments based on the following methods and compared the results with our model: CNN + rand, CNN + word2vec, and CNN + word2vec_LDA. For all the experiments below we still carried out 150 times of iterations for fair comparison.

CNN + rand: The network architecture of the CNN model remained unchanged, but there was no pretraining of stems to represent text features, and the distributed features of the network's inputs were randomly initialized by Gaussian distribution.

CNN + word2vec: The network architecture of the CNN model remained unchanged, but word2vec algorithm was used to pretrain stem vectors to represent text features, and stem vectors obtained by the training were input to CNN.

CNN + word2vec_LDA: The network architecture of the CNN model remained unchanged, but word2vec algorithm was used to pretrain stem vectors and the LDA algorithm was used to train the topic model on stem sequence texts. Then, by superimposing the matrices obtained from the two models, a new feature matrix was formed and input to the CNN.

The classification results based on the methods mentioned above are shown in Table 9.

Table 9. Classification results based on different text representation methods.

Methods	Corpus	Accuracy (%)	Macro-F1	Loss
CNN + rand	Uyghur	91.64	0.92	0.35
	Kazakh	91.23	0.93	0.33
CNN + word2vec	Uyghur	94.98	0.96	0.17
	Kazakh	94.95	0.95	0.18
CNN + word2vec_LDA	Uyghur	95.21	0.96	0.16
	Kazakh	95.13	0.96	0.18
CNN + word2vec_TFIDF	Uyghur	95.43	0.97	0.15
	Kazakh	95.39	0.97	0.14

As can be seen from Table 9, the word2vec_TFIDF fusion feature representation method proposed in this paper outperformed the other text representation methods mentioned above in terms of accuracy, Macro-F1 value, and loss. Compared with the CNN + rand method, the CNN + word2vec method introduced word embedding to represent words, and then obtained a distributed representation of the entire short text, which increased the classification accuracy rate from 91.64% and 91.23% to 94.98% and 94.95% on two data sets. The word2vec_LDA features gave better results than just the plain word2vec text representation method, but because there was a weak correlation between the components of the Dirichlet distributed random vectors, some potential topics tended to seem irrelevant in this method; actually there was maybe a strong relationship among them. Therefore, its performance was not better than that of the word2vec_TFIDF method. The word2vec_TFIDF fusion feature representation method not only realized the distributed representation of text features, but also better took into account the impact of a given single stem on whole texts by using TFIDF weighting, so it obtained much better classification results.

4. Conclusions

Uyghur and Kazakh are morphologically rich agglutinative languages in which words are formed by a stem attached to several suffixes, and this property allows infinite vocabulary, in theory. Suffixes provide semantic and syntactic functions. So stem extraction and morphological analysis are the efficient way of NLP. Word embedding techniques can map language units into a sequential vector space based on context. It is a natural way to extract and predict OOV from context information. In this paper, we discuss a text classification method based on morpheme embedding with *tfidf* weighting and neural network architecture. Based on the CNN model, Uyghur and Kazakh text classification tasks were implemented on both word and morpheme units separately. The experimental results based on morpheme segmentation showed an improved performance for morpheme units compared to word units.

Author Contributions: Conceptualization, A.H. and M.A.; Methodology, S.P. and M.A.; Software, M.A. and S.P.; Validation, A.H., M.A. and S.P.; Formal analysis, S.P. and M.A.; Investigation, S.P.; Resources, A.H.; Data curation, S.P., M.A. and A.H.; Writing—original draft preparation, S.P.; Writing—review and editing, A.H., M.A. and S.P.; Visualization, S.P.; Supervision, A.H.; Project administration, A.H.; Funding acquisition, A.H.

Funding: This research was funded by National Natural Science Foundation of China, grant number 61462085, 61662078, 61633013, and by National Key Research and Development Plan of China, grant number 2017YFC0820603.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ablimit, M.; Kawahara, T.; Pattar, A.; Hamdulla, A. Stem-Affix based Uyghur Morphological Analyzer. *Int. J. Future Gener. Commun. Netw.* **2016**, *9*, 59–72. [[CrossRef](#)]
2. Ablimit, M.; Parhat, S.; Hamdulla, A.; Zheng, T.F. A multilingual language processing tool for Uyghur, Kazak and Kirghiz. In Proceedings of the 2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference, Kuala Lumpur, Malaysia, 12–15 December 2017; pp. 737–740.
3. Hamdulla, A. An acoustic parametric Database for Uyghur Language. In Proceedings of the 2009 International Joint Conference on Artificial Intelligence, Hainan Island, China, 25–26 April 2009; pp. 405–408.
4. Tuerxun, P.; Dingyi, F.; Hamdulla, A. The KNN based Uyghur Text Classification and its Performance Analysis. *Int. J. Hybrid Inf. Technol.* **2015**, *8*, 63–72. [[CrossRef](#)]
5. Wang, P.; Xu, J.; Xu, B.; Liu, C.; Zhang, H.; Wang, F.; Hao, H. Semantic clustering and convolutional neural network for short text categorization. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, Beijing, China, 26–31 July 2015; pp. 352–357.
6. Johnson, R.; Zhang, T. Effective use of word order for text categorization with convolutional neural networks. *arXiv* **2014**, arXiv:1412.1058.
7. Zhu, M.; Yang, X. Chinese Texts Classification System. In Proceedings of the 2019 IEEE 2nd International Conference on Information and Computer Technologies, Kahului, HI, USA, 14–17 March 2019; pp. 149–152.
8. Kim, Y. Convolutional Neural Networks for Sentence Classification. *arxiv* **2014**, arXiv:1408.5882.
9. Imam, S.; Parhat, R.; Hamdulla, A.; Li, Z. Performance analysis of different keyword extraction algorithms for emotion recognition from Uyghur text. In Proceedings of the 9th International Symposium on Chinese Spoken Language Processing, Singapore, Singapore, 12–14 September 2014; pp. 351–353.
10. Yergesh, B.; Bekmanova, G.; Sharipbay, A.; Yergesh, M. Ontology-Based Sentiment Analysis of Kazakh Sentences. In *International Conference on Computational Science & Its Applications*; Springer: Berlin/Heidelberg, Germany, 2017; Volume 10406, pp. 669–677.
11. McCallum, A.; Nigam, K. Text classification by bootstrapping with keywords, EM and shrinkage. In Proceedings of the ACL 1999-Workshop for Unsupervised Learning in Natural Language Processing, Washington, WA, USA, 21 June 1999; pp. 51–58.
12. Zhang, Y.; Zincer-Heywood, N.; Milios, E. Narrative text classification for automatic key phrase extraction in web document corpora. In *Proceedings of the 7th Annual ACM International Workshop on Web Information and Data Management*; ACM: New York, NY, USA, 2005; pp. 52–58.
13. Sebastiani, F. Machine learning in automated text categorization. *ACM Comput. Surv.* **2002**, *34*, 1–47. [[CrossRef](#)]
14. Maas, A.L.; Daly, R.E.; Pham, P.T.; Huang, D.; Ng, A.Y.; Potts, C. Learning word vectors for sentiment analysis. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Portland, OR, USA, 19–24 June 2011; Volume 1, pp. 142–150.
15. Bing, Z.; Yao, Y.Y.; Luo, J. Cost-sensitive three-way email spam filtering. *J. Intell. Inf. Syst.* **2014**, *42*, 19–45.
16. Aggarwal, C.C.; Zhai, C. *A Survey of Text Classification Algorithms, Mining Text Data*; Springer: New York, NY, USA, 2012; pp. 163–222.
17. Wallach, H.M. Topic modeling: Beyond bag-of-words. In Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh, PA, USA, 25–29 June 2006; pp. 977–984.
18. Hu, J.; Yao, Y. Research on the Application of an Improved TFIDF Algorithm in Text Classification. *J. Conver. Inf. Technol.* **2013**, *8*, 639–646.

19. Pavlinek, M.; Podgorelec, V. Text classification method based on self-training and LDA topic models. *Expert Syst. Appl.* **2017**, *80*, 83–93. [[CrossRef](#)]
20. Bengio, Y.; Ducharme, R.; Vincent, P.; Jauvin, C. A neural probabilistic language models. *J. Mach. Learn. Res.* **2003**, *3*, 1137–1155.
21. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed representation of words and phrases and their compositionality. *Adv. Neural Inf. Process. Syst.* **2013**, *26*, 3111–3119.
22. Lai, S.; Xu, L.; Liu, K.; Zhao, J. Recurrent convolutional neural networks for text classification. In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, Austin, TX, USA, 25–30 January 2015; Volume 333, pp. 2267–2273.
23. Goldberg, Y.; Levy, O. Word2vec Explained: Deriving Mikolov et al.’s negative-sampling word-embedding method. *arxiv* **2014**, arXiv:1402.3722.
24. Chen, Y.Q. Implementing the k-nearest neighbour rule via neural network. In Proceedings of the ICNN’95-International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; Volume 1, pp. 136–140.
25. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2323. [[CrossRef](#)]



© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).