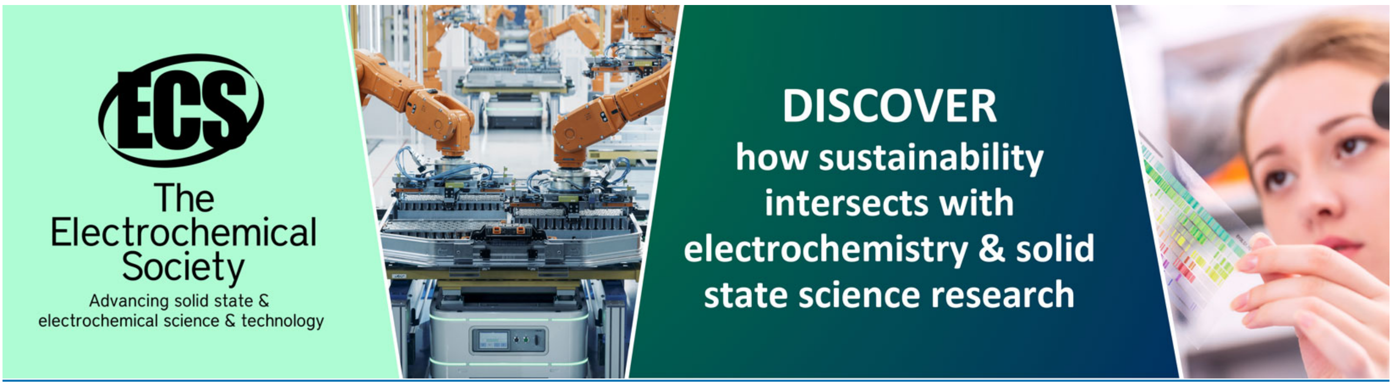
Study on Tibetan Word Vector based on Word2vec


ECS
The Electrochemical Society
Advancing solid state & electrochemical science & technology
DISCOVER how sustainability intersects with electrochemistry & solid state science research

Study on Tibetan Word Vector based on Word2vec

Ning Yang[1], Guanyu Li[1*], Hailan Ding[1], Chunwei Gong[1]

[1]Key Laboratory of National language Intelligent Processing Gansu Province, Northwest Minzu University, Lanzhou, China

*guanyu-li@163.com

Abstract. This paper uses Word2vec to study Tibetan word vector. Word2vec is optimized by two methods: Hierarchical Softmax and Negative Sampling in CBOW and Skip-gram models. Through the training of neural network, the words in Tibetan sentences are converted into vector form. Word2vec transforms the Tibetan text content processing into a simple vector space operation, calculates the similarity in the vector space, and then obtains the semantic similarity of the text, providing an accurate word vector for the training of the language model.

1. Introduction
Speech recognition is an important field of artificial intelligence research, and the quality of the language model to a great extent determines the accuracy of speech recognition in the language model training, which requires the vectorization of words in a Tibetan text corpus in advance compared to the traditional one-hot word vector.Word2vec is a distributed representation in the form of a word vector, according to the relationship of context to define the term vectors, each of those dimensions is real Numbers in a row, and high correlation of the word has a closer distance. Compared with Word2vec, one-hot word vector has many problems, such as dimension disaster, data sparse, the isolation of semantics and the difficulty of fuzzy matching, etc., with stronger expressive force and more internal features of corpus data.

Tibetan is a very important minority language in China. According to incomplete statistics, more than 6.4 million people use Tibetan. This research can provide theoretical and data support for Tibetan phonetic teaching, phonetic technology engineering application, and perfect the research of Tibetan phonetic direction. At the same time, strengthening the study of Tibetan pronunciation can not only effectively solve the problem of language barriers between Tibetan areas and other regions of China, which is promoting inter-ethnic exchanges and mutual understanding among ethnic groups, but also promote the Tibetan area's economy, science and technology.

2. Word2vec structural model
This section introduces the word vector in distributed representation form used by Word2vec, which introduces the use and optimization of Word2vec. Finally, the specific flow and structure of Word2vec's CBOW model and Skip-Gram model are introduced, as well as the performance comparison between the two models.

2.1. The distributed representation word vector
Distributed representation putted forward by Hinton as early as in 1986, by training every word map into a fixed length of ‘short’ vector (as opposed to one-hot vector ‘long’), and all these constitute a word vector space, and each vector can be seen as a point in the space. The introduction of ‘distance' on the

space, and then judged by the distance between the word and the word on the lexical and semantic similarity. Distributed representations of words in a vector space help learning algorithms to achieve better performance in a natural language processing task by grouping similar words[1]. Word2vec uses this vector representation of the distributed representation. It is a low-dimensional real number vector in the form: [0.832, - 0.169, -0.397, 0.339, -0.894...]. Word vectors are widely used, including computing similarity (searching for similar words, information retrieval), which is the input of SVM/LSTM and other models (Chinese word segmentation, nomenclature recognition), sentence representation (affective analysis), document representation (document subject identification) and so on.

2.2. Introduction to Word2vec

Word2vec is a tool that Google has open sourced in 2013 to express words as real-value vectors. The models are CBOW and Skip-Gram.Word2vec uses a one-layer neural network to map the word vector in one-hot form to the word vector in distributed representation form. Through training, the Tibetan text content processing is simplified to vector operation in k-dimensional vector space, and the similarity degree in vector space can be used to express the similarity of text semantics. Therefore, the word vector output is used to do a lot of NLP related work, such as the Tibetan language clustering studied in this paper. Our technique is inspired by the recent work in learning vector representations of words using neural networks[2]. The words processed by Word2vec can be used to add, subtract, multiply and divide vectors so that the expression of words can be vectorized, and the similarity between them can be calculated more easily.

Word2vec's training algorithm is a vector corresponding to a word. That is, there is only a single interpretation (one-to-one mapping) of the semantic attribution of words. Word2vec's analysis of semantics is not based on common sense, nor is it based on the grammatical rules of natural languages to construct models, but on the basis of the rules of lexical statistics to make a statistical hash of lexical units. Actually according to the result of clustering proceeds to directly obtain one kind of classification rule. So Word2vec vector's clusters proceed from the actual semantic orientation and the analysis of inner value, which requires the expectation for further analysis and research on this field to arrive at accurate conclusions.

2.3. Word2vec model structure

Word2vec is an intermediate product of the neural probabilistic model, which is a byproduct of the neural network to learn a language model. Specifically, a language model refers to CBOW and Skip-gram. And two language model specific learning process method of approximation of two will be used to reduce the complexity of the Hierarchical Softmax and Negative Sampling.

2.3.1 Continuous word bag model CBOW. The CBOW model predicts the central words according to the words around the central word y_j, while the Skip-Gram model predicts the surrounding words according to the central word x_k. Below are the CBOW model structure diagram and Skip-Gram model structure diagram respectively, as shown in Figure 1.

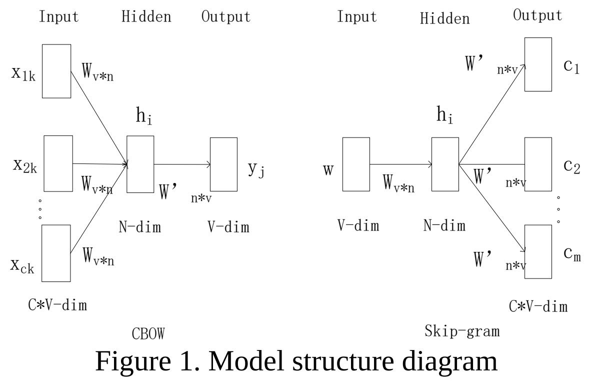


Figure 1. Model structure diagram

The CBOW predicts the current word based on the context, and the Skip-Gram predicts surrounding words given the current word[3]. Contrary to CBOW, Skip-Gram model is based on the input at the center of the word to predict around the word.

- Input layer: Enter the one-hot coding of the context word around the target word, where the window size is C and the vocabulary size is V.
- Hidden layer: The vector sum averaging vector as the hidden layer, the size is 1 * N and dimensions for N.
- Be one-hot coding input vector through a weight matrix W, the dimensions of W is V * N which connected to the hidden layer.
- Output layer: Be one-hot code word y

Hidden layer by the weight of a , N * V matrix W ', connected to the output layer. Finally, vector 1*v is obtained. After the activation function softmax normalization, the v dimension probability distribution is obtained. The word with the highest probability is the predicted intermediate word y.

In the Skip-Gram model, we are given a corpus of words w, and their contexts c_k, c is 1 to m. We consider the conditional probabilities is p(c|w) and given a corpus Text, and the goal is to set the parameters θ of p(c|w; θ), so as to maximize the corpus probability in formula (1):

$$\arg\max_{\theta} \prod_{w \in Text} \left[\prod_{c \in C(w)} p(c \mid w; \theta) \right] \qquad (1)$$

C(w) is the set of contexts of word w. Alternatively as shown in formula (2):

$$\arg\max_{\theta} \prod_{(w,c) \in D} p(c \mid w; \theta) \qquad (2)$$

D is the set of all word and context pairs we extract from the text[4].

2.4. CBOW model process example

It is simple to assume that the present corpus is a document with only four words: { དགེ། བཀུར སློབ གཅེས }, the target word is སློབ, the window size is 2. According to the word དག།, བཀུར and གཅེས aim to predict a central word སློབ. And what we want to predict is that the word སློབ, The Specific processes are shown in Figure 2.

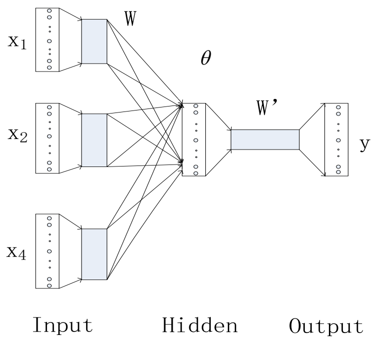


Figure 2. CBOW model

Example corpus = { དགེ། བཀུར སློབ གཅེས }

Window size: 2

Target word: སློབ

Context word: དགེ། , བཀུར , གཅེས

- *x_1: $W^{\text{དགེ}}$ = [1,0,0,0]*

  x_2: $W^{\text{བཀུར}}$ = [0,1,0,0]

  Label is x_3: $W^{\text{སློབ}}$ =[0,0,1,0]

  x_4 : $W^{\text{གཅེས}}$ = [0,0,0,1]

- Initialize:

$$w=\begin{bmatrix}1 & 2 & 3 & 0\\ 1 & 2 & 1 & 2\\ -1 & 1 & 1 & 1\end{bmatrix}$$

Ex：W $_{\text{བཀར}}$ *= [0,1,0,0]*

Wx_2 = v_2

$$\begin{bmatrix}1 & 2 & 3 & 0\\ 1 & 2 & 1 & 2\\ -1 & 1 & 1 & 1\end{bmatrix}\begin{bmatrix}0\\1\\0\\0\end{bmatrix}=\begin{bmatrix}2\\2\\1\end{bmatrix}$$

- $$\frac{V1+V2+V4}{3}=\theta$$

$$\frac{1}{3}\left(\begin{bmatrix}1\\1\\-1\end{bmatrix}+\begin{bmatrix}2\\2\\1\end{bmatrix}+\begin{bmatrix}0\\2\\1\end{bmatrix}\right)=\begin{bmatrix}1\\1.67\\0.33\end{bmatrix}$$

- Initialize:

$$W'=\begin{bmatrix}1 & 2 & -1\\ -1 & 2 & -1\\ 1 & 2 & 2\\ 0 & 2 & 0\end{bmatrix}$$

$$W'\theta=U_0$$

$$\begin{bmatrix}1 & 2 & -1\\ -1 & 2 & -1\\ 1 & 2 & 2\\ 0 & 2 & 0\end{bmatrix}\begin{bmatrix}1.00\\1.67\\0.33\end{bmatrix}=\begin{array}{c|l}\begin{bmatrix}4.01\\2.01\\5.00\\3.34\end{bmatrix} & \begin{array}{l}\text{U1}\\ \text{U2}\\ \text{U00}\\ \text{U4}\end{array}\end{array}$$

- Softmax(U_0)=y

$$soft\max\left(\begin{bmatrix}4.01\\2.01\\5.00\\3.34\end{bmatrix}\right)=\begin{bmatrix}0.23\\0.03\\0.62\\0.12\end{bmatrix}$$

Use the above five steps to get the most likely target words. The probability of སློབ is 0.62. We desire probability generated to match the true probability x_3[0,0,1,0]. And use gradient descent to update W and W'.

3. Word2vec realize

This part mainly introduces the implementation of wrord2vec, including the main work of data preparation and the final results of the experiment.

3.1. data preparation

- Letters from the Tibetan WeChat Official Account crawl a large number of text.

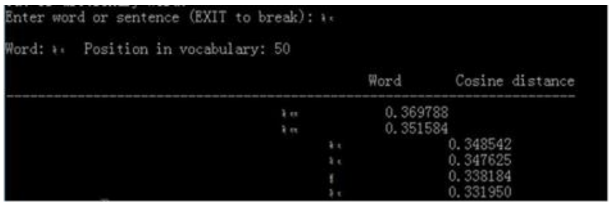
- Getting rid of complete gibberish, in both Chinese and English and digital work, which is getting the exact Tibetan sentences.
- Segmentation of Tibetan text corpus separated by Spaces or TAB key to the corpus.
- Named after copy to Word2vec trunk, which runs on the server.

3.2. Word2vec main work
There are some main words in Word2vec:

- Segmentation, stem extraction, and morphological reduction. The difficulty is to break each sentence down into an array of words.
- Construct the dictionary, the word frequency statistics. Going through all the text to find all the appear words and count how often they appear.
- Construct a tree structure. Huffman tree is constructed according to occurrence probability, and all categories should be in leaf nodes.
- The generated code in the binary code to reflect the positions of the nodes in the tree.
- Initialize all the nodes in the middle of the vector and the vector of words in a leaf node. In the leaf node, the word vector of each word is stored as the input of the neural network. The non-leaf nodes store intermediate vectors, which together with the input determine the classification results for the parameters used in the hidden layer in the neural network.
- Training and word vector in the middle. For a single word, which changes only the intermediate vector of the node on its path at most, and the other nodes remain the same.

4. Experimental result

4.1. Semantic result
After entering Tibetan words གཅེས, which is 1499 places in the corpus text, and list a number of words that are semantically relevant གཅེས, because Word2vec calculates the cosine value, the range of distance is between 0 and 1. The larger value means the higher correlation between the two words. So the more words on top, the more closely related to གཅེས. The result is shown in Figure 3 below.



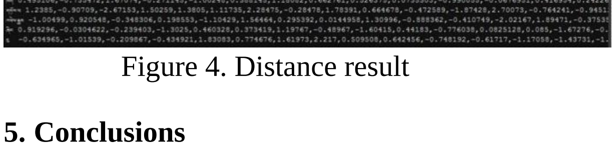
```
Enter word or sentence (EXIT to break): [illegible]

Word: [illegible]   Position in vocabulary: 50

                                   Word       Cosine distance
--------------------------------------------------------------
                          [illegible]    0.369788
                          [illegible]    0.351584
                              [illegible]      0.348542
                              [illegible]      0.347625
                              [illegible]      0.338184
                              [illegible]      0.331950
```

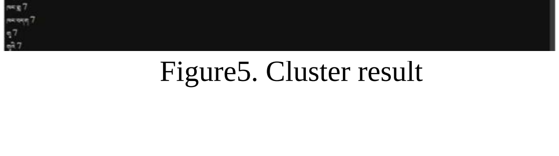
Figure 3. Semantic result

4.2. Distance result and Cluster result
Figure 4 is the vectors.bin model file after training. It shows each word and its corresponding vector in Tibetan corpus. The vector dimension is the size of the parameters set during the training. By sorting a file for keyword clustering, the partial clustering results are captured as shown in Figure 5.



```
[illegible] -0.682657,07.19661,1.35824,-0.890598,-1.73474,0.895112,-0.723618,-0.654312,0.209572,0.125997,-0.274847,1.20902,1.55115,-1.14324,
[illegible] 1.21907,1.26718,-0.530594,-2.10431,-1.30903,-0.373515,1.4053,-1.03786,0.49691,0.55105,-0.358545,0.377926,1.62904,-0.22062,-0.299
[illegible]  -0.0757671,0.978531,0.790304,-0.0250936,-1.10581,-0.0380668,-0.0781739,-1.7586,0.271328,1.31418,0.149594,0.043342,3.09725,0.23
[illegible] 0.475738,1.05581,-0.36887,0.204112,0.13028,-1.6957,-1.19614,-0.763605,-0.563961,1.80147,-1.18567,-1.72189,3.64981,-1.50295,0.093
[illegible]  -0.240369,-0.373754,-1.06019,0.33293,1.63094,-0.537226,1.78206,0.0555687,-0.905837,1.56716,-0.410595,0.646897,0.788931,0.37995
[illegible] -1.1033,0.798718,0.722804,1.54821,0.445301,0.672839,-0.100556,1.29969,-1.10176,-0.947318,0.725011,-0.208525,-0.197284,0.264334,0
[illegible] 0.184165,0.826825,-0.720878,-0.213585,0.0376372,0.335026,0.350088,0.225676,-1.25829,1.60256,-0.452775,-1.44718,1.54548,-0.56175,
[illegible] 2.22305,0.249937,1.05272,0.135355,0.881195,0.0589232,-2.30738,0.583851,2.41722,-2.07353,-1.0741,-1.35678,3.24519,-1.00462,1.2455
[illegible]  1.25168,0.748885,0.185483,-0.312651,-1.544,-0.149575,1.05694,-1.17616,0.0697866,0.1524,0.0273746,-0.564893,3.0328,-0.0469952,-
[illegible]  -0.326353,-0.410843,0.169261,-1.35834,-1.38106,0.322605,-0.125515,-0.181002,0.457972,-0.0539774,0.593357,0.561166,-0.146985,-0
[illegible] 0.495106,-0.735472,1.67074,-0.271145,-1.00248,0.388145,1.18082,0.662761,0.526375,0.0735303,-0.990055,-0.0676951,0.416954,0.24226
[illegible] 1.2385,-0.90709,-2.67153,1.50259,1.3805,1.11735,2.28475,-0.28478,1.78391,0.664678,-0.472589,-1.87428,2.70073,-0.764241,-0.9451
[illegible] -1.00499,0.920548,-0.348306,0.198553,-1.10429,1.56464,0.295392,0.0144958,1.30996,-0.888362,-0.410749,-2.02167,1.89471,-0.37535
[illegible] 0.919296,-0.0304622,-0.239403,-1.3025,0.460328,0.373419,1.19767,-0.48967,-1.60415,0.44183,-0.776038,0.0825128,0.085,-1.67276,-0.
[illegible] -0.634965,-1.01539,-0.209867,-0.434921,1.83083,0.774676,1.61973,2.217,0.509508,0.642456,-0.748192,-0.61717,-1.17058,-1.43731,-1.
```

Figure 4. Distance result



```
[illegible] 6
[illegible] 6
[illegible] 6
[illegible] 6
[illegible] 6
[illegible] 6
[illegible] 7
[illegible] 7
[illegible] 7
[illegible] 7
[illegible] 7
```

Figure5. Cluster result

5. Conclusions

In this paper, we present two modifications to the original models in Word2vec that improve the word embeddings obtained for syntactically motivated tasks[5]. Word2vec constructs a multi-layer neural network, then obtains the corresponding input and output in Tibetan text corpus, and constantly modifies the parameters of the neural network during the training process. Finally, the word vector is obtained. The specific Word2vec uses two training models, CBOW and Skip-Gram, and combines hierarchy softmax and Negative Sampling to optimize the parameters, which obtains the distributed word vector. In short, Word2vec greatly simplifies the network structure, at the same time, because the simple network structure brings low computational complexity, we can calculate the very accurate

Acknowledgments
Supported by the Fundamental Research Funds for the Central Universities (Research on Unified Acoustic Models of Mandarin and Tibetan, 31920170145)
high-dimensional word vector on the larger dataset.

References
[1] Mikolov T, Sutskever I, Chen K， et al. Distributed Representations of Words and Phrases and their Compositionality [J]. 2013,26:3111-3119.
[2] Le Q V, Mikolov T. Distributed Representations of Sentences and Documents[C]//ICK/IL. 2014, 14: 1188-1196.
[3] Mikolov T, Chen K, Corrado G, et al. Efficient Estimation of Word Representations in Vector Space [J]. Computer Science,2013.
[4] Goldberg Y, Levy O. word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method[J]. arXiv preprint arXiv: 1402.3722, 2014.
[5] Ling W, Dyer C, Black AW, et al. Two/too simple adaptations of word2vec for syntax problems[C]//Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2015: 1299-1304.