COMP 642 — Machine Learning Feb 17, 2022

Lecture 12: Data Types 1: Documents with Embedding Models

Lecturer: Anshumali Shrivastava *Scribe By: Hongyu Ni, Bo Wang, Xinna Pan, Chule Hou*

Disclaimer: *These lecture notes are intended to develop the thought process and intuition in machine learning. The materials are not thoroughly reviewed and can contain errors.*

1 Background of Word Embedding

Words have meaning(s) associated with them. As a result, we can represent word tokens in a dense vector space (few hundred real numbers), where the location and distance between words indicates how similar they are semantically (See Figure 1). This representation is called word embeddings.

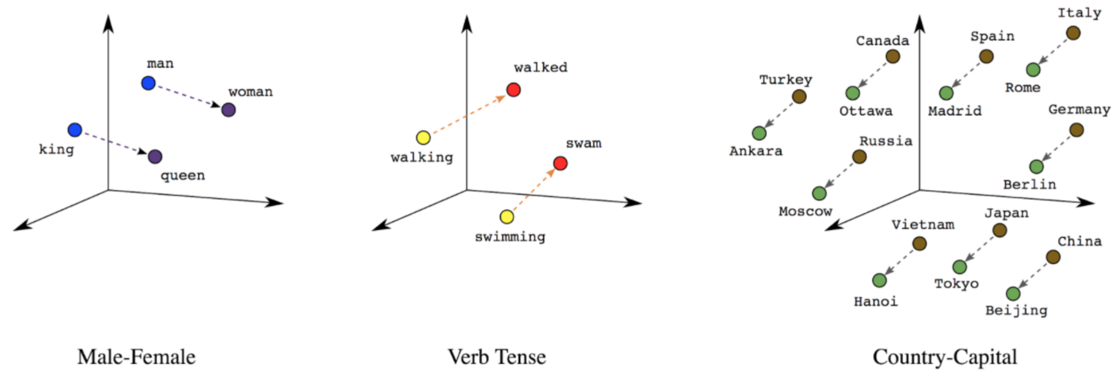


Figure 1: Word Embeddings

The semantic relationship between different embedding vectors can also be illustrated by the following example:

Say that we have the embedding vectors of the word "**king**", "**queen**", "**man**", and "**woman**". The resulting vectors of subtracting "**queen**" from "**king**" and "**woman**" from "**man**" would be very similar direction-wise since they both carry similar values for that ***Gender*** feature.

"**king**" – "**queen**" ≃ "**man**" – "**woman**"

or equivalently,

"**king**" – "**queen**" + "**woman**" ≃ "**man**"

1.1 Embedding Models Introduction

Sequence models often have an embedding layer as their first layer. This layer learns to turn word index sequences into word embedding vectors during the training process, such that each word index gets mapped to a dense vector of real values representing that word's location in semantic space (See Figure 2).

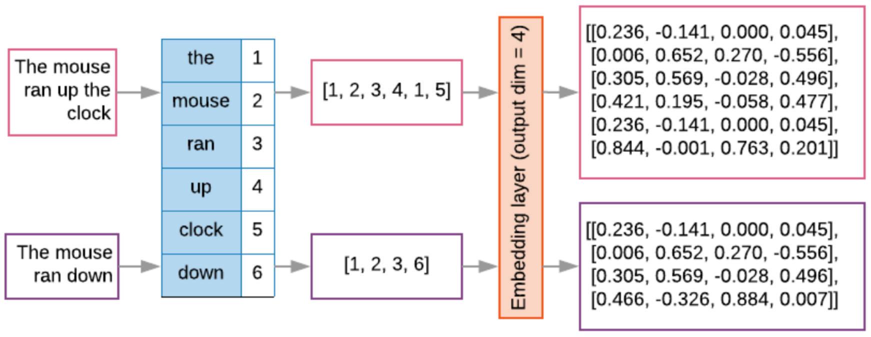


Figure 2: Embedding Layer

The embedding Model is essentially the same as one-hot encoding. They both vectorize words from the corpus, but one-hot only lets the computer know that the word exists. Embedding maps the one-hot encoding vector to a new space.

1.2 Word Embedding

Existing machine learning methods often cannot process text data directly, so it is necessary to find a suitable method to convert text data into numerical data, which leads to the concept of Word Embedding. Word embedding (See Figure3), an early pre-training technique, is a general term for language models and representation learning techniques in natural language processing (NLP). It refers to embedding a high-dimensional space, whose dimension is the number of all words, into a continuous vector space of much lower dimension. And each phrase is mapped as a vector on the real number domain, which is also a distributed representation that each dimension of a vector has no practical meaning, while the whole represents a concrete concept. Type of text representation:

Bag-of-words based on one-hot, tf-idf, textrank, etc.;

Topic models: LSA (SVD), pLSA, LDA;

Fixed representations based on word vectors: word2vec, fastText, glove;

Dynamic representation based on word vectors: ELMO, GPT, bert;

They are also the most commonly used text representations in the NLP field. The text is composed of each word, while one-hot is the simplest word vector. But it comes with many issues like dimensional disasters and semantic gap. For example, high computational complexity will be caused by constructing a co-occurrence matrix and then using SVD to analyze the construction of word vectors. While early research on word vectors usually comes from language models, such as NNLM and RNNLM, whose main purpose is language models, with a by-product of word vectors.

2 Word2Vec

Word2vec is one of the standard word embedding models [1][2]. There are two architectures proposed for word2vec: CBOW and Skip-Gram.

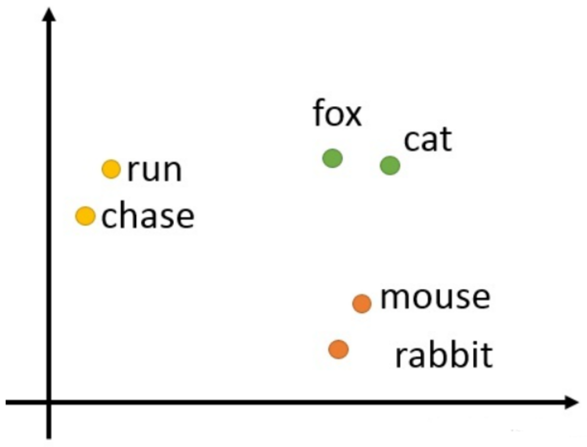


Figure 3: Encode to Vector based on Meanings

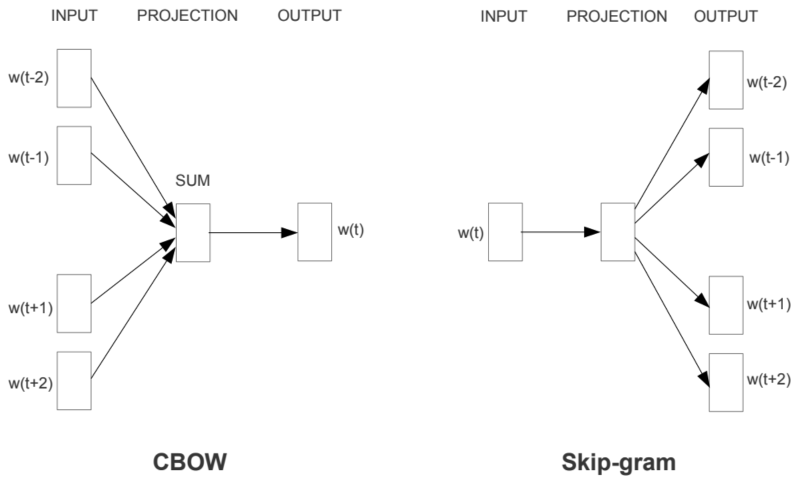


Figure 4: CBOW and Skip-gram

CBOW predicts a masked word using its context (fill in the blanks model). For each word, it learns two vectors to represent the two roles that a word can perform - first, when it is present in the context of the masked word, and second, when it is masked.

Let $U = [\boldsymbol{u}_1, ..., \boldsymbol{u}_N]^T \in \mathbb{R}^{N \times D}$ and $V = [\boldsymbol{v}_1, ..., \boldsymbol{v}_N]^T \in \mathbb{R}^{N \times D}$ where N is the size of the vocabulary, and D is the size of the word vector. U models the first role and is used to calculate the context vector, c, given by

$$c = \sum_{i \in [-b,b]-\{0\}} \boldsymbol{u}_{w_i} \tag{1}$$

where b is the size of the context window and w_i is the index of each word (w_0 is the index of the masked word; the rest are the indices of the context words). V models the second role and learns the masked word vector for w0, given by v_{w_0} . The probability p of w0 to occur in the context of $\{w_b, .., w_{-1}, w_1, .., w_b\}$ is given by

$$p(w_0 \mid W_{[-b,b]-\{0\}}) \propto exp \boldsymbol{v}_{w_0}^T \boldsymbol{c} \tag{2}$$

When the context has only one word, the model can be simplified to use the current word x to predict its next word y.(See Figure5)

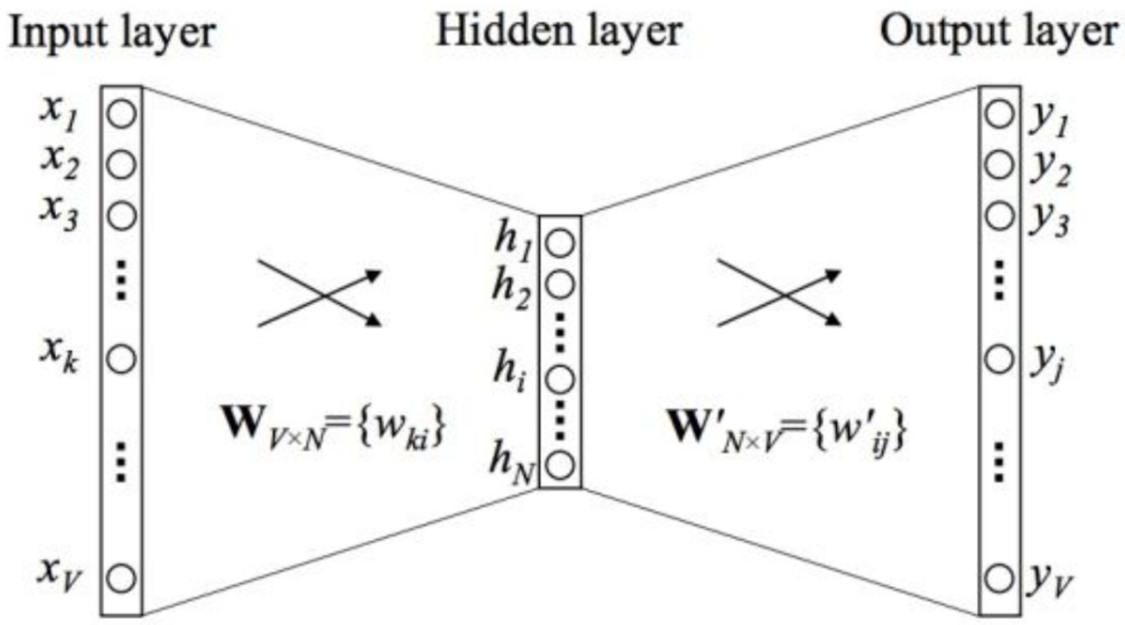


Figure 5: CBOW Model

The Skip-Gram model of word2vec is similar to CBOW.(See Figure 6) The key difference is that it predicts the context words using the masked word.

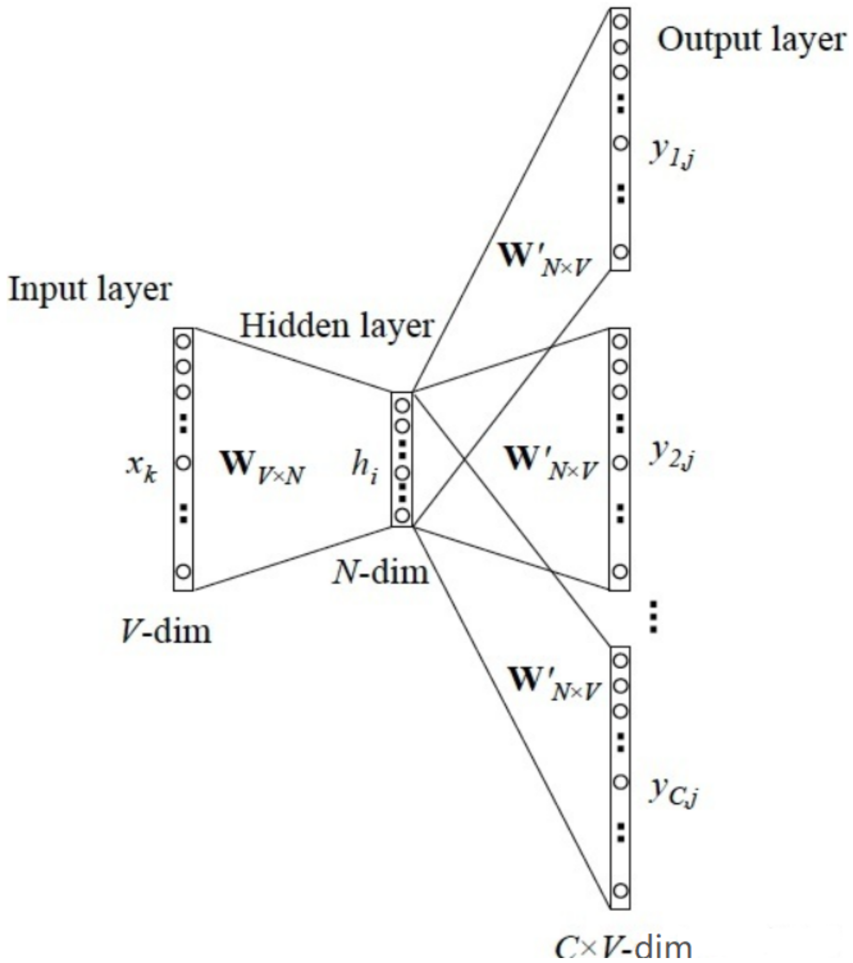


Figure 6: Skip-Gram Model

By using a large corpus for model training, the weighted model from the input layer and hidden layer can be generated. Although Word2Vec can generate the good result, the model still have some defects, such as the model does not consider word order and dose not consider statistical full-text words.

3 New Model: Attention Word Embedding (AWE)

3.1 The AWE Model

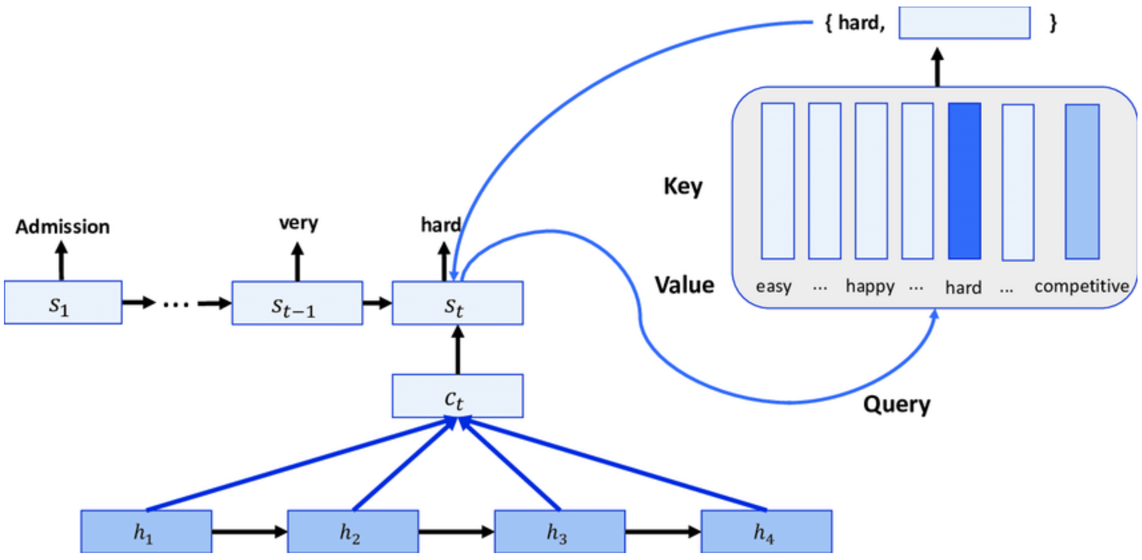


Figure 7: AWE Model

We now explain AWE[3], our proposed word embedding model that incorporates the attention mechanism. AWE augments the CBOW (continuous bag-of-words) of word2vec with the attention mechanism in two different ways. First, we introduce two new matrices, a key matrix, $K \in \mathbb{R}^{(N \times D')}$ and a query matrix, $Q \in \mathbb{R}^{(N \times D')}$, where N is the size of the vocabulary and $D' \in \mathbb{Z}$. With the attention mechanism, the context vector c is not modeled as a simple sum in (1) but rather as a *weighted* sum of context word vector embeddings

$$c = \sum_{i \in [-b,b]-\{0\}} a_{w_i} \boldsymbol{u}_{w_i} \tag{3}$$

where a_{wi} is the attention weight of each context word vector $\boldsymbol{u}_{w_i}$ calculated using the key matrix K and the query matrix Q

$$a_{w_i} = \exp\left(\boldsymbol{k_{w0}^T q_{wi}}\right) \tag{4}$$

Second, we share the weights between the context word embedding matrix and the masked word embedding matrix in our model, i.e., we set $U = V$. Sharing weights is a natural and intuitive choice, since both matrices embed the meaning of a word in its vector representation, and the meaning of the word remains the same irrespective of it occurring in the context window, or as the masked word. In CBOW, the choice is to have two separate matrices, U and V, is justified as it leads to increase in performance. However, in AWE, the intuitive choice to have U same as V works better and adds interpretability to the model. Op of that, it has an added advantage. The number of parameters in AWE are much less as compared to CBOW even though AWE has one more matrix than CBOW. The reason is that the key matrix $K \in \mathbb{R}^{(N \times D')}$ and a query matrix, $Q \in \mathbb{R}^{(N \times D')}$ are much samller as compared to the value matrix, $V = [\boldsymbol{u}_1, ..., \boldsymbol{u}_N]^T \in \mathbb{R}^{N \times D}$.

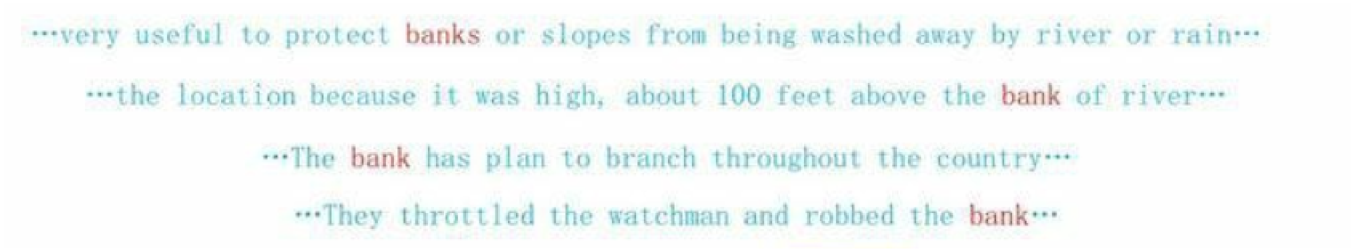
3.2 GloVe Model

GloVe is called Global Vectors as the global corpus statistics are captured directly by the model. It is an unsupervised word embedding method, word vector learning method. Glove is a global log bilinear regression model. As the name suggests, the model uses the global features of the corpus. The co-occurrence frequency matrix of words, and the optimization objective function is log-linear and solved in the form of regression.

The number of co-occurrence of words is often not strictly proportional to their semantic relevance, so the direct use of covariance to characterize the relevance between words is not effective, so the authors characterize the relevance by introducing a third word, through the difference between words. The differences were chosen to better determine the correlation between words by the number of co-occurrence probabilities of two words with the same word.

4 Challenge

As we know, polysemy is a frequent phenomenon in our natural language, representing highly flexibility and efficiency of different language. However, word Embedding can not solve this problem well.



…very useful to protect banks or slopes from being washed away by river or rain…

…the location because it was high, about 100 feet above the bank of river…

…The bank has plan to branch throughout the country…

…They throttled the watchman and robbed the bank…

Figure 8: Static Word Embedding

As shown in the figure above(See Figure 8), the polysemous word 'Bank' has two common meanings. But when Word Embedding encodes the word, it cannot distinguish them. Although the words appearing in different contexts, the same word occupies the parameter space. So when trained with a language model, the same 'bank' will be predicted, no matter what kind of context passes through word2vec model. And this leads that two different context information will be encoded to the same in the word embedding space. Therefore, word embedding cannot distinguish the different semantics of polysemy, which is a serious problem in NLP field.

The word vector obtained by word embedding cannot solve the problem of polysemy. Therefore, dynamic representation methods based on language model are introduced, like ELMO, GPT and Bert.

Take ELMO(See Figure 9) as an example. The essential idea of is to use the language model to learn the Word Embedding in advance. At this time, polysemous words still cannot be distinguished, but when the Word Embedding is actually used, the word already has a specific context. Then you can use the semantics of the context to adjust the Word Embedding, which will make it better represent the specific meaning in this context. In summary, ELMO is an attempt of dynamically adjusting Word Embedding according to the current context.

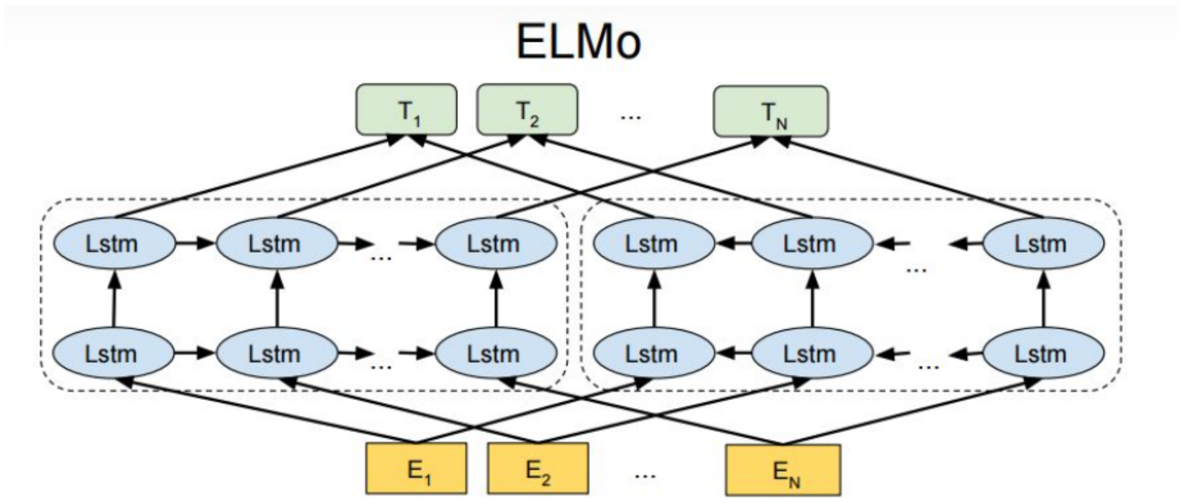


Figure 9: ELMo Network

References

[1] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[2] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, vol. 26, 2013.

[3] S. Sonkar, A. E. Waters, and R. G. Baraniuk, "Attention word embedding," *arXiv preprint arXiv:2006.00988*, 2020.

[4] https://developers.google.com/machine-learning/guides/text-classification/step-3 Step 3: Prepare Your Data