Words Embedding and News Topic Classification

Jean Lucien Randrianantenaina
Applied Mathematics, Machine Learning and Artificial Intelligence
Stellenbosch University

***Abstract*—abstract**

I. Introduction

II. Data and Data preparation

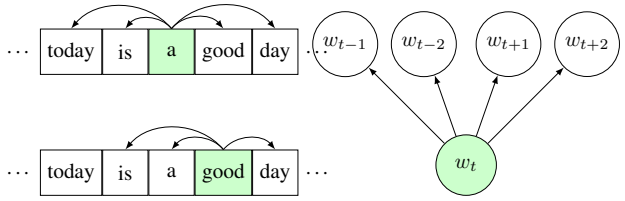
III. Word Embeddings and Skip-Gram Model

Natural Language Processing (NLP) tasks often benefit from representing words as vectors in a high-dimensional space. Word embeddings capture semantic and syntactic information about words, allowing models to reason about word relationships. While one-hot encoding is a simple approach, it fails to capture these important relationships.

This section explores word embeddings and the Skip-Gram model, a popular method for learning word representations. The Skip-Gram model, proposed by Mikolov et al. [2], predicts surrounding context words given a central word. This approach helps the model learn the semantic meaning of words based on their co-occurrence in text.

A. Skip-Gram

The Skip-Gram model, models the words representation by considering a center word c and predicting M words that are around c, the M is refered as window size. The Figure 1 illustrate a skipgram model with a window size of 2.



(a) Skip-Gram Illustraion (b) Skip-Gram Model

Fig. 1: The skip-gram model

In other words, we compute $p(w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2}|w_t)$. The following assumptions simplify the model:

- Each window is an identically and independently distributed sample.
- Within each window, each context word is conditionally independent given the center word.
- Each context word can be predicted from the center word in the same way, irrespective of its position.

Considering the sequence $w_{1:T}$ and a window size M, we can derive the negative likelihood of our model written as follows, with the two first assumptions:

$$J(\theta) = -\log \prod_{t=1}^{T} P_\theta(w_{t-M:t-1}, w_{t+1:t+M}|w_t) \tag{1}$$

$$= -\sum_{t=1}^{T} \sum_{\substack{-M \le j \le M, \\ j \neq 0}} \log P_\theta(w_{t+j}|w_t) \tag{2}$$

To obtain $P_\theta(w_{t+j}|w_t)$ we consider the third assumption. We also denote the vector represntation of a context word w $\mathbf{u}_w$, and $\mathbf{v}_w$ if w is centre word. Then for a centre word c and context word o, we have:

$$P_\theta(w_{t+j} = o|w_t = c) = P(o|c) = \frac{e^{\mathbf{u}_o^\top \mathbf{v}_c}}{\sum_{w \in \mathcal{V}} e^{\mathbf{u}_w^\top \mathbf{v}_c}} \tag{3}$$

where $\mathcal{V}$ is the set of vocabulary. By denoting $\mathbf{f}_\theta(\cdot)$ our, model, then for a centre word c we have

$$\mathbf{f}_\theta(w_t = c) = \frac{1}{\sum_{w \in \mathcal{V}} e^{\mathbf{u}_w^\top \mathbf{v}_c}} \begin{bmatrix} e^{\mathbf{u}_o^\top \mathbf{v}_c} \\ e^{\mathbf{u}_o^\top \mathbf{v}_c} \\ \vdots \\ e^{\mathbf{u}_o^\top \mathbf{v}_c} \end{bmatrix} = \text{softmax}\left(\mathbf{U}\mathbf{v}_c\right) \tag{4}$$

In that model, the parameters are two $D \times |\mathcal{V}|$ matrices, $\mathbf{U}$ and $\mathbf{V}$, where each column of $\mathbf{U}$ represents the vector embedding of a word in the vocabulary as a context word, and each column of $\mathbf{V}$ represents the vector embedding of a word in the vocabulary as a center word.

Then, to find the optimal parameters, we can use Gradiend descent or its variant to find the optimal parameters by minimizing the negative log likelihood $J(\theta)$.

One of the major issue of with this approach is the computational complexity induced by the softmax, when we deal with a big dataset. A solution to this is proposed in [3] that we will present in the next section.

B. Improved Word2vec with subsampling and negative sampling

Some words in the corpus have a high frequency, but irrevelents than some rare words. To address this imbalance and focus on more informative words, we can discard some frequent words using a heuristic probability:

$$p(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}} \tag{5}$$

Where $f(w_i)$ if the frequency of the word w_i, and t a choosen threshold suggested to be 10^{-5}. By doing this, we reduce the size of our vocabulary while keeping more informative words.

On the other hand, from the Noise Contrastive Estimation says that a good model should be able to differntiate data from noise by the mean of a logistic regression [3]. Based on that, for each pair (c, o) (positive) we sample K pairs (c, w_i) (negatives) then our model should predict if a given pair is a valid pair i.e, a valid centre context words or not. Thus, for a single pair we have the following negative log likelihood:

$$J_{c,o}(\theta) = -\log \sigma(\mathbf{u}_o^\top \mathbf{v}_c) - \sum_{i=1}^{k} \log \sigma(\mathbf{u}_{w_i}^\top \mathbf{v}_c) \tag{6}$$

This approcah reduce the computaion of a big softmax into sevaral sigmoid.

C. Words embedding evaluation

This section discusses how to evaluate word embeddings and their usage in text classification. Cosine similarity is a common metric used to measure the similarity between two word embeddings, denoted by $\mathbf{u}$ and $\mathbf{v}$. It is defined as:

$$\cos(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u}^\top \mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|} \in [-1, 1] \tag{7}$$

Words with similar meanings will have a cosine similarity close to 1, while dissimilar words will have a value closer to -1. Additionally, the distance between two embeddings can be calculated using:

$$\mathrm{d}(\mathbf{u}, \mathbf{v}) = 1 - \cos(\mathbf{u}, \mathbf{v}) \in [0, 2] \tag{8}$$

However, evaluating the overall quality of word embeddings is a complex task. Here are two common approaches:

- **Visualization:** Techniques like PCA, t-SNE, UMAP can be used to project word embeddings into a lower-dimensional space for visual inspection. By looking at known similar words and how they cluster together, we can gain insights into the quality of the embeddings.
- **Intrinsic Evaluation:** This involves evaluating the model's ability to capture semantic relationships between words. We can compare the cosine similarity of known synonym or analogy pairs with the model's predictions.

D. Text Classification

One powerful application of word embeddings is text classification. We can leverage the embedding vectors to represent the meaning of an entire sentence.

$$\mathbf{w}_{\text{sentence}} = \frac{1}{T} \sum_{T}^{t=1} \mathbf{w}_t \tag{9}$$

where $\mathbf{w}_t$ is the embedding of the t^{th} word and T is the total number of words. This sentence embedding is then fed into a text classifier to predict its category.

IV. Results and Discussion

V. Conclusion

References

[1] Herman Kamper. *NLP817*. https://www.kamperh.com/nlp817/, 2022–2024.

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

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