Group Members:
Tjhin Stephen Taharuddin (1155199520)
Lam Ting Wai (1155212391)

Word Embeddings in Topic Models

ABSTRACT

We intend to learn about the latest research on topic models and how word embeddings are used to obtain a topic distribution of a given document. Topic models are algorithms that discover abstract topics in a body of text based on the semantic information contained in the text without having to pre-label the text. One of the most well-known topic models is Latent Dirichlet Allocation (LDA). However, LDA is oblivious to the ordering, semantics, and meanings of the words in the document, which may not provide the best result. A newer topic model, Top2Vec, is introduced to address these issues. The author of Top2Vec claims that it can overcome the limitations of LDA and perform better. In this project, we aim to use the metrics of coherence score to verify and replicate the results the author of Top2Vec claims and to demonstrate the use of probability for topic model evaluation. In the end, we can show that Top2Vec consistently outperforms LDA in various datasets, which matches the author's claims on Top2Vec's performance.

1. Introduction

Topic models are statistical models that discover topics contained in a set of documents without pre-labeling the texts with the topics contained within them [1]. Topic modelling has many uses for applications such as fuzzy text searching, text summarization, and text data mining. Topic modelling has advanced from simple word frequency and distribution analyses (as seen in LDA (Latent Dirichlet Allocation) and PLSA (Probabilistic Latent Semantic Analysis)) to utilizing the latest advances in natural language processing research, such as embeddings and transformer models (as seen in Top2Vec and BERTopic).

Word embeddings seem to be a good candidate to improve and enhance pre-existing bag-of-word style models such as LDA since word embeddings like Word2Vec carry semantic information that simpler bag-of-word models simply cannot encode [1]. Word embeddings allow the model to "understand" the nuances of human language due to its cognizance of the context within which a certain word is used in the text [2]. Word embeddings represent words as high-dimensional vectors in what is called the embedding space [2]. Embedding-based topic models expand on this idea by representing entire documents as vectors in the embedding space, allowing the models to also discover topic words relevant to the document by finding embedding vectors in the neighborhood of the document embedding vector [1].

We aim to study how word embeddings can be used to enhance the performance of topic models by adding semantic meaning and increasing the ease of use of these newer topic models. In doing so, we plan to compare 2 topic models, with one not using any embeddings (LDA) and the other relying heavily on embeddings (Top2Vec). We will perform quantitative comparisons of the quality of the topics and topic words produced by each model and provide an explanation of how and why these differences have come about.

1.1. Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation (LDA) is a topic model that uses a bag-of-words (BOW) model to keep track of the words contained in the documents and topics within a certain text corpus. LDA operates on the assumption that each document was generated by randomly sampling a per-document topic distribution from a topic Dirichlet distribution common to the text corpus and randomly sampling a per-topic word distribution from a word Dirichlet distribution common to the entire corpus. Each word in the document is then formed by sampling a topic from the topic distribution of the current document and then sampling a word from the word distribution of the chosen topic. While this assumption may not be true, it is a valid assumption for the topic model, which has no a priori knowledge about the text corpus fed into it or the generation process of each document. The Dirichlet distribution is suitable to model the per-topic document distributions and per-word topic distributions since each topic and each word make up a certain "proportion" of each document and each topic, respectively, that is, LDA attempts to find a mix of topics and topic words for the text corpus that would be able to faithfully reconstruct each document to a certain degree if the words of each document were to be randomly sampled as outlined above. [3]

The problem of topic modelling then reduces to finding the most likely parameters to the Dirichlet distributions that were used to sample the per-document topic distributions and the per-topic word distributions. This is equivalent to maximizing the posterior probabilities of the parameters to the Dirichlet distributions given the words observed in each document. The dependencies between the parameters of the distributions and the words observed in the documents are then modelled by a Bayesian network, which can then be efficiently solved by existing algorithms. It then remains to repeat the sampling process to produce the per-document topic distribution and the per-topic word distributions. [3]

However, due to the rather simplistic bag-of-words and random sampling assumptions, LDA may fail to discover certain words that refer to the same topic if, for example, each document heavily uses a certain word from a certain topic while avoiding another word from the same topic. LDA's performance will also suffer in the presence of frequency outliers in the text corpus, i.e., extremely common words and extremely rare words, leading to the necessity of lemmatization, the use of a list of stop-words, and other document preprocessing techniques [1]. LDA also requires the number of topics to be identified to be known beforehand since, otherwise, the algorithm would not be able to know how many topic distributions have to be sampled to obtain the topic representation [1].

1.2. Top2Vec

Top2Vec, on the other hand, makes use of word embeddings to represent the semantic relationships and cluster out different topics found in the documents [1]. Top2vec makes use of pre-trained word embedding models specified by the user, such as Doc2Vec, Universal Sentence Encoder, or MiniLM to generate the word and document vectors, where the semantic relationship of the word and document is represented by the distance between them (commonly using the Euclidean distance metric) [1, 4]. The closer the word and document vector, the higher the semantic similarity is between them. To effectively cluster the sparsely distributed high-dimensional document vectors, Top2Vec uses uniform manifold approximation and projection (UMAP) for dimension reduction. Top2Vec can then learn the semantic relationship between the word and document vectors to cluster them into different dense topic clusters using hierarchical density-based spatial clustering of applications with noise (HDBSCAN). Hence, the centroid of the document vector within a dense cluster can be calculated to represent the topic vector that represents the topic of all documents within the cluster [1].

Top2Vec has various advantages over LDA. The first is that Top2Vec does not need stemming and lemmatization. LDA makes use of BOW representation of documents that does not take semantic relationships into account, hence requiring the use of stemming and lemmatization to make the words into the base form, i.e., changing *danced* and *dancing* to *dance* [5], causing LDA to be able to provide the most accurate results. However, Top2Vec does not require stemming and lemmatization because Top2Vec can already take the semantic relationships of words into account [1]. Secondly, Top2Vec does not need the use of a stop-words list, unlike LDA. The words that have the highest probability in the topic mostly consist of common words that are stop-words, including *the* and *are*, which are not informative. With Top2Vec, on the other hand, word vectors of common words will have almost identical distances across all document vectors, so they would be identified as noise by HDBSCAN [1]. Thirdly, Top2Vec does not require prior knowledge of the number of topics in the text corpus, which LDA requires. LDA requires users to manually state the number of topics to be found, which is not convenient nor is it user-friendly, as we may not have prior knowledge about the datasets nor the number of topics that should be set. As Top2Vec can automatically generate the topic vectors from the dense cluster, the total number of topics can be automatically calculated by the total number of topic vectors [1].

2. Methodology

2.1. Topic Coherence

We intend to run a comparative study on the performance of the LDA and Top2Vec topic models using some common large-text corpora. What constitutes good modelling of a document by topics has been acknowledged to be a subjective evaluation and requires a thorough understanding of the documents being modelled [6]. However, manually evaluating the suitability of a set of topics for each document is time-consuming, expensive, and unrealistic. As a result, researchers have proposed topic quality measures, such as the C_V score, perplexity score, and the UMass score, that allow the performance of topic models to be quantified and compared [7].

A popular metric used to evaluate topic models, which we will use to compare LDA and Top2Vec's performance, is called the C_V score. The C_V score measures the coherence of a topic model or how well a topic is supported by words in the documents fed into the topic model. The C_V score relies on a fundamental assumption in natural language research that words that refer to similar topics generally co-occur together, often referred to as the distributional hypothesis. Therefore, the C_V score will assign a higher score when words chosen to be of the same topic co-occur often in every document. The C_V score has been shown to correlate well with human topic interpretability rankings. [6]

The C_V score takes a list of topic words from each topic and the set of all documents as input. It first takes the most probable words in each list of topic words (denoted $W = \{w_1, w_2, \ldots, w_{|W|}\}$) and forms pairs of the most probable words (denoted (w_i, w_j)) from each list of topic words. The frequency at which the pair of words co-occurs is calculated by counting the number of co-occurrences in a Boolean sliding window of length 110, which is advanced over each document at a rate of one word every step. The Boolean sliding window constructs a sub-document of length 110 from the given document and checks if the pair of words co-occurs in this sub-document. The Boolean sliding window attempts to introduce some locality by producing a local context within which topic words should co-occur, thereby measuring how much the distributional hypothesis is valid for the pair of words. The number of co-occurrences is then used to approximate the joint probability of the pair of words co-occurring in the text corpus (denoted $p(w_i, w_j)$) by dividing the number of co-occurrences

by the number of sub-documents produced by the Boolean sliding windows. We can also calculate the word probabilities of single words (denoted $p(w_i)$) simply by dividing the number of occurrences of a word by the total number of words in the text corpus. [6]

The probabilities are then used to calculate the normalized pointwise mutual information (NPMI), which measures how much more likely two words will co-occur with a certain joint probability distribution calculated from the text corpus than how likely the two words would co-occur if their occurrence were assumed to be independent of each other. The NPMI would therefore be higher when two topic words co-occur together often, which is generally the case for highly probable topic word choices. The formula for NPMI is

$$NPMI(w_i, w_j) = \frac{log\frac{p(w_i, w_j)+\varepsilon}{p(w_i)p(w_j)}}{-log(p(w_i, w_j)+\varepsilon)}$$

where the ε term is introduced to avoid taking the logarithm of zero when two words never co-occur in any document. The values of the NPMI of a given word paired with each word under the same topic are calculated and coalesced into a word context vector $\vec{v}(w_i)$,

$$\vec{v}(w_i) = \{NPMI(w_i, w_j)^{\gamma}\}_{j \in \{1, 2, |W|\}}$$

where the γ term is introduced to increase the weight of higher NPMI values relative to lower NPMI values. A confirmation measure is calculated for each pair of context vectors under a topic by taking the cosine similarity between both context vectors $\vec{u}$ and $\vec{w}$,

$$\phi_{S_i}(\vec{u}, \vec{w}) = \frac{\sum_{k=1}^{|W|} u_k w_k}{||\vec{u}||_2 ||\vec{w}||_2}$$

where $||\cdot||_2$ denotes the Euclidean norm. The C_V score is then given by taking the arithmetic mean of all the confirmation measures calculated for each pair of context vectors under each topic. [6]

2.2. Experimental Setup

We compared the C_V scores of LDA and Top2Vec for several text corpora. We ran our experiments on Google Colaboratory. The Top2Vec model used was obtained from the open-source library published by the Top2Vec paper's author, Dimo Angelov, which is available on Github and PyPI [1, 4]. For the LDA model, we used scikit-learn's LatentDirichletAllocation (`sklearn.decomposition.LatentDirichletAllocation`) [8]. The C_V score was calculated using Gensim's CoherenceModel (`gensim.models.CoherenceModel`) [9].

In the experiments, we have used five datasets to evaluate Top2Vec and LDA's performance. These datasets include the 20Newsgroups dataset [10], the arXiv abstracts dataset [11], the Yahoo Answers dataset [12], a selection of Wikipedia articles [13], and the IMDB reviews dataset [14]. All of the datasets except for the 20Newsgroups dataset were obtained from Huggingface [15]. The 20Newsgroups dataset was obtained from scikit-learn's fetch_20newsgroups (`sklearn.datasets.fetch_20newsgroups`) [8].

3. Results

From the above experiment setup, we obtain the results as follows. For clarity, we round off the C_V score to 4 decimal places. Firstly, in the 20Newsgroups dataset, the C_V score of Top2Vec and LDA is 0.5395 and 0.2777, respectively, with C_V score of Top2Vec higher

than that of LDA by approximately 94.27%. Secondly, in the arXiv Abstracts dataset, the C_V score of Top2Vec and LDA is 0.4969 and 0.2029, respectively, with C_V score of Top2Vec higher than that of LDA by 144.90%. This trend continues across other datasets: in the Yahoo Answers dataset, Top2Vec scores 0.5077 while LDA scores 0.4407, with a difference of approximately 15.20%; in the Wikipedia dataset, Top2Vec scores 0.4515 while LDA scores 0.3928, with a difference of approximately 14.94%; in the IMDB reviews dataset, Top2Vec scores 0.3075 while LDA scores 0.2439, with a difference of approximately 26.08%. These results suggest that Top2Vec can consistently outperform LDA across multiple testing datasets.

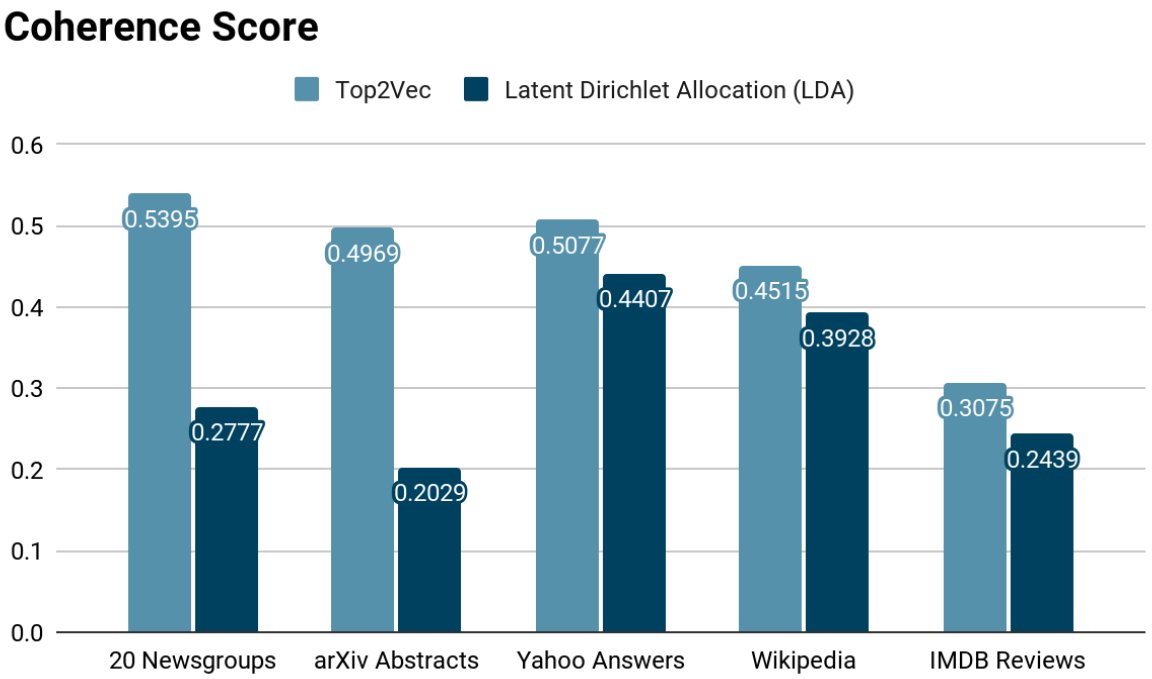

Figure 1: Top2Vec and LDA's C_V scores on different text corpora

Our results are consistent with the author's claims that Top2Vec performs better than LDA [1]. This suggests that Top2Vec, with its advantages over LDA, can be a potential supplement or replacement to LDA for topic modelling purposes.

4. References

[1] D. Angelov, "Top2Vec: Distributed Representations of Topics," 2020, doi: 10.48550/arxiv.2008.09470

[2] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," 2013, doi: 10.48550/arxiv.1301.3781

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," vol. 3, no. 4–5, pp. 993–1022, 2003, doi: 10.1162/jmlr.2003.3.4-5.993

[4] D. Angelov, *Top2Vec*. Github Repository, 2020. Available: https://github.com/ddangelov/Top2Vec. [Accessed: Oct. 2024]

[5] J. Murel and E. Kavlakoglu, "What are stemming and lemmatization?," Dec. 10, 2023. Available: https://www.ibm.com/think/topics/stemming-lemmatization. [Accessed: Dec. 19, 2024]

[6] S. Syed and M. Spruit, "Full-Text or Abstract? Examining Topic Coherence Scores Using Latent Dirichlet Allocation," presented at the 2017 IEEE International Conference on Data Science and Advanced Analytics (DSAA), 2017, pp. 165–174. doi: 10.1109/DSAA.2017.61

[7] M. Yuan, P. Lin, L. Rashidi, and J. Zobel, "Assessment of the Quality of Topic Models for Information Retrieval Applications," presented at the Proceedings of the 2023 ACM SIGIR International Conference on Theory of Information Retrieval, in ICTIR '23. New York, NY, USA: Association for Computing Machinery, 2023, pp. 265–274. doi: 10.1145/3578337.3605118

[8] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," vol. 12, pp. 2825–2830, 2011, doi: 10.5555/1953048.2078195

[9] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," presented at the Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, Valletta, Malta: ELRA, 2010, pp. 45–50.

[10] K. Lang, *NewsWeeder: Learning to Filter Netnews*. in Machine Learning Proceedings 1995. San Francisco (CA): Morgan Kaufmann, 1995, pp. 331–339. doi: 10.1016/B978-1-55860-377-6.50048-7

[11] C. B. Clement, M. Bierbaum, O. Kevin, P., and A. A. Alemi, "On the Use of ArXiv as a Dataset," 2019, doi: 10.48550/arxiv.1905.00075

[12] S. Patil, "Yahoo Answers Topics." in Huggingface Community Datasets. Huggingface, 2022. Available: https://huggingface.co/datasets/community-datasets/yahoo_answers_topics. [Accessed: Nov. 2024]

[13] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer Sentinel Mixture Models," 2016, doi: 10.48550/arxiv.1609.07843

[14] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning Word Vectors for Sentiment Analysis," presented at the Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Portland, Oregon, USA: Association for Computational Linguistics, 2011, pp. 142–150. Available: http://www.aclweb.org/anthology/P11-1015

[15] Q. Lhoest *et al.*, "Datasets: A Community Library for Natural Language Processing," presented at the Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Association for Computational Linguistics, 2021, pp. 175–184. Available: https://aclanthology.org/2021.emnlp-demo.21