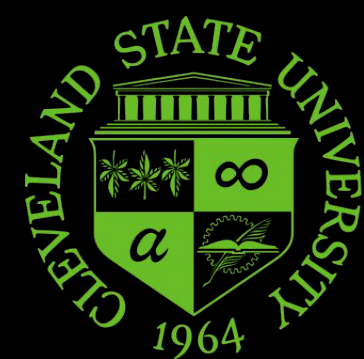


Distributed Representations of Documents for Document Clustering and Searching

Mike D'Arcy, Utkarsh Patel; Advisor: Dr. Sunnie Chung



Washkewicz College
of Engineering

Introduction

Being able to quickly and efficiently find and categorize documents is a crucial part of research in any computer information science or artificial intelligence based discipline. Information retrieval is undoubtedly one of the most important disciplines behind building systems that can group related texts and find documents based on natural language queries.

At the core of most information retrieval systems is *document vectorization*, the process of converting long text documents into concise vector representations of the words that can be searched over efficiently.

Approaches

1. Bag Of Words (BOW) Model and TF-IDF

- Latent Semantic Analysis (LSA) is a technique to find the relations between words and documents by vectorizing them in a ‘concept’ space. In LSA, the documents are represented as Bag-of-words (BOW).
- In TF-IDF (term frequency inverse document frequency) vectorization, a document is represented by a vector of each word count in the document that is normalized by its frequency in the document and its inverted frequency in all documents in the search space.

The tf-idf weight of a term is the product of its tf weight and its idf weight.

$$w_{t,d} = \log(1 + tf_{t,d}) \times \log_{10}(N/df_t)$$

Score for a document given a query

$$score(q, d) = \sum_{t \in q \cap d} tf \cdot idf_{t,d}$$

Comparing query with document with cosine similarity

$$\cos(q, d) = \frac{q \cdot d}{\|q\| \|d\|} = \frac{q}{\|q\|} \cdot \frac{d}{\|d\|} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2} \sqrt{\sum_{i=1}^{|V|} d_i^2}}$$

Drawbacks of BOW

- Ignores the syntax and semantics of words
 - n-grams, phrases
 - synonyms, polysemous words
- Loses word order

2. Paragraph Vectors

Word2vec: Continuous Skip-Gram Model

- The word2vec Skip-Gram model (Figure 1) is a Neural Network architecture that learns semantically meaningful vector representations for words.
- Given a target word w_t , the skip-gram model is trained to predict the surrounding context words $w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2}$ in a phrase window size $n = 5$.
- After training, the weights in the embedding layer capture the semantics through backpropagation as an indirect result of the prediction task, and become the word vectors.
 - Words with similar meanings are mapped to similar positions in the vector space.

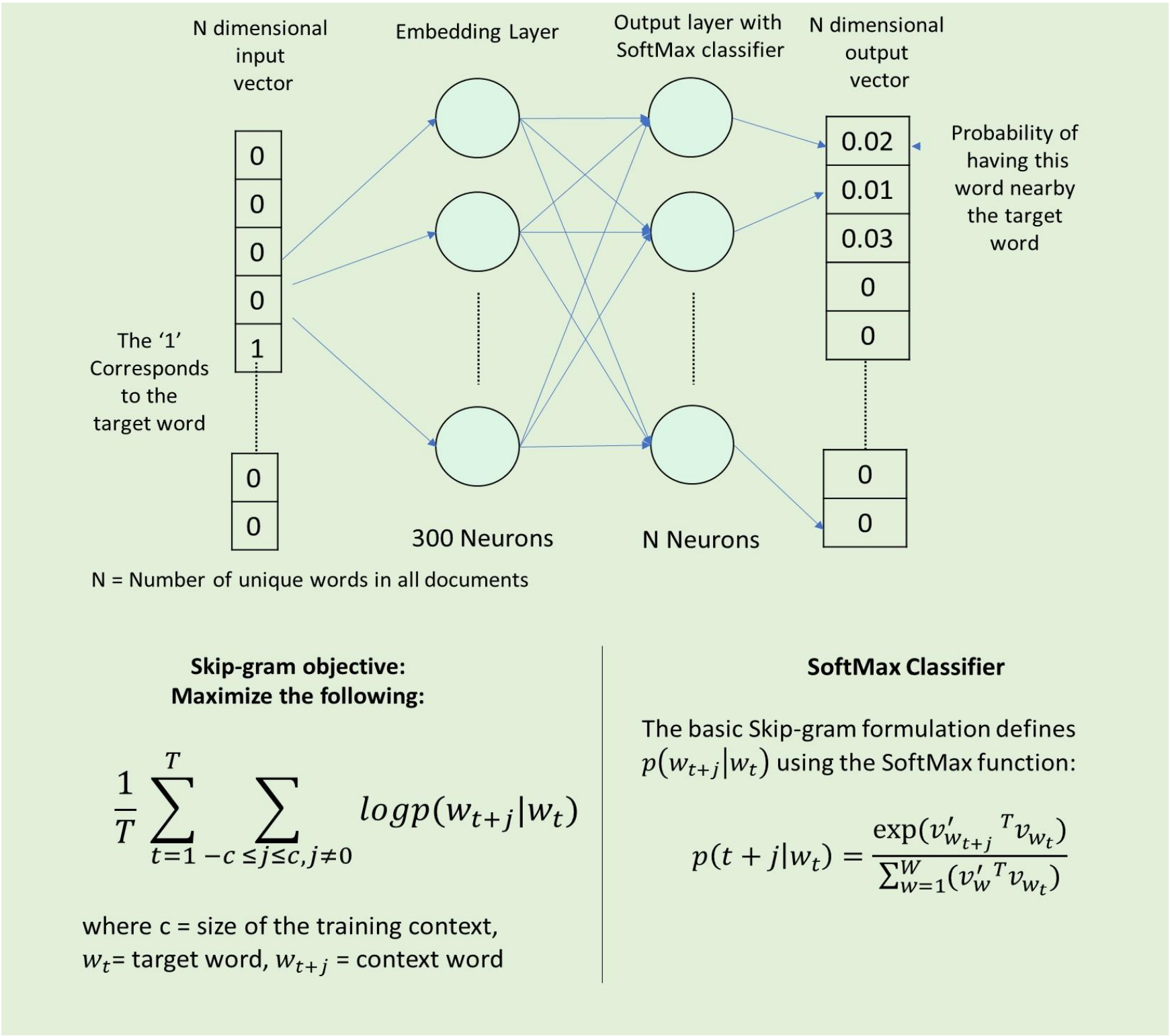


Figure 1: Skip-Gram Model

Paragraph Vector Model

- Using the word vectors (W) learned with word2vec, and the document IDs, the paragraph vectors model (Figure 2) learns the document vectors (P).
- Given many context words from the document and the document ID, the model is trained to predict the next word after the context.
- After training, as an indirect result of the prediction task, the document vectors (P) capture the topic of the document and act as a memory of what is missing from the context.

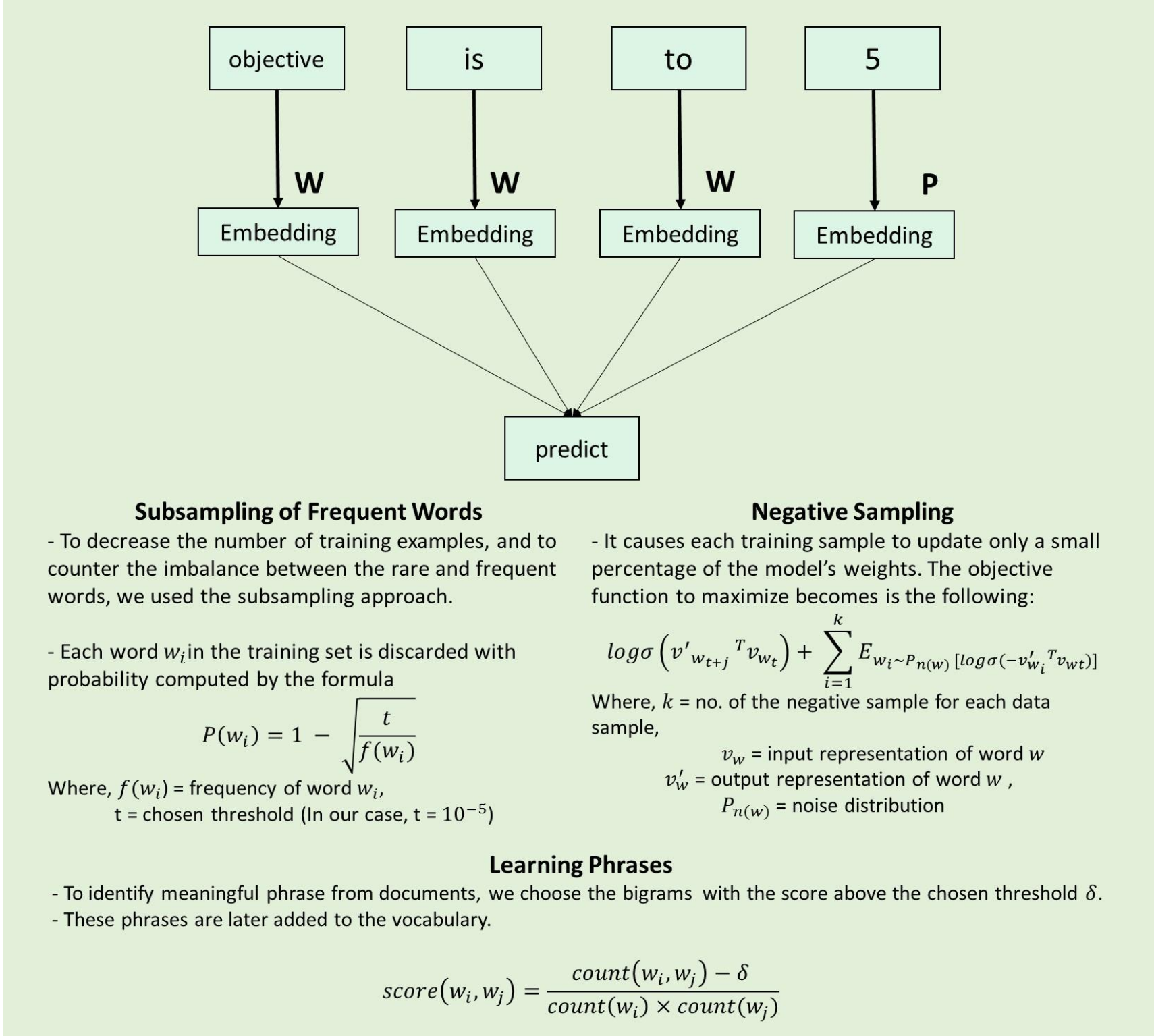


Figure 2: Paragraph Vectors

Results and Evaluation

We tested each vectorization method on two tasks using ArXiv papers as the dataset, which contains about 1 million published research papers.

1. Document Clustering

Our primary investigation was how well each approach could capture semantic information to group related documents. We made groups of related documents by running queries on Google Scholar and finding documents that appeared together in the results for the same query. We vectorized the documents using each approach and used truncated singular value decomposition to project them onto a 2-dimensional space for visualization.

Figures 3 and 4 show the resulting 2D latent spaces of Paragraph Vectors and TF-IDF, respectively. The color of each data point represents the query for which it was a result.

A qualitative analysis shows a clear advantage for Paragraph Vectors as opposed to TF-IDF. Paragraph Vectors clearly groups the documents belonging to the same queries, whereas TF-IDF mixes the groups for different queries.

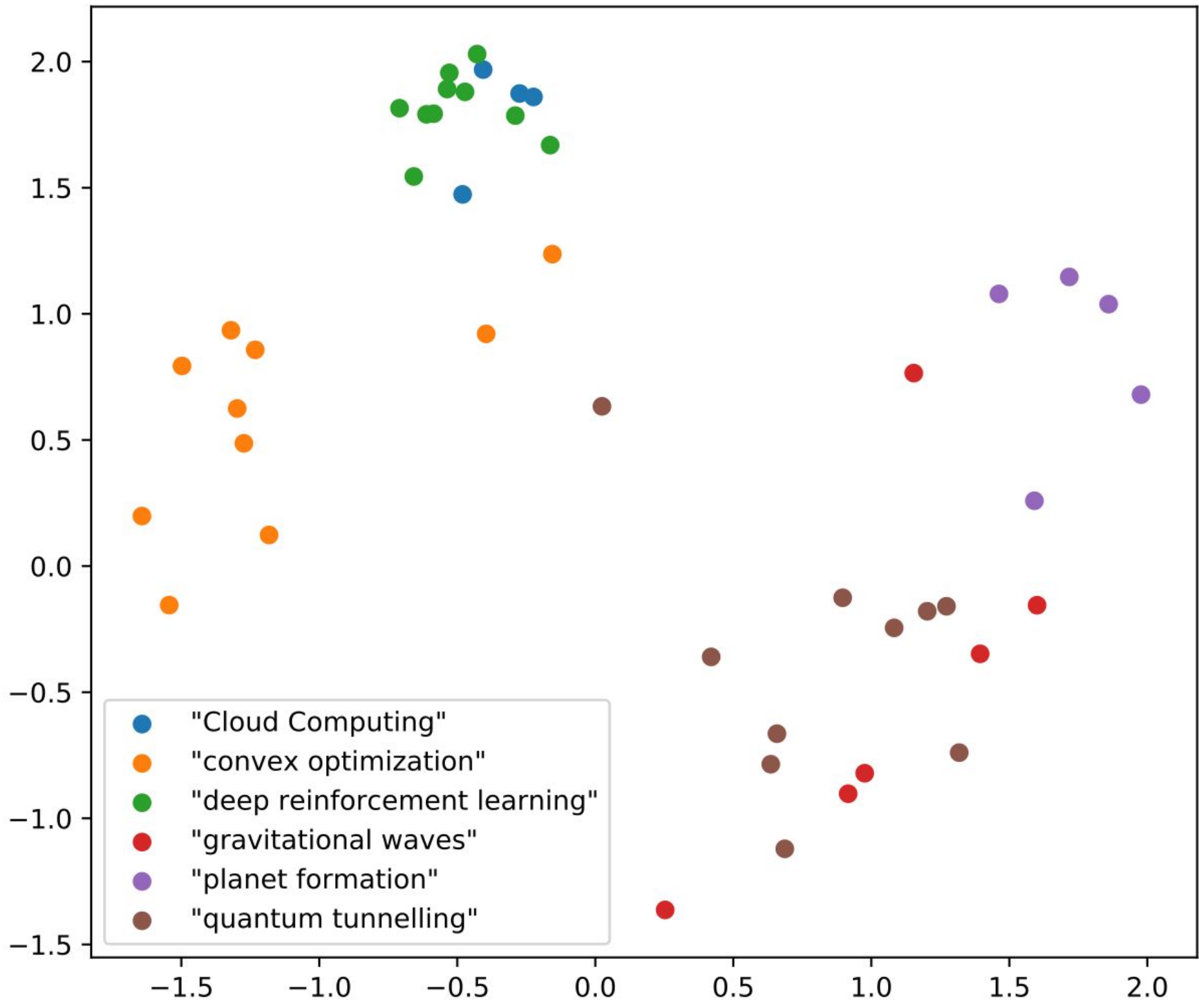


Figure 3: Document Clusters using Paragraph Vectors

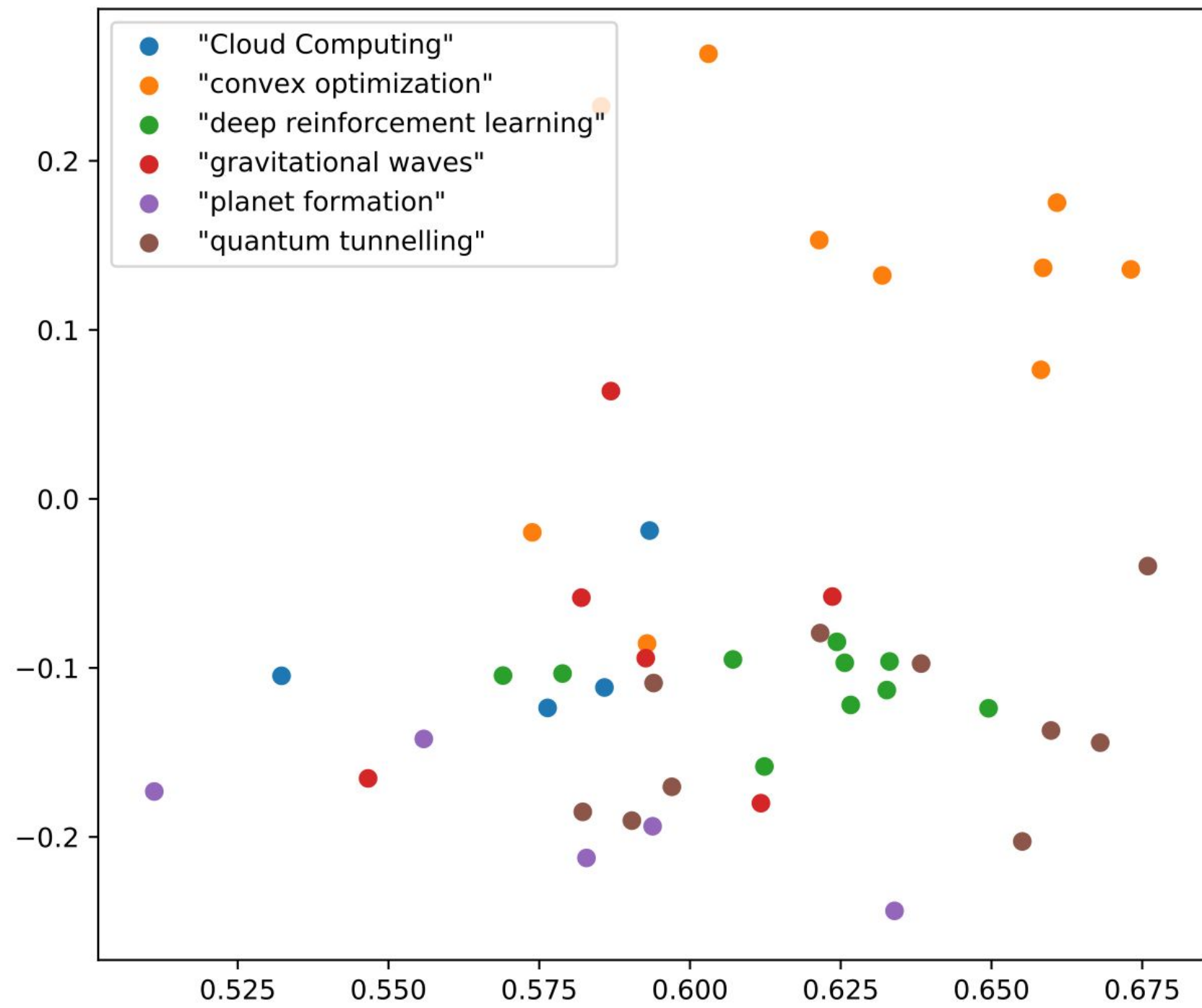


Figure 4: Document Clusters using TF-IDF vectors

2. Document Searching

We tested each vectorization method for searching over the Cranfield corpus, a set of 1400 aeronautics abstracts with expert-judged correct answers on a set of 225 queries. Both the documents and queries were vectorized, and the search algorithm was simply a nearest-neighbor search in the vector space using Cosine Similarity as the distance metric.

The effectiveness of each method was measured using a precision vs recall (PR) curve. This curve is produced by taking the top K results for all K between 1 and N, and measuring what fraction of the top K results are true positives (precision) and what fraction of the true relevant results are in the top K results (recall). We used the area under the PR curve (AUC-PR) as our quantitative performance metric.

We found that TF-IDF achieved the better result on this corpus, with an AUC-PR of **0.32**. The Paragraph Vector approach has an AUC-PR of **0.14**. This is a surprising result because it comes despite Paragraph Vectors achieving superior performance on the clustering task.

We believe the reason for this result is related to the length of the texts in this task. Both the queries and documents were much shorter than the ArXiv documents, which meant the Paragraph Vector model did not have sufficient contextual information to learn the semantics of the texts.

Extended Work

We intend to experiment with several modifications to the document vectorization discussed here:

1. Encode word root information into the model:

Instead of treating each word as completely distinct, the model will likely learn better if it knows that words like "compute" and "computing" have similar spellings.

2. Incorporate task-related information:

If the goal is to do document search, the model can learn better vector representations by directly taking into account how the vectors affect the quality of search results.

3. Learn via heirarchy:

Textual information can be thought of as a heirarchy, starting at characters, which make up words, forming phrases, then sentences, paragraphs, and so on. Skip-gram and Paragraph Vectors only learn at the word/phrase level, which likely limits their ability to achieve a full understanding of the text.

References

- Dai Andrew, Olah Christopher, Le Quoc , "Document Embedding with Paragraph Vectors", (Google) arXiv:1507.07998v1, 2015.
- Le Quoc, and Mikolov Tomas. "Distributed representations of sentences and documents." *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*. 2014.
- Mikolov Tomas , Sutskever Ilya, Chen Kai, Corrado Greg, and Dean Jeffrey. "Distributed Representations of Words and Phrases and their Compositionality". *Proceedings of NIPS*, 2013.
- Mikolov Tomas, Chen Kai, Corrado Greg, and Dean Jeffrey. Efficient Estimation of Word Representations in Vector Space. In Proceedings of Workshop at ICLR, 2013.