Mapintel Project Report

David Silva, Prof. Fernando Bação

April 21, 2021

Abstract

Briefly summarize your previous work, goals and objectives, what you have accomplished, and future work. (100 words max) If you have a question, please use the help menu ("?") on the top bar to search for help or ask us a question.

Introduction

The Mapintel project aims at providing *Agência para o Investimento e Comércio Externo de Portugal* (AICEP) with a Competitive Intelligence (CI) tool to explore up to date articles from a myriad of national and international news sources, allowing for a new and interactive way of discovering information.

CI is concerned with gathering and analyzing information on any aspect of the business environment (competition, customers, legal framework, etc.) needed to support executives in strategic planning and decision-making. AICEP's mission is to promote Portuguese exports and to secure foreign direct investment, playing a major role in the economic development and job creation in Portugal. CI is a key component of AICEP's activity, which requires keeping track of current affairs and sift through the endless flow of news about markets, trade, industries, countries and politics.

Information Retrieval (IR) is defined as "finding material of an unstructured nature that satisfies an information need from within large collections" (Schütze et al., 2008, p. 1). This process is done daily by millions of users, for a multitude of reasons. The main interaction we have with IR is through web search engines, which allows a user to make ad hoc queries and receive relevant results. The IR process requires that the user defines a specific information need and is able to express it through a given query. This may result in a problem as often times the need cannot be framed as a query. This happens mainly when the user wants to go through a set of documents without any particular objective but to assess them and understand which information they contain, both as a whole and individually. This is what we denominate as *Document Exploration*. Document exploration usually involves a user reading through a large document collection and progressively gaining knowledge of the topics mentioned by each document. This activity is costly as the documents need to be analysed manually by the same individual. The cost of the task increases when we consider a dynamic collection of documents which is updated with daily or even hourly frequency. In this work, we look to facilitate the exploration of a dynamic document collection by proposing a system capable of semantically organize it, while providing a visual and interactive interface that captures the local and global patterns of the documents and their relationships. The system should help the user to quickly gain knowledge about the current state of the collection and easily satisfy any particular information need that might emerge.

The Vector Space model (VSM) (Schütze et al., 2008, p. 120-126) consists in representing a set of documents as vectors in a common vector space, while also allowing free text queries to be represented in this same space. The fundamental assumption of VSM is that similar documents will be placed close together in the vector space, whereas dissimilar documents will be far away. This model is essential for dealing with large document collections as it ranks the documents matching a query, providing only the most relevant results. This is contrasted by the Boolean Retrieval model (Schütze et al., 2008, p. 1-18), which given a Boolean expression of terms, can return a number of matching documents superior to the one a human user could possibly sift through. The VSM ranks each document d in decreasing order of their cosine similarity with a query q:

$$cosine(q,d) = \frac{\vec{V}(q).\vec{V}(d)}{|\vec{V}(q)||\vec{V}(d)|} \tag{1}$$

, where $\overrightarrow{V}(q)$ corresponds to the query vector and $\overrightarrow{V}(d)$ corresponds to the document vector. VSM can be easily extended to clustering and classification applications as these require text input to be represented as fixed-length vectors.

One of the most common fixed-length representations of text is bag-of-words (Harris, 1954), due to its simplicity, efficiency and often surprising accuracy. Term frequency-inverse document-frequency (TF-IDF) (Jones, 1972) is another representation which tries to correct some of the problems of bag-of-words by not giving the same importance to every feature. Both these models have two main disadvantages: the word order is lost, which means different sentences can have the same representation as long as they have the same word frequencies e.g. "I like maths but not physics" has the same representation as "I like physics but not maths", and semantics are ignored, which results in representations that don't account for the actual meaning of the sentence. To solve the later issue and in particular the inability of these representations to deal with synonymy (the same information can be described using different terms) and polysemy (a term can express several distinct meanings depending on its context), Deerwester et al. (1990) proposes the Latent Semantic Indexing (LSI). LSI consists in applying singular-value decomposition to construct a low-rank approximation of the term-document matrix, thus mapping both terms and documents into a low k-dimensional orthogonal semantic space that preserves, to some extent, the relative distances between vectors in the original space. Once this mapping is obtained, new documents and queries can be represented in the semantic space and similarities can be computed as in the VSM. The representations provided achieve great savings in both storage and query time. Intuitively, LSI is able to capture the semantic structure in the data because it is forced to squeeze the terms/ documents down to a low k-dimensional space bringing together terms with similar co-occurrences. Therefore, the documents (and queries) are no longer represented by terms, but by the underlying concepts referred by the terms.

Despite the good empirical results in IR (Dumais et al., 1988; Berry et al., 1995), LSI does not provide a mathematical explanation for its effectiveness, leaving us with unanswered questions regarding the extent to which it captures the semantic structure of a corpus (Papadimitriou et al., 2000). Hofmann (1999) addresses this issue by proposing Probabilistic LSI (pLSI), a statistical latent variable model that defines a proper generative model for general co-occurrence data. pLSI models each word $w \in W = \{w_1, ..., w_M\}$ in a document $d \in D = \{d_1, ..., d_N\}$ as a sample from a document-specific word distribution $P(w|d)$ that is characterized as a mixture model (Murphy, 2012, p. 337-380) of K latent factors $Z = \{z_1, ..., z_K\}$ with mixture weights $\theta = P(z|d)$, where $z \in Z$. More concretely, pLSI is defined as a joint probability model:

$$P(d, w) = P(d)P(w|d), where \tag{2a}$$

$$P(w|d) = \sum_{k=1}^{K} P(w|z_k)P(z_k|d) \tag{2b}$$

The pLSI model posits that observation pairs (d, w) are generated independently i.e. "bag-of-words" assumption, and that documents and words are conditionally independent given a topic z. The model parameters, $P(d)$, $P(w|z)$ and $P(z|d)$ are estimated by maximizing the log-likelihood function:

$$\mathcal{L} = \sum_{n=1}^{N} \sum_{m=1}^{M} n(d_n, w_m) log P(d_n, w_m) \tag{3}$$

, where $n(d, w)$ gives the term frequency of w in d. The optimization algorithm used is a variation of the Expectation Maximization (EM) (Murphy, 2012, p. 348-369). pLSI represents each document in the training set with the mixture weights $P(z|d)$ and thus queries (and new documents) need to be folded-in by computing $P(z|q)$, where q denotes a query made by the user, using EM iteration. Then, the low dimensional representations are used by VSM to rank each document according to a query.

Latent Dirichlet Allocation (LDA) (Blei et al., 2003) is a generative probabilistic model of a corpus. Contrarily to pLSI, LDA can naturally assign probabilities to previously unseen document, thus being a well-defined generative model of documents. Also, in pLSI the number of parameters grows linearly with the size of the corpus, which makes it prone to overfitting, whereas in LDA the number of parameters is independent of it. Both these improvements are achieved by treating the topic mixture weights θ as a K-dimensional Dirichlet hidden random variable rather than a $K \times N$ matrix of parameters, where the mixture weights are linked to the training documents D. According to LDA, a document d can be generated as follows:

$$P(d|\alpha,\beta) = \int P(\theta|\alpha)\left(\prod_{m=1}^{M} P(w_m|\theta,\beta)\right) d\theta, where \tag{4a}$$

$$P(w_m|\theta,\beta) = \sum_{k=1}^{K} P(w_m|z_k,\beta)P(z_k|\theta) \tag{4b}$$

, where $\theta \sim Dir(\alpha)$, α is a K-vector with components $\alpha_i > 0$ and β is a $K \times V$ matrix, where $\beta_{ij} = P(w^j = 1|z^i = 1)$ and V is the vocabulary size. θ can be seen as a document-specific probability distribution over the topics, α is the Dirichlet distribution parameter and each row of β gives a topic-specific probability distribution over the vocabulary. By assuming that documents in a corpus are independent, we can obtain the probability of a corpus D:

$$P(D|\alpha,\beta) = \prod_{n=1}^{N} P(d_n|\alpha,\beta) \tag{5}$$

LDA is a three-level hierarchical Bayesian model, thus the generation of corpus can be divided into three stages: sample the corpus-level parameter α and β once, sample the document-level variables θ_n once per document and sample the word-level variables z_nm and w_nm once for each word in each document.

While the the probabilistic approaches mentioned above can represent a document as a vector of topic probabilities, thus being frequently referred as topic modeling, there is a more recent and equally common alternative to represent documents as a dense fixed-length vector that utilizes a neural network architecture and an unsupervised task to learn the representations. This approach is usually called neural embedding modeling and is able to produce document embeddings (i.e. document vector representations) that account for word order and capture the semantics of the words, two of the main issues of the classical BOW model. A prime example of this approach is the Paragraph vector model (Le and Mikolov, 2014) (or Doc2Vec model), an unsupervised algorithm that learns both word and document vectors by minimizing the error of predicting the next word in a paragraph (a variable-length piece of text) given the paragraph and previous word vectors. The neural network is parameterized by the word and document vectors which are trained using stochastic gradient descent and backpropagation. This architecture is inspired by the CBOW (Continuous BOW) and Skip-gram word embedding models in Mikolov et al. (2013).

With the recent advent of the Transformer architecture, significant improvements were made in several tasks related with Natural Language Processing (NLP) Vaswani et al. (2017). This new architecture is based solely on attention mechanisms, providing parallelization capabilities and significant improvements in training time. Also, the Transformer can more easily learn long-range relationships between terms in the input sequence than the pre-existing Recurrent Neural Network (RNN) () and Convolutional Neural Network (CNN) () architectures. BERT (Bidirectional Encoder Representation from Transformers) is a Transformer based pre-trained language model that can be easily fine-tuned to obtain state-of-the-art level performance for multiple NLP tasks (Devlin et al., 2019). BERT differences itself from the other pre-trained language models by utilizing a "masked language model" pre-training task that uses both left and right context to predict randomly masked terms from the input sequence. In addition, the model is also pre-trained on a "next sentence prediction" task that jointly pre-trains text-pair representations.

Despite the versatility and good performance obtained by BERT in various sentence classification and sentence-pair regression tasks, there isn't a natural way to obtain an independent sentence embedding since by design BERT takes as input two sentences separated by a special token and outputs the sentence token representations. Most importantly, "finding the most similar pair in a collection of 10,000 sentences requires about 50 million inference computations (65 hours) with BERT" (Reimers and Gurevych, 2019), making this an inadequate approach for obtaining document embeddings to be used for semantic textual similarity (STS) and in particular for IR. Sentence-BERT or SBERT (Reimers and Gurevych, 2019) solves this issue by using a siamese network structure that adds a pooling layer to the output of a pre-trained BERT model. By passing a single sentence as input to BERT and then aggregating its outputs with the pooling layer, we can derive a fixed sized sentence embedding, reducing to around 5 seconds the time taken to perform the task described above. The weights are fine-tuned with Natural Language Inference (NLI) data, resulting in embeddings that can capture the semantics of each sentence and be used with VSM.

Related work

Our work is inspired by previous previous applications related to visual analytics, SOM and exploration of large collection of documents.

Ji et al. (2019) proposes a system for visual exploration of neural document embeddings to gain insights into the underlying embedding space and to promote the utilization in prevalent IR applications. t-SNE (Van der Maaten and Hinton, 2008) is used to project the high-dimensional data onto a 2D surface. This technique is able to capture both local and global structure from the high-dimensional data in an efficient and reliable way. In this work, the documents are embedded using the Paragraph Vector model (Le and Mikolov, 2014). The system visualizes neural document embeddings as a configurable document map and enables guidance and reasoning, facilitates to explore the neural embedding space, identifies salient neural dimensions (semantic features) per task and domain interest and supports advisable feature selection (semantic analysis) along with instant visual feedback to promote IR performance. Overall, the system provides users with insights and confidence in neural document embeddings given their black-box nature.

Lafia et al. (2019) uses SOM and Latent Dirichlet Allocation (LDA) (Blei et al., 2003) to convey the relatedness of research themes in a multidisciplinary university library. LDA is a generative probabilistic model of a corpus. The basic idea is that documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words. That said, each document is embedded in a vector space of N dimensions, corresponding to the number of topics selected. SOM produces a landscape for exploring the topic space and provides users with an overview of the document collection and the ability to navigate (discover items of interest), change the level of detail, select individual documents and discover relationships between documents.

Kaski et al. (1998) presents the WEBSOM system - a system that organizes a textual document collection using a SOM-based graphical map display that provides an overview of the collection and facilitates interactive browsing. Kohonen (2013) revisits the topic and provides some enhancements. Here, the documents are represented with a TF-IDF weighting (Jones, 1972) and a random projection is used to reduce the dimensionality of the vector space, while preserving the similarity structure between documents. A SOM is constructed and each document is mapped into the node that best represents it. This provides exploring, searching and filtering capabilities. For example, when a node in the map is clicked, the titles of the corresponding documents and eventually some additional information such as descriptive words are presented. Also, the map is described by an automatic annotation procedure explained in Lagus and Kaski (1999), which helps to understand the semantics encoded in each map region. The user can also perform queries either using a set of keywords or a descriptive sentence. The query is then mapped into the reduced vector space and matched with the most similar documents and/or nodes. A zooming feature is also present which allows the user to explore specific regions of the map with finer detail.

Henriques et al. (2012) proposes the GeoSOM suite, a tool for geographic knowledge discovery using SOM. This tool is designed to integrate geographic information and aspatial variables in order to assist the geographic analyst's objectives and needs. The tool provides several dynamically linked views of the data consisting of a geographic map, a u-matrix, component plate plots, hit-map plots, parallel coordinate plots, boxplots and histograms. These views and their connection allows for an interactive exploration of the data.

Methods

The methodology adopted in this project can be summarized by Figure 1. First, we set up a data collection process to automatically absorb the continuous flow of news articles, then some pre-processing was applied to the data to make it usable by the embedding models and imp rove the quality of the produced document vectors. A SOM was applied to build a two-dimensional grid that is able to represent the multi-dimensional input space and therefore, the properties and relationships of the articles. Finally, a document exploration interface was built to provide the user the ability to explore and query the articles. The code developed for the project can be accessed at github.com/DavidSilva98/mapintel_project.

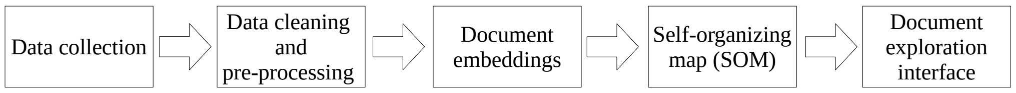
Data collection
Data cleaning and pre-processing
Document embeddings
Self-organizing map (SOM)
Document exploration interface

Figure 1: Methodology

Data Collection

In this project we decided to focus on how NLP and particularly sentence embeddings could help in organizing, exploring and retrieving text documents. Since the objective is to explore news articles, we used a REST API [1] to continuously retrieve English articles from multiple international sources several times a day. The API calls are performed through the AWS Lambda service [2] and the articles, as well as their metadata, are stored using the MongoDB Atlas cloud database service [3]. One particularly useful feature of the metadata is the category of the article. This can be one of the following: business, entertainment, general, health, science, sports or technology. It is important to note that the API we are using imposes some limitations that affect the data collection such as the articles being provided with 1 hour delay, having a maximum of 100 requests per day and the content of the article being truncated to 200 characters. We also developed a simple Optical Character Recognition (OCR) pipeline using the Tesseract OCR engine [4]. The purpose of this pipeline was to integrate internal documents from AICEP in our application, however we haven't focused on these documents so far.

Data Cleaning and Pre-processing

After loading the document corpus from the database, we concatenated the title, description and content fields in order to obtain longer and more informative documents. We proceed to clean the documents by removing non-textual patterns such as URLs and HTML tags and by removing non-English articles which are still present despite the filter applied previously. We split the document corpus into train and test set to allow for downstream unbiased performance assessments. A pre-process pipeline is applied on both corpora [5] to reduce the dimensionality of the vocabulary. The pipeline consists of removing stop words (very frequent words that are irrelevant), lowering the letter's case, removing accents and punctuation, applying stemming [6] and removing words that appear in just one document or in more than 90% of the documents.

Document Embeddings

Once the corpus is pre-processed, we encoded each document as a single vector of information. Here we used several approaches and compared them using a standardized evaluation design. As a baseline we used a Bag-of-Words (BOW) approach (Harris, 1954), which represents each document as a token histogram of the top 10000 most frequent tokens. We also used a Term Frequency - Inverse Document-Frequency (TF-IDF) approach (Jones, 1972) which tries to adjust for the fact that some words appear more frequently in general by offsetting each token frequency with the number of documents that contain the token. In these 10000 tokens, we included n-grams containing one to three words so word order information could be included in the embedding. We also used the proposed models by Le and Mikolov (2014), namely the Distributed Memory Model of Paragraph Vectors (PV-DM) and the Distributed Bag of Words version of Paragraph Vector (PV-DBOW), to learn continuous distributed fixed-length vector representations from variable-length pieces of text. These models are trained on the task of predicting words in a paragraph by looking at the context of the target word, which is encoded in the words and paragraph vectors. These vectors are the parameters of the model and are adjusted using stochastic gradient descent and backpropagation. One of the disadvantages of these methods is that to infer the embeddings of new documents, the model needs to train them which can be a problem when dealing with user queries.

[1] newsapi.org
[2] A serverless compute service that lets you run code without provisioning or managing servers
[3] A fully-managed cloud database service
[4] github.com/tesseract-ocr/tesseract
[5] Note: the pipeline is fitted only on the train set to avoid data leakage
[6] Term normalization process that removes the morphological and inflectional endings from words

Model Evaluation

To evaluate the quality of each approach, we used the corresponding embeddings and categories of each document in various tasks, similarly to some of the ones in Conneau and Kiela (2018). One of the approaches was training a logistic regression model using the embedding vectors of the train corpus to predict the article category. The accuracy of the classifier on the test corpus was used to evaluate the embeddings as the model is kept constant over the different approaches. We realized that, even though the model is kept constant, we cannot control for the interactions between each feature set and the logistic regression, which means the scores obtained don't completely isolate the embeddings performance. For this reason, a second task was proposed which consisted in classifying whether each unique test document pair belonged to the same category, based solely on their cosine similarity. The cosine similarities were converted to a range between 0 and 1 using the min-max transformation and the average binary cross-entropy over all unique pairs of documents was obtained. We also looked at the t-SNE projections of the embeddings to visualize how well they captured the semantics of the documents. Our hope was that if some semantic properties were captured, the embedding vectors would have been grouped by their categories. In the results section we will analyze how the different embedding models produced different evaluations.

Self-Organizing Map

After comparing the several embedding models, the SOM model was applied on the train embeddings of the best performing model using a fork of the SOMPY package (Moosavi et al., 2014). The grid is composed of regular hexagons as they are "visually much more illustrative and accurate, and are recommended" (Kohonen, 2013). Also, we selected the lengths of the horizontal and vertical dimensions of the grid to comply with the relation of the two largest principal components, while providing enough nodes to adequately represent the details and clusters of the input space. The oblong regular arrays have the advantage over the square ones of guaranteeing faster and safer convergence in learning. The nodes were initialized as a regular, two-dimensional sequence of vectors taken along a hyperplane spanned by the two largest principal components of the input space, providing faster ordering and convergence (Kohonen, 2001). Finally, we relied on the minimization of the quantization error (the mean distance of every data point to the corresponding best-matching unit) to select the remaining hyper-parameters of the model.

Document Exploration Interface

We extracted the fitted codebook matrix and we utilized it to build a U-matrix (Ultsch, 1993) to visualize the structures of the high-dimensional input space. By adding an interactive component to this visualization, we were able to encode several details and information within each unit of the matrix such as the distance from its neighbor units, the number of observations allocated to it and also some aggregated information from these observations. We also used the approach in Lagus and Kaski (1999) to characterize regions of the U-matrix by optimal positioning of descriptive keywords. These words function as landmarks i.e. navigational cues that help in maintaining a sense of location during the exploration of the map. Finally, we integrated the remaining components of the interface such as the search bar, a pane to preview the articles retrieved by the query and some more dynamic components to facilitate the user interaction.

Results

Present an observation (results), then explain what happened (analysis). Each paragraph should focus on one aspect of your results. In that same paragraph, you should interpret that result. In other words, there should not be two distinct paragraphs, but instead one paragraph containing one result and the interpretation and analysis of this result. Here are some guiding questions for results and analysis:

When describing your results, present your data, using the guidelines below:

- What happened? What did you find?
- Show your experimental data in a professional way. Refer to Grammar Guidelines for Reports for details on formatting. Be sure to reference figures before they appear in your paper (see Figure

2). Be sure to do the same for tables (see Table 1). For a good tool for making tables, go to tablesgenerator.com.

Figure 2: Captions go beneath figures.

Table 1: Captions go above tables.

Parameter	Symbol	Value
Residence Time	θ	90 s
Hydraulic Gradient	G	500 s^{-1}

After describing a particular result, within a paragraph, go on to connect your work to fundamental physics/chemistry/statics/fluid mechanics, or whatever field is appropriate. Analyze your results and compare with theoretical expectations; or, if you have not yet done the experiments, describe your expectations based on established knowledge. Include implications of your results. How will your results influence the design of AguaClara plants? If possible provide clear recommendations for design changes that should be adopted. Show your experimental data in a professional way using the following guidelines:

- Why did you get those results/data?
- Did these results line up with expectations?
- What went wrong?
- If the data do not support your hypothesis, is there another hypothesis that describes your new data?

Discussion

Study comparison with other studies. What were the limitations?

Conclusions

Explain what you have learned and how that influences your next steps. Why does what you discovered matter to AguaClara? Make sure that you defend your conclusions. (this is conclusions, not opinions!)

Future Work

For implementing the query feature of the system, the query is embedded in the same space of the news article corpus and the distance with each SOM unit is computed. The query is then matched with the closest SOM unit and the documents allocated to that unit are retrieved. This approach is fast since there are many fewer units than documents. The unit's documents are ranked by computing the distance between them and the query. The search quality is expected to not decrease significantly as long as the Mean Quantization Error (MQE) (i.e. the mean euclidean distance each input vector to its BMU) remains low.

We plan to provide a zooming capability on the SOM U-matrix so the user can explore specific regions of the map in detail. There are two ways we have been discussing on how to implement this: one possibility would be to allow the user to select a specific unit or group of units on the map and then provide a projection of the underlying documents using either t-SNE (Van der Maaten and Hinton,

2008) or UMAP (McInnes et al., 2018); a second possibility would be to allow the user to digitally zoom in on the U-matrix, just like it is done in Kaski et al. (1998). An appealing attribute of this option is the preservation of the landmark labels, which are updated according to the zooming of the map.

There's also some discussion on how to integrate release date information on the article's representation. This would allow the documents to be organized not only according to their semantics but also according to their release date. This could also improve the query results as the users are most likely interested on current information. Another feature related to release date would be to relate documents in a time line, allowing a specific subject to be tracked through time.

We would also like to improve the data collection pipeline since we are relying directly on NewsAPI free subscription which has some limitations already described. This would require a substantial effort since web scrapping would most likely be the necessary solution. This approach would provide us with the full article content and would allow us to collect articles as soon as they are released. Multilingual articles could also be collected and integrated into the system by using multilingual embeddings models such as Conneau et al. (2019).

Some more ideas to explore consist on: build a single or multi article summary feature, to provide a brief resume of the content of a specific article or of a specific SOM unit (collection of articles); add a news article feed based on individual user viewing history. If we plan to expand the application to multiple users, an implicit feedback collaborative filtering (Hu et al., 2008) approach could be used.

Some research on understanding the document embedding dimensions' would also be interesting as these usually present a correlation structure which captures the latent semantical topics of the document collection as seen in Ji et al. (2019). This would provide the user with the necessary confidence on the neural document embeddings that is lacking because of the black-box nature of these models.

References

Berry, M. W., Dumais, S. T., and O'Brien, G. W. (1995). Using linear algebra for intelligent information retrieval. *SIAM review*, 37(4):573–595.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., and Stoyanov, V. (2019). Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.

Conneau, A. and Kiela, D. (2018). Senteval: An evaluation toolkit for universal sentence representations. *arXiv preprint arXiv:1803.05449*.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs]*.

Dumais, S. T., Furnas, G. W., Landauer, T. K., Deerwester, S., and Harshman, R. (1988). Using latent semantic analysis to improve access to textual information. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 281–285.

Harris, Z. S. (1954). Distributional structure. *Word*, 10(2-3):146–162.

Henriques, R., Bacao, F., and Lobo, V. (2012). Exploratory geospatial data analysis using the GeoSOM suite. *Computers, Environment and Urban Systems*, 36(3):218–232.

Hofmann, T. (1999). Probabilistic latent semantic indexing. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 50–57.

Hu, Y., Koren, Y., and Volinsky, C. (2008). Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*, pages 263–272. Ieee.

Ji, X., Shen, H.-W., Ritter, A., Machiraju, R., and Yen, P.-Y. (2019). Visual exploration of neural document embedding in information retrieval: Semantics and feature selection. *IEEE transactions on visualization and computer graphics*, 25(6):2181–2192.

Jones, K. S. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation.*

Kaski, S., Honkela, T., Lagus, K., and Kohonen, T. (1998). WEBSOM–self-organizing maps of document collections. *Neurocomputing*, 21(1-3):101–117.

Kohonen, T. (2001). Software Tools for SOM. In *Self-Organizing Maps*, pages 311–328. Springer.

Kohonen, T. (2013). Essentials of the self-organizing map. *Neural networks*, 37:52–65.

Lafia, S., Last, C., and Kuhn, W. (2019). Enabling the Discovery of Thematically Related Research Objects with Systematic Spatializations. In *14th International Conference on Spatial Information Theory (COSIT 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Lagus, K. and Kaski, S. (1999). Keyword selection method for characterizing text document maps.

Le, Q. and Mikolov, T. (2014). Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196. PMLR.

McInnes, L., Healy, J., and Melville, J. (2018). Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426.*

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *arXiv:1301.3781 [cs].*

Moosavi, V., Packmann, S., and Vallés, I. (2014). SOMPY: A Python Library for Self Organizing Map (SOM).

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective.* MIT press.

Papadimitriou, C. H., Raghavan, P., Tamaki, H., and Vempala, S. (2000). Latent semantic indexing: A probabilistic analysis. *Journal of Computer and System Sciences*, 61(2):217–235.

Reimers, N. and Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *arXiv:1908.10084 [cs].*

Schütze, H., Manning, C. D., and Raghavan, P. (2008). *Introduction to Information Retrieval*, volume 39. Cambridge University Press Cambridge.

Ultsch, A. (1993). Self-organizing neural networks for visualisation and classification. In *Information and Classification*, pages 307–313. Springer.

Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-SNE. *Journal of machine learning research*, 9(11).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention Is All You Need. *arXiv:1706.03762 [cs].*