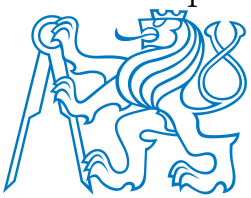
Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science



Research Trend Detection

Shuhailo Oleksii

2023

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Shuhailo Oleksii** Personal ID number: **478014**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Computer Science**

Study program: **Open Informatics**

Specialisation: **Cyber Security**

II. Master’s thesis details

Master’s thesis title in English:

Research Trend Detection

Master’s thesis title in Czech:

Detekce výzkumných trendů

Guidelines:

The work aims the goal to create a system capable to identify and present research trends in a given segment of science. Research papers and web pages are considered as input data.
Recommended steps of the solution:
1/ Create a review of methods dealing with research trend detection based on research papers, web pages and their metadata.
2/ Select a suitable set of methods and create appropriate processing chain. If a suitable implementation of a selected method does not exists, develop it.
3/ Verify experimentally the performance of the processing chain. Use available datasets or create it yourself.
4/ Discuss the results. Propose future improvements.

Bibliography / sources:

• Mitchell, Ryan. 2015. Web Scraping with Python. O’Reilly Media, Inc.
• Hapke, Hannes, Cole Howard, and Hobson Lane. 2019. Natural Language Processing in Action. Simon and Schuster.
• Bird, Steven, Ewan Klein, and Edward Loper. 2009. Natural Language Processing with Python. O’Reilly Media, Inc.
• Kubat, Miroslav. 2018. Introduction to Machine Learning. S.L.: Springer International Pu.
• Angelov, Dimo: Top2Vec: Distributed Representation of Topics, 2020, arXiv.

Name and workplace of master’s thesis supervisor:

Ing. Radek Mařík, CSc. Department of Telecommunications Engineering FEE

Name and workplace of second master’s thesis supervisor or consultant:

Date of master’s thesis assignment: **10.02.2023** Deadline for master's thesis submission: **26.05.2023**

Assignment valid until: **22.09.2024**

Ing. Radek Mařík, CSc. Supervisor’s signature	Head of department’s signature	prof. Mgr. Petr Páta, Ph.D. Dean’s signature

III. Assignment receipt

The student acknowledges that the master’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master’s thesis, the author must state the names of consultants and include a list of references.

. | Date of assignment receipt

Student’s signature

© ČVUT v Praze, Design: ČVUT v Praze, VIC

Declaration

This master's thesis is a product of my mind. Wherever there is a mansion of someone's work, all efforts are made to make it clear. My Master's thesis was done under the supervision of Marik Radik.

Signature

Date

Abstract

The rapid growth of scientific knowledge and the ever-expanding volume of research publications pose significant challenges in identifying emerging trends and understanding the evolving research landscape. This diploma thesis investigates the field of research trend detection, aiming to explore existing technologies and methods, and develop a novel pipeline to detect and analyze research trends based on article titles.

The thesis is organized into four main chapters. The first chapter provides an introduction to the research topic, highlighting the importance of trend detection in the context of scientific advancements and knowledge discovery. It also outlines the objectives and research questions that guide the investigation.

The second and third chapters review methods and algorithms employed in research trend detection. Various techniques, such as clustering, topic modeling, and natural language processing, are discussed, along with their strengths and limitations. The third chapter goes into detail about articles and papers addressing challenges in trend detection within the research domain, providing valuable insights into the field's current state.

The fourth chapter details the design and implementation of a novel pipeline for research trend detection. The pipeline leverages article titles as the primary input and incorporates various techniques, including topic extraction, document embedding, and clustering. The assembly of these components forms a cohesive framework capable of identifying and analyzing research trends. Additionally, the pipeline addresses challenges related to parameter selection and dataset variability.

The fifth chapter showcases the results obtained from applying the developed pipeline to the arXiv dataset. It demonstrates the pipeline's capability to detect and visualize research trends over time.

Contents

Chapter 1

Introduction

Science and technology are advancing at a rapid pace, with new discoveries and innovations constantly emerging. This progress is bringing about significant changes in society, creating new opportunities and challenges for researchers, practitioners, and decision-makers. As a result, staying up-to-date with the latest trends and developments in science and technology has become more important than ever.

With the increasing volume of scientific papers being published, it is becoming more difficult to identify emerging trends and important discoveries. Researchers and practitioners are often overwhelmed by the sheer amount of information available, which can lead to missed opportunities for collaboration and delayed progress in research. Therefore, there is a growing need for more effective methods for detecting emerging trends in scientific literature.

The consequences of failing to recognize emerging trends in the scientific literature can be significant. For example, researchers who are unaware of new findings or approaches may continue to use outdated methods or pursue research directions that are no longer relevant. This can result in wasted resources, missed opportunities for scientific breakthroughs, and even negative impacts on public health and safety.

Existing methods for detecting emerging trends in the scientific literature have their own strengths and limitations. For example, bibliometric analysis can provide valuable insights into the citation patterns and impact of scientific papers, but it may not capture the full scope of emerging trends. Text mining can identify relevant keywords and topics in scientific papers, but it may not be able to capture the context or nuances of scientific language. Social network analysis can reveal the connections and collaborations among researchers, but it may not be able to capture the full range of emerging trends.

In this paper, we propose a new approach for detecting emerging trends in scientific papers. Our method is based on machine learning algorithms and natural language processing techniques and aims to overcome some of the limitations of traditional methods. Specifically, our method can capture the complex relationships between different scientific concepts and provide a more accurate and comprehensive way to identify emerging trends in scientific literature.

In the next section, we will provide a more detailed review of the literature on detecting emerging trends in scientific papers. We will also discuss the strengths and limitations of existing methods and identify the gaps in the literature that our proposed method aims to address. In section three, we will describe our method in detail, including the data sources, algorithms, and evaluation metrics we used. In section four, we will present the results of our experiments, comparing the performance of our method with traditional methods on a large dataset of scientific papers. Finally, in section five, we will discuss the implications of our findings, highlight the potential applications of our method, and suggest directions for future research.

Chapter 2

Tools and Methods for trend detection

2.1 Community detection

2.1.1 Louvain algorithm

Community detection is a fundamental task in network analysis, with applications ranging from social network analysis to biology and transportation network analysis. It involves identifying groups of nodes in a network graph that are more densely connected to each other than to nodes outside the group. These groups are known as communities, and their identification can provide insights into the structure and function of the network.

The Louvain method is a popular algorithm for detecting communities in networks. The basic idea behind the Louvain method is to optimize a quality function to identify communities in a network. This quality function measures the degree to which nodes within a community are densely connected to each other and sparsely connected to nodes in other communities.

The Louvain method works in two phases. In the first phase, the algorithm optimizes the modularity of the network at the node level to identify communities.

Modularity is a measure of the quality of a community partition. It measures the difference between the number of edges within communities and the expected number of edges in a random network with the same degree sequence. A higher modularity score indicates a better community partition.

The modularity Q is defined as follows:

$$Q = \frac{1}{2m} * \sum \sum [A_{ij} - \frac{(k_i * k_j)}{2m}] * \delta(c_i, c_j) \tag{2.1}$$

where A_{ij} is the weight of the edge between nodes i and j, k_i, and k_j are the degrees of nodes i and j, respectively, m is the sum of all edge weights in the network, c_i, and c_j are the communities to which nodes i and j belong, respectively, and $\delta(c_i, c_j)$ is the Kronecker delta, which equals 1 if $c_i = c_j$ and 0 otherwise.

The Louvain method uses a greedy algorithm to optimize modularity. The algorithm starts with each node in its own community and then iteratively moves nodes between communities to maximize the increase in modularity. Specifically, for each node i, the algorithm calculates the change in modularity $\triangle Q$ that would result from moving i to each of its neighboring communities and selects the move that results in the largest increase in $\triangle Q$. If no move results in a positive increase in $\triangle Q$, node i remains in its current community. The algorithm continues to iterate over all nodes until no further increase in modularity is possible.

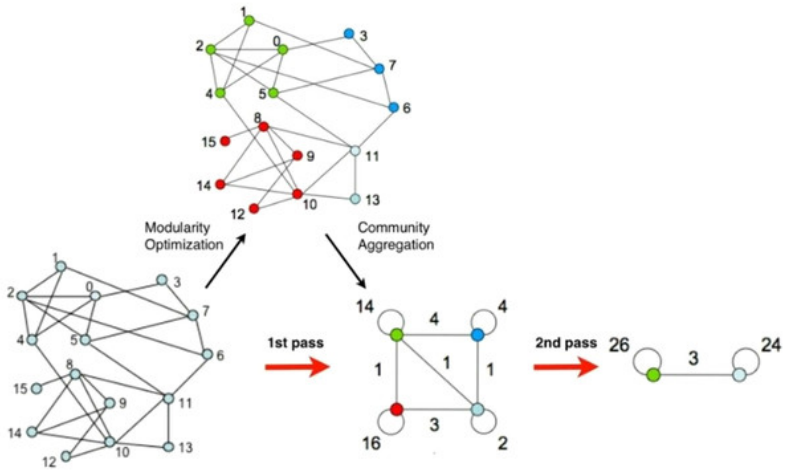


Figure 2.1: Process of community detection for Louvain algorithm from Mourchid, El Hassouni, and Cherifi in 2015.

The node-level optimization phase of the Louvain method is computationally efficient and can handle large networks with millions of nodes. However, it can also lead to the detection of small, spurious communities due to noise or random fluctuations in the network. To address this issue, the Louvain method includes a community-level optimization phase, which is described next.

In the second phase, the Louvain method constructs a new network where nodes are communities and edges represent the sum of weights between nodes in different communities. This step is referred to as community aggregation and leads to the identification of larger communities. The community aggregation process is iterative, meaning that communities identified in the previous iteration become nodes in the next iteration. The algorithm repeats this process until a maximum level of aggregation is reached.

Once the community aggregation is complete, the Louvain method applies the node-level optimization algorithm to the new network, which corresponds to optimizing modularity at the community level. This process is similar to the node-level optimization described in the previous section but is applied to communities instead of individual nodes. The algorithm iteratively moves communities between neighboring communities, attempting to maximize the modularity score.

During the community-level optimization process, the Louvain method applies a second quality function called the participation coefficient. The participation coefficient measures how well a node is connected to communities other than its own and is used to prevent isolated nodes or communities from forming. This quality function ensures that nodes or communities with sparse connections are not considered separate communities.

Similar to the node-level optimization process, community-level optimization continues until no further increase in modularity is possible. At this point, the algorithm terminates, and the resulting community partition is considered final. The Louvain method produces a hierarchical clustering of the network, which can be visualized as a dendrogram. The dendrogram shows the different levels of community aggregation and the corresponding modularity scores at each level.

The Louvain method for community detection has become a widely used algorithm in the field of network analysis. However, there are several limitations to the method that researchers should be aware of. Next, we will discuss these limitations and some extensions of the Louvain method that have been proposed to address them.

The Louvain method is sensitive to the size of the network. As the size of the network increases, the computation time and memory requirements of the algorithm become

prohibitive. This is because the number of communities and edges in the network grows exponentially with the size of the network.

The Louvain method is also sensitive to the structure of the network. Specifically, the method tends to produce communities that are relatively balanced in size, which can be problematic in networks with highly skewed degree distributions.

The Louvain-igraph algorithm is an extension of the Louvain method that has been implemented in the igraph software package. This extension addresses some of the limitations of the original method by incorporating a faster and more memory-efficient algorithm for computing modularity.

The multilevel Louvain method is an extension of the Louvain method that aims to produce communities that are more balanced in size. This extension works by recursively applying the Louvain method to smaller subgraphs of the original network and then merging the resulting communities into larger communities. The multilevel Louvain method has been shown to be effective in producing communities that are more balanced in size and that have higher modularity than those produced by the original Louvain method.

In summary, while the Louvain method for community detection is a powerful tool for analyzing networks, it is important to be aware of its limitations. Researchers should carefully consider the size and structure of their networks when using this method and may want to consider using extensions such as the Louvain-igraph algorithm or the multilevel Louvain method to address some of these limitations.

2.2 Artificial Neural Networks

2.2.1 Recurrent Neural Networks

Recurrent Neural Networks (RNN) are artificial neural networks developed as an extension to dense networks to work more efficiently with sequential data.

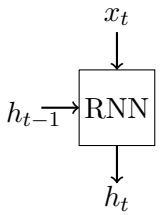


Figure 2.2: The one iteration RNN, with $\boldsymbol{x_t}$ as an input, $\hat{\boldsymbol{h}}_{\boldsymbol{t}}$ as an output and $\boldsymbol{h}_{t-1}$ as a previous or hidden state.

RNN's computes the output from every input without considering the connection between previous and new computations, which makes it hard for this type of network to work with sequences. On the other hand, Recurrent Networks are made of recurrent connections, which allows them to take the output as additional information from previous computations and pass it to the next layer. Performing such forwarding requires those types of networks somehow to save such information. It is done with the extra variable that every recurrent layer posses that is also called as the hidden state.

LSTM, which stands for Long Short-Term Memory, is a type of artificial neural network architecture that is particularly effective at modeling sequential data, such as time series or natural language text. LSTMs were introduced in 1997 by Hochreiter and Schmidhuber as an improvement over earlier recurrent neural network (RNN) architectures.

The key innovation of LSTMs is their ability to selectively remember or forget information over long periods of time. This is accomplished through the use of a set of specialized memory cells that are connected to each other and to the rest of the network through a set

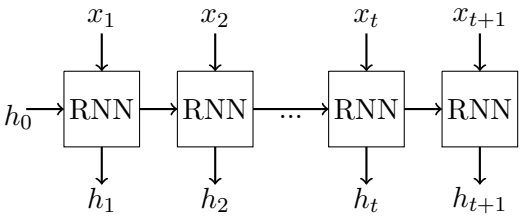


Figure 2.3: Unfolded RNN of the input sequence vector $\boldsymbol{x} = [x_1, x_2, ..., x_{t+1}]$, where h_0 is the initial hidden state and every next computed output h_t is then forwarded to the next step.

of gates. These gates, which are themselves implemented as neural networks, determine how much of the input data should be remembered, forgotten, or output at each time step. The LSTM architecture consists of several key components:

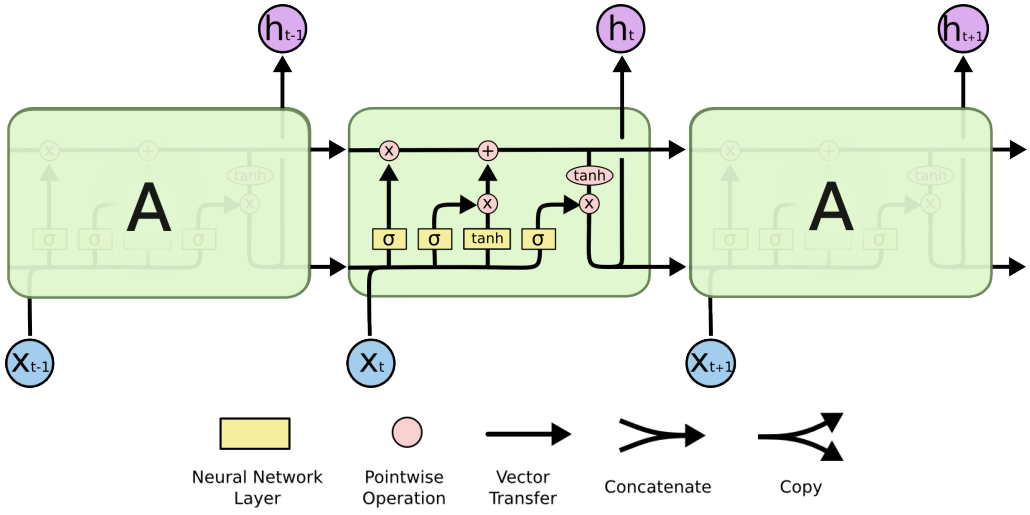


Figure 2.4: Descriptive picture with a notation of the LSTM cell work from Olah in 2015.

Memory cells: these are the primary building blocks of the LSTM network, and they are responsible for storing information over long periods of time. Each memory cell has a set of internal states, including a cell state and a hidden state, that are updated at each time step.

Input gate: this gate determines how much of the new input data should be added to the cell state. It takes as input the current input data and the previous hidden state and outputs a value between 0 and 1 that represents the fraction of the input data to be added.

Forget gate: this gate determines how much of the current cell state should be forgotten. It takes as input the current input data and the previous hidden state and outputs a value between 0 and 1 that represents the fraction of the cell state to be retained.

Output gate: this gate determines how much of the current cell state should be output. It takes as input the current input data and the previous hidden state and outputs a value between 0 and 1 that represents the fraction of the cell state to be output.

Hidden state: this is the final output of the LSTM network, and it is calculated based on the current cell state and the output gate.

During training, the weights and biases of the LSTM network are adjusted using backpropagation through time, which involves calculating the gradient of the loss function with respect to the network parameters at each time step and then propagating these gradients

backward through the network.

LSTMs have been shown to be particularly effective at a wide range of tasks, including speech recognition, natural language processing, image captioning, and more. Their ability to selectively remember or forget information over long periods of time makes them well-suited for tasks that involve complex, structured data with long-term dependencies.

2.3 Natural language processing tools

2.3.1 Word embedding

Word embedding is a language modeling technique used in natural language processing (NLP) that aims to represent words as numerical vectors in a continuous vector space. The concept of word embeddings was introduced in 2003 by Bengio et al. in their paper [4]. Word embeddings have since become a popular technique in NLP, used for a variety of tasks such as sentiment analysis, machine translation, and text classification.

The idea behind word embeddings is to create a vector representation of words that captures their semantic and syntactic relationships. The vectors are learned through a process called training, in which an algorithm analyzes a large corpus of text to determine the context in which words are used. The resulting vectors can then be used to perform a wide range of NLP tasks.

One of the main papers that contributed to the development of word embeddings is [15] by Mikolov et al. 2013. The paper introduced the word2vec algorithm, which is a neural network-based method for learning word embeddings from large amounts of text data. The algorithm uses a neural network to predict a word's context (i.e., the words that appear nearby in a sentence) and learns the vector representation of each word in the process. This paper introduced two variations of the word2vec algorithm, called continuous bag-of-words (CBOW) and skip-gram. CBOW predicts a target word based on the context words, while skip-gram predicts context words based on a target word. These two methods have different strengths and weaknesses and can be used in different NLP tasks depending on the specific needs of the task.

Another influential paper is [21] by Pennington, Socher, and Manning in 2014. This paper introduced the GloVe algorithm, which is a count-based method for learning word embeddings. The algorithm constructs a co-occurrence matrix of word pairs and learns the vector representation of each word by optimizing a weighted least-squares objective function.

Since these seminal papers, there have been numerous other advances in word embedding research, including the development of contextualized embeddings such as ELMo and BERT, which take into account the context in which a word appears to generate a more nuanced representation of the word. Word embeddings have become a fundamental tool in NLP and continue to be an active area of research.

Overall, word embeddings have had a significant impact on the field of NLP, allowing for more accurate and efficient processing of natural language text. They have also paved the way for the development of more advanced NLP techniques, such as deep learning-based models.

2.3.2 Introduction Word2Vec

Word2vec is a popular natural language processing technique that generates word embeddings, which are dense, high-dimensional vectors that represent the semantic and syntactic properties of words. Word2vec was developed by Tomas Mikolov and his team at Google in 2013, and it is based on a neural network architecture that learns word embeddings from large amounts of text data.

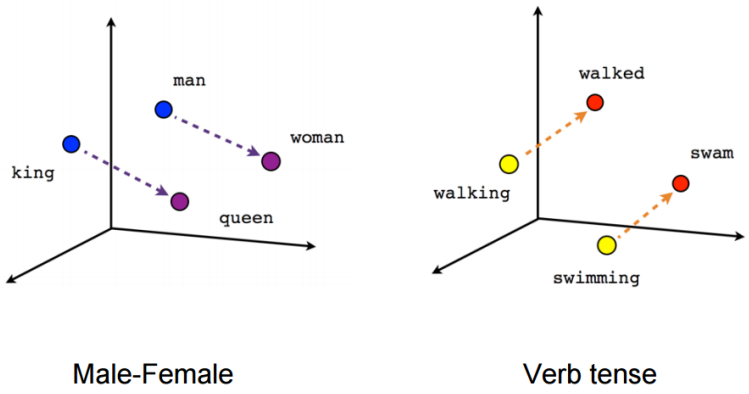


Figure 2.5: Word2vec visualization from Kung-Hsiang in 2018

Word2vec has become an essential tool in natural language processing because it enables computers to understand the meaning of words in context. Before the advent of word2vec, natural language processing systems relied on handcrafted rules or shallow statistical methods to analyze text data. These approaches were limited in their ability to capture the complexity and nuances of natural language. With word2vec, however, computers can learn the meaning of words from large amounts of text data without being explicitly programmed with linguistic rules. This makes it possible to apply natural language processing techniques to a wide range of applications, such as machine translation, text classification, sentiment analysis, and speech recognition. Additionally, word2vec has been used in various research fields, such as computational linguistics, psychology, and neuroscience, to study human language processing and cognitive mechanisms.

2.3.3 What is word2vec?

Word embeddings are numerical representations of words that capture their semantic and syntactic properties. Word embeddings are dense, high-dimensional vectors that are learned by a neural network trained on a large corpus of text data. Each word is represented by a unique vector, which is composed of real-valued numbers. The distance between the vectors for two words indicates the similarity between them; for example, the vectors for "cat" and "dog" will be closer together than the vectors for "cat" and "car." Word embeddings can be used in a variety of natural language processing applications, including language modeling, sentiment analysis, and text classification.

Word2vec can be trained using two different models: continuous bag-of-words (CBOW) and skip-gram. The CBOW model takes as input a sequence of context words and learns to predict the target word in the center of the sequence. The skip-gram model, on the other hand, takes as input a target word and predicts the context words that surround it. Both models use a single hidden layer neural network to generate word embeddings, but they differ in terms of their input and output layers.

In the CBOW model, the input layer receives a concatenated vector of one-hot encoded context words, and the output layer produces a probability distribution over the vocabulary, indicating the likelihood of each word being the target word. In contrast, the skip-gram model takes a one-hot encoded target word as input and produces a probability distribution over the context words.

The choice of model depends on the specific natural language processing task and the characteristics of the text data being used. In general, the skip-gram model is better suited for infrequent words or words with multiple meanings, while the CBOW model is better

suited for frequent words or words with consistent meanings in context.

The basic structure of a word2vec neural network consists of an input layer, a hidden layer, and an output layer. The input layer receives one-hot encoded vectors representing the input words, and the output layer produces a probability distribution over the vocabulary, indicating the likelihood of each word being the context or target word. The hidden layer contains the word embeddings, which are learned during the training process.

The size of the hidden layer determines the dimensionality of the word embeddings. The larger the size of the hidden layer, the more complex and detailed the representation of the words. During the training process, the weights of the neural network are updated using backpropagation and stochastic gradient descent. The objective of the training process is to maximize the probability of predicting the target word given the context words (for CBOW) or vice versa (for skip-gram).

This is achieved by minimizing the negative log-likelihood of the target word given the context words (or vice versa). The choice of hyperparameters, such as the learning rate and the number of epochs, can significantly affect the performance of the word2vec model. Once the training is complete, the word embeddings are extracted from the hidden layer and can be used for a variety of natural languages processing tasks, such as sentiment analysis, text classification, and language translation.

2.3.4 How does word2vec work?

The first step in using word2vec is to preprocess the text data. This involves cleaning the text data and converting it into a format that can be used by the word2vec model. The goal of preprocessing is to remove noise and irrelevant information from the text data and to retain only the meaningful words and phrases.

Tokenization is the process of breaking up a text document into individual words or tokens. This is done by splitting the text on whitespace or punctuation. For example, the sentence "The quick brown fox jumps over the lazy dog" can be tokenized into individual words: "The", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog".

Stop words are common words that occur frequently in a language, such as "the", "and", "is", and "of". These words do not carry much meaning on their own and can be removed from the text data to reduce noise. Stop word removal can be done using a predefined list of stop words or by calculating the frequency of words in the text and removing the most frequent words.

Stemming and lemmatization are techniques used to reduce words to their base form. Stemming involves removing the suffixes from words to reduce them to their root form, while lemmatization involves mapping words to their base form using a dictionary. For example, the word "running" can be stemmed to "run", and the word "jumps" can be lemmatized to "jump".

Text normalization involves converting text to a standard format by removing special characters, converting all characters to lowercase, and replacing numbers with their textual representation. For example, the sentence "John's email address is john123@gmail.com" can be normalized to "johns email address is john number one two three at gmail dot com".

Once the text data has been preprocessed, it is converted into a numerical representation that can be used by the word2vec model. This is done using a technique called one-hot encoding, which represents each word in the text as a binary vector with a 1 in the position corresponding to the index of the word in the vocabulary and 0s elsewhere.

The size of the vocabulary depends on the number of unique words in the text data. For large datasets, the vocabulary can be too large to fit into memory, so a technique called subsampling can be used to randomly discard some of the frequent words. This helps to reduce the computational complexity of the model and improves its performance.

Once the text data has been preprocessed and converted into a numerical representation, it can be used to train the word2vec model. The word2vec model is a neural network that learns to predict the context words given a target word, or vice versa.

There are two main types of word2vec models: the continuous bag-of-words (CBOW) model and the skip-gram model. The CBOW model predicts the target word given the context words, while the skip-gram model predicts the context words given the target word. The skip-gram model is more popular than the CBOW model, as it tends to perform better on smaller datasets and is more robust to rare words.

The architecture of the skip-gram model consists of an input layer, a hidden layer, and an output layer. The input layer represents the target word as a one-hot encoded vector, while the output layer represents the context words as probability distributions over the vocabulary. The hidden layer represents the learned representation of the target word.

During training, the skip-gram model learns to maximize the probability of observing the context words given the target word, or vice versa. This is done by adjusting the weights of the neural network using the backpropagation algorithm. The backpropagation algorithm calculates the gradient of the loss function with respect to the weights of the neural network and updates the weights accordingly.

The loss function used in the skip-gram model is the negative log-likelihood of the observed context words, given the target word. This is equivalent to minimizing the cross-entropy between the predicted probability distribution and the true probability distribution of the context words.

The training process involves iterating over the text data multiple times, or epochs, and updating the weights of the neural network after each epoch. The size of the context window, or the number of words on either side of the target word, is a hyperparameter that can be tuned to optimize the performance of the model.

Once the skip-gram model has been trained, the learned representation of each word in the vocabulary can be used to perform various natural language processing tasks, such as text classification, sentiment analysis, and machine translation. This is because the learned representation captures the semantic and syntactic properties of each word, allowing similar words to have similar representations in the vector space.

After the word2vec model has been trained, the learned representation of each word in the vocabulary can be used to generate word vectors. Word vectors are dense, low-dimensional representations of words that capture the semantic and syntactic properties of each word.

The most common way to generate word vectors from the trained word2vec model is to use the learned weights of the hidden layer as the word vectors. The weights of the hidden layer represent the learned representation of each word in the vocabulary, which captures the contextual information of each word in the text corpus.

The dimensionality of the word vectors is a hyperparameter that can be set during training. Typically, word vectors are generated with a dimensionality between 100 and 300, as higher dimensionalities may lead to overfitting and lower dimensionalities may lead to loss of information.

Once the word vectors have been generated, they can be used for various natural language processing tasks. One common use case is to measure the semantic similarity between words. This is done by computing the cosine similarity between the word vectors of two words. Words that have similar meanings will have higher cosine similarity values than words that have different meanings.

Another use case is to perform vector arithmetic on the word vectors. This involves adding or subtracting the word vectors of two or more words to generate a new word vector that represents a semantically related concept. For example, the word vector for "king" minus the word vector for "man" plus the word vector for "woman" results in a

new word vector that is closest to the word vector for "queen".

In addition to semantic similarity and vector arithmetic, word vectors can be used for various other natural language processing tasks, such as text classification, sentiment analysis, and machine translation. These tasks can be performed using various machine learning algorithms that take the word vectors as input features.

In summary, generating word vectors from the trained word2vec model involves using the learned weights of the hidden layer as the word vectors. The dimensionality of the word vectors is a hyperparameter that can be set during training. The word vectors can be used for various natural language processing tasks, such as measuring semantic similarity, performing vector arithmetic, and training machine learning models for text classification, sentiment analysis, and machine translation.

2.3.5 Applications of word2vec

Word2vec has become a popular tool in natural language processing (NLP) due to its ability to generate high-quality word embeddings that capture the semantic and syntactic properties of words. Word embeddings generated by word2vec can be used for a variety of NLP tasks, such as text classification, sentiment analysis, machine translation, and information retrieval.

Text classification is the task of assigning predefined categories to text documents. Word2vec can be used to generate word embeddings that capture the context of each word in the text corpus. These embeddings can be used as input features for machine learning algorithms such as support vector machines (SVMs) or neural networks to classify text documents into different categories.

Sentiment analysis is the task of determining the sentiment polarity of text, such as positive or negative. Word2vec can be used to generate word embeddings that capture the semantic meaning of words in the text corpus. These embeddings can be used as input features for machine learning algorithms such as logistic regression or convolutional neural networks (CNNs) to predict the sentiment polarity of text.

Machine translation is the task of translating text from one language to another. Word2vec can be used to generate word embeddings for both the source and target languages. These embeddings can be used to train machine learning models such as sequence-to-sequence models or transformer models to perform machine translation.

Information retrieval is the task of retrieving relevant documents from a large text corpus based on a user's query. Word2vec can be used to generate word embeddings that capture the semantic meaning of words in the text corpus. These embeddings can be used to perform similarity matching between the user's query and the text documents in the corpus.

In addition to these tasks, word2vec can also be used for other NLP applications such as named entity recognition, relation extraction, and summarization. These applications require understanding the context and relationships between words in a text corpus, which can be achieved using word embeddings generated by word2vec.

One of the primary applications of word2vec is in the field of information retrieval. The goal of information retrieval is to retrieve relevant documents from a large text corpus based on a user's query. Word2vec can be used to generate word embeddings that capture the semantic meaning of words in the text corpus, which can be used to perform similarity matching between the user's query and the text documents in the corpus.

In traditional information retrieval systems, the user's query is matched against the keywords in the text documents using a bag-of-words model. This approach considers each word in the query and the text document as independent units and does not consider the relationships between words in the text corpus. This often leads to poor retrieval performance, as the semantic meaning of words is not captured.

Word2vec can be used to address this limitation by generating word embeddings that capture the semantic meaning of words in the text corpus. These embeddings can be used to represent each word in the query and the text documents as a vector in a high-dimensional space. The similarity between the query and each text document can then be computed as the cosine similarity between the corresponding vectors.

One of the key advantages of using word2vec for information retrieval is that it can handle synonyms and polysemous words. Synonyms are words that have the same or similar meanings, while polysemous words have multiple meanings depending on the context. Traditional information retrieval systems often struggle to handle these types of words, as they rely on exact keyword matching. However, word2vec can capture the semantic relationships between words, including synonyms and polysemous words, which can improve retrieval performance.

Another advantage of using word2vec for information retrieval is that it can handle out-of-vocabulary words. Out-of-vocabulary words are words that are not present in the vocabulary of the word2vec model, which can be a common problem in information retrieval systems. However, word2vec can generate embeddings for out-of-vocabulary words by using the embeddings of the surrounding words in the text corpus.

Another important application of word2vec is in the field of recommendation systems. Recommendation systems are used to suggest products or services to users based on their preferences and behavior. Word2vec can be used to generate embeddings for user and item profiles, which can be used to calculate the similarity between users and items and make personalized recommendations.

In traditional recommendation systems, user and item profiles are represented as vectors of features, such as demographic information or product attributes. However, these features may not capture the complex relationships between users and items or the semantic meaning of the items. Word2vec can address this limitation by generating embeddings that capture the semantic meaning of items and user behavior in the context of the entire corpus of user-item interactions.

To generate embeddings for user and item profiles, the user-item interactions can be used as the training data for the word2vec model. The user profiles can be represented as a bag-of-words model of the items they have interacted with, while the item profiles can be represented as a bag-of-words model of the users who have interacted with the item. The word2vec model can then be trained on this data to generate embeddings for the items and users.

Once the embeddings have been generated, they can be used to calculate the similarity between users and items. For example, the similarity between a user's embedding and an item's embedding can be computed as the cosine similarity between the two vectors. The items with the highest similarity to the user's embedding can then be recommended to the user.

One of the key advantages of using word2vec for recommendation systems is that it can capture the latent features of items and user behavior. Latent features are the hidden characteristics of items and users that are not directly observable, but influence their preferences and behavior. Word2vec can capture these features by analyzing the co-occurrence patterns of items and users in the training data and generating embeddings that capture the semantic meaning of these patterns.

Another advantage of using word2vec for recommendation systems is that it can handle cold-start and sparsity problems. Cold-start problems occur when there is little or no data available for a new user or item, making it difficult to make accurate recommendations. Sparsity problems occur when there are many items in the system but each user has interacted with only a small subset of them. Word2vec can address these problems by generating embeddings for the new users or items based on their interactions with the

existing items or users in the system.

In summary, word2vec is a versatile and impactful tool used in recommendation systems and information retrieval. It has the capability to capture latent features of items and user behavior, addressing challenges like cold-start and sparsity problems. By generating embeddings that capture semantic meaning, word2vec enables personalized and precise recommendations tailored to user preferences. Additionally, in information retrieval, it adeptly handles synonyms, polysemous words, and out-of-vocabulary terms, ultimately enhancing retrieval performance and delivering accurate and relevant search results.

2.3.6 Advantages and limitations of word2vec

Word2vec has several advantages over traditional approaches to natural language processing and machine learning.

The first key advantage is Semantic meaning. Word2vec can capture the semantic meaning of words and their relationships, which is critical for many natural language processing tasks. By analyzing the co-occurrence patterns of words in a large corpus of text, word2vec can generate embeddings that represent the meaning of the words and their relationships to other words. This can enable more accurate language modeling, text classification, and information retrieval.

The second one is Efficiency. Word2vec is highly efficient and can process large amounts of text data quickly and accurately. The use of neural networks and optimization techniques such as stochastic gradient descent allows word2vec to scale to very large datasets and generate embeddings for millions of words or phrases.

The third one is Flexibility. Word2vec is highly flexible and can be used for a wide range of natural language processing tasks, including language modeling, text classification, sentiment analysis, and recommendation systems. The ability to generate embeddings for both words and phrases allows word2vec to capture the complex relationships between words and their context.

The next one is Robustness. Word2vec is highly robust and can handle noisy and incomplete data, making it well-suited for real-world applications. The use of a sliding window approach and negative sampling allows word2vec to generate accurate embeddings even when the input data is noisy or incomplete.

The last one is Generalizability. Word2vec is highly generalizable and can be applied to a wide range of languages and domains. The ability to generate embeddings for both words and phrases allows word2vec to capture the unique characteristics of different languages and domains.

Despite its many advantages, word2vec also has some limitations that should be considered when using the model for natural language processing and machine learning tasks.

The first key limitation is Polysemy. Word2vec can struggle with words that have multiple meanings, or are polysemous. Since the model generates embeddings based on the co-occurrence patterns of words in a corpus of text, it can be challenging to disambiguate the different meanings of a word. For example, the word "bank" could refer to a financial institution or the edge of a river, and word2vec may struggle to capture the different meanings.

The second one is Contextual meaning. Word2vec can struggle to capture the contextual meaning of words. While the model is good at capturing the semantic meaning of words based on their co-occurrence patterns, it may not be able to capture the nuances of how words are used in specific contexts. For example, the word "hot" could refer to temperature or attractiveness, and the meaning of the word would depend on the context in which it is used.

The third one is Out-of-vocabulary words. Word2vec can struggle with out-of-vocabulary words, or words that are not present in the corpus of text used to train the model. This

can be particularly challenging for languages or domains with limited amounts of text data available for training. While techniques such as transfer learning and pre-training can help to address this issue, it remains a limitation of word2vec.

The next one is Lack of interpretability. The embeddings generated by word2vec are not easily interpretable by humans, which can make it challenging to understand how the model is making predictions or to debug errors in the model. While techniques such as visualization can help to make the embeddings more interpretable, this remains a limitation of word2vec.

The last one is Lack of transparency. The inner workings of the neural network used by word2vec can be opaque, making it difficult to understand how the model is generating embeddings and making predictions. This lack of transparency can be problematic for applications where interpretability and transparency are critical, such as legal or regulatory contexts.

In summary, word2vec has several advantages over traditional approaches to natural language processing and machine learning. It can capture the semantic meaning of words and their relationships, is highly efficient and flexible, can handle noisy and incomplete data, and is generalizable to a wide range of languages and domains.

On the other side, it has several limitations that should be considered when using the model for natural language processing and machine learning tasks. It can struggle with polysemy and contextual meaning, out-of-vocabulary words, lack of interpretability and transparency, and more. However, these limitations can often be addressed through careful model design and data preprocessing, and word2vec remains a powerful tool for a wide range of natural language processing tasks.

2.3.7 Universal sentence encoder

The Universal Sentence Encoder (USE) is a pre-trained machine learning model developed by Google that generates high-quality fixed-length numerical embeddings for input text. Unlike word embedding models such as word2vec, which generate vector representations for individual words, the Universal Sentence Encoder generates embeddings for entire sentences or paragraphs, enabling it to capture the overall meaning and context of a piece of text.

The Universal Sentence Encoder is based on a deep neural network architecture that combines both convolutional and recurrent layers to process text inputs. The model is trained on a large corpus of text data and can generate embeddings for a wide range of natural language tasks, including sentiment analysis, text classification, and natural language inference.

The Universal Sentence Encoder has become increasingly important in the field of natural language processing due to its ability to generate high-quality embeddings for entire sentences or paragraphs. This capability enables it to capture the overall meaning and context of a piece of text, making it useful for a wide range of natural language tasks.

One of the key advantages of the Universal Sentence Encoder is its ability to generate both sentence-level and paragraph-level embeddings. Sentence-level embeddings are designed to capture the meaning of individual sentences, while paragraph-level embeddings capture the overall meaning and context of longer passages of text. This makes the model useful for tasks such as text similarity, document clustering, and text generation.

The Universal Sentence Encoder is also pre-trained, meaning that it has already been trained on a large corpus of text data and can be used out-of-the-box for a wide range of natural language processing tasks. Additionally, the model can be fine-tuned on domain-specific data to further improve its performance on specific tasks.

Overall, the Universal Sentence Encoder is a powerful tool for natural language processing and machine learning tasks that require the processing of entire sentences or para-

graphs. Its ability to generate high-quality embeddings for input text makes it a valuable tool for a wide range of applications, including sentiment analysis, text classification, and natural language inference.

2.3.8 Architecture of the Universal Sentence Encoder

The Universal Sentence Encoder is based on a deep neural network architecture that combines both convolutional and recurrent layers to process text inputs. The architecture is designed to generate high-quality embeddings for input text by capturing both the meaning and context of sentences or paragraphs.

The neural network architecture of the Universal Sentence Encoder consists of two main components: the encoder and the projection layer. The encoder is responsible for processing the input text and generating initial embeddings, while the projection layer maps these embeddings to a fixed-length vector space.

The encoder component of the Universal Sentence Encoder consists of a hierarchical neural network architecture that processes input text at multiple levels of granularity. The first level consists of a character-based convolutional neural network (CNN) that processes individual characters to capture information such as spelling and capitalization. The output of the character-level CNN is then fed into a word-based CNN that processes individual words to capture information such as word order and syntax.

In addition to the CNN layers, the Universal Sentence Encoder also includes a recurrent neural network (RNN) layer that processes input text at the sentence level. The RNN layer uses a long short-term memory (LSTM) architecture to capture information about the meaning and context of sentences.

Once the input text has been processed by the encoder component of the Universal Sentence Encoder, the resulting embeddings are passed through a projection layer that maps the embeddings to a fixed-length vector space. The projection layer uses a combination of fully connected layers and a final normalization layer to generate high-quality embeddings that can be used for a wide range of natural language processing tasks.

Overall, the neural network architecture of the Universal Sentence Encoder is designed to capture both the meaning and context of input text, enabling it to generate high-quality embeddings for a wide range of natural language tasks. The combination of character-based and word-based CNN layers, along with the LSTM layer, provides the model with the ability to capture a wide range of linguistic features and nuances in input text.

The Universal Sentence Encoder is built on a complex neural network architecture that incorporates both convolutional and recurrent layers to process text inputs. These layers work in tandem to capture both the meaning and context of sentences, enabling the model to generate high-quality embeddings for a wide range of natural language processing tasks.

Convolutional Layers

The Universal Sentence Encoder uses convolutional layers to process both character and word-level features of input text. These layers are based on a convolutional neural network (CNN) architecture, which is commonly used in computer vision tasks to extract features from images. However, in the Universal Sentence Encoder, the CNN architecture is adapted for text inputs.

The character-based CNN layer processes individual characters to capture spelling and capitalization information, while the word-based CNN layer processes individual words to capture word order and syntax. By combining the output of these two CNN layers, the model is able to capture a wide range of linguistic features and nuances in input text.

Recurrent Layers

In addition to convolutional layers, the Universal Sentence Encoder also uses recurrent layers to process input text. Recurrent neural networks (RNNs) are commonly used in natural language processing tasks to model the sequential nature of language.

The Universal Sentence Encoder uses a long short-term memory (LSTM) architecture for the recurrent layer, which is a type of RNN that is designed to capture long-term dependencies in input text. The LSTM layer is responsible for capturing the meaning and context of sentences by processing input text at the sentence level.

The combination of convolutional and recurrent layers in the Universal Sentence Encoder enables the model to capture both the fine-grained details and the overall meaning of input text. The CNN layers capture individual characters and words, while the LSTM layer processes the sequence of sentences to capture the broader context. By combining these different layers, the model is able to generate high-quality embeddings that capture both the meaning and context of input text.

Overall, the complex neural network architecture of the Universal Sentence Encoder enables it to process input text in a sophisticated and nuanced way, capturing both fine-grained details and broader context. The combination of convolutional and recurrent layers ensures that the model can capture a wide range of linguistic features and nuances, making it a powerful tool for a wide range of natural language processing tasks.

The Universal Sentence Encoder generates high-quality embeddings for input text using a two-step process. The first step involves encoding the input text using the neural network architecture described in the previous section. The second step involves post-processing the encoded text to generate the final embeddings.

During the encoding step, the Universal Sentence Encoder processes input text using its neural network architecture, which incorporates both convolutional and recurrent layers. The input text is first tokenized into words and then processed by the character-based CNN layer and the word-based CNN layer. The output of these layers is then passed to the LSTM layer, which processes the sequence of sentences to capture the broader context of the input text.

Once the input text has been processed by the neural network architecture, the Universal Sentence Encoder generates two sets of embeddings: one set of word-level embeddings and one set of sentence-level embeddings.

Word-level embeddings capture the meaning of individual words in the input text, while sentence-level embeddings capture the overall meaning and context of the entire sentence. The word-level embeddings are generated by taking the output of the word-based CNN layer, while the sentence-level embeddings are generated by taking the final state of the LSTM layer.

After the encoding step is complete, the Universal Sentence Encoder performs post-processing on the generated embeddings to further refine their quality. This post-processing step involves normalization and projection of the embeddings into a lower-dimensional space.

Normalization involves scaling the embeddings to ensure that they have a consistent magnitude across different sentences. This is achieved by dividing each embedding by its L2 norm, which results in embeddings that lie on the surface of a unit sphere.

Projection involves mapping the embeddings from their original high-dimensional space to a lower-dimensional space. This is done using a projection matrix, which is learned during the training process of the Universal Sentence Encoder. The lower-dimensional embeddings generated by this projection step are more computationally efficient and can be used for a wider range of downstream tasks.

Overall, the embedding generation process of the Universal Sentence Encoder involves a combination of sophisticated neural network architecture and post-processing techniques. The resulting embeddings capture both the meaning and context of input text, and can be used for a wide range of natural language processing tasks.

2.3.9 Types of Embeddings Generated by Universal Sentence Encoder

The Universal Sentence Encoder generates high-quality sentence-level embeddings that capture the meaning and context of the entire sentence. These embeddings are particularly useful for a wide range of natural language processing tasks, such as text classification, semantic similarity detection, and sentiment analysis.

The sentence-level embeddings generated by the Universal Sentence Encoder are designed to be robust to a wide range of variations in sentence structure and wording. They are able to capture the underlying meaning and context of sentences even when the sentences use different words or grammatical structures to express the same idea.

The sentence-level embeddings are generated using a combination of convolutional and recurrent layers, which are trained on a large corpus of text to learn to capture the underlying meaning and context of input text. The final state of the LSTM layer is used to generate the sentence-level embeddings, which represent the overall meaning and context of the entire sentence.

One advantage of the sentence-level embeddings generated by the Universal Sentence Encoder is that they are relatively small in size, typically around 512 dimensions. This makes them computationally efficient to use in a wide range of downstream tasks, including those with limited computational resources.

Another advantage of the sentence-level embeddings is that they are capable of capturing a wide range of semantic and contextual information. For example, they can capture the sentiment of a sentence, the topics discussed in a sentence, and the relationships between different entities mentioned in the sentence.

Overall, the sentence-level embeddings generated by the Universal Sentence Encoder are a powerful tool for natural language processing tasks that require an understanding of the meaning and context of input text. They are robust, efficient, and capable of capturing a wide range of semantic and contextual information, making them a valuable resource for a wide range of applications.

In addition to generating high-quality sentence-level embeddings, the Universal Sentence Encoder is also capable of generating paragraph-level embeddings that capture the meaning and context of entire paragraphs of text. These embeddings can be useful for a variety of natural language processing tasks that require a more comprehensive understanding of the overall context and meaning of a piece of text.

The paragraph-level embeddings generated by the Universal Sentence Encoder are similar in architecture to the sentence-level embeddings, but they are trained on larger units of text and are designed to capture a broader range of contextual information. Specifically, the architecture includes a hierarchical pooling layer that aggregates information across multiple levels of the neural network, allowing the model to capture contextual information at different levels of granularity.

One advantage of the paragraph-level embeddings generated by the Universal Sentence Encoder is that they can capture the overall theme or topic of a piece of text, which can be useful for tasks such as topic modeling and text summarization. They can also be used to identify similarities and differences between different pieces of text, or to determine the overall sentiment or tone of a longer piece of text.

Like the sentence-level embeddings, the paragraph-level embeddings are computationally efficient and relatively small in size, typically around 512 dimensions. This makes them suitable for a wide range of downstream natural language processing tasks, including those with limited computational resources.

Overall, the paragraph-level embeddings generated by the Universal Sentence Encoder are a powerful tool for natural language processing tasks that require a deeper understanding of the overall context and meaning of a piece of text. They are capable of capturing a wide range of semantic and contextual information, and can be used in a variety of

applications, from topic modeling to sentiment analysis and beyond.

While both sentence-level and paragraph-level embeddings are generated by the Universal Sentence Encoder, there are some key differences between these two types of embeddings.

First and foremost, sentence-level embeddings are designed to capture the meaning and context of individual sentences, while paragraph-level embeddings are designed to capture the meaning and context of entire paragraphs of text. As a result, sentence-level embeddings tend to be more granular and detailed, while paragraph-level embeddings tend to be more comprehensive and broad.

One important difference between sentence-level and paragraph-level embeddings is the amount of contextual information that they capture. Because sentence-level embeddings are generated based on individual sentences, they are able to capture the nuances of sentence-level context, such as the role of each word in a given sentence and the relationship between different words in the sentence. In contrast, paragraph-level embeddings are generated based on larger units of text, and are designed to capture broader patterns and themes within the text, rather than the specific details of individual sentences.

Another important difference between sentence-level and paragraph-level embeddings is the size and dimensionality of the embeddings. Sentence-level embeddings are typically smaller and more compact than paragraph-level embeddings, with a size of around 512 dimensions. This makes them more computationally efficient and easier to work with in certain contexts. In contrast, paragraph-level embeddings are typically larger and more complex, with a size of up to 2048 dimensions. This larger size allows them to capture more comprehensive information about the overall context and meaning of a piece of text, but can also make them more challenging to work with in some contexts.

Ultimately, the choice between sentence-level and paragraph-level embeddings depends on the specific natural language processing task at hand. For tasks that require a more granular understanding of individual sentences, such as sentiment analysis or named entity recognition, sentence-level embeddings may be more appropriate. For tasks that require a broader understanding of the overall context and meaning of a piece of text, such as topic modeling or text summarization, paragraph-level embeddings may be more useful. In many cases, both types of embeddings may be used together to provide a more comprehensive understanding of a piece of text.

2.3.10 Applications of the Universal Sentence Encoder

The Universal Sentence Encoder (USE) is a versatile tool that can be used for a variety of natural language processing tasks. Here are some of the key applications of the USE:

Sentiment analysis is the process of identifying the sentiment or emotional tone of a piece of text. The USE can be used to generate sentence-level embeddings for text, which can then be fed into a machine learning model to predict the sentiment of the text.

Text classification is the process of categorizing text into one or more predefined categories. The USE can be used to generate sentence-level or paragraph-level embeddings for text, which can then be fed into a classification model to predict the category of the text.

Natural language inference (NLI) is the task of determining whether a hypothesis can be inferred from a given premise. The USE can be used to generate embeddings for the premise and hypothesis, which can then be fed into an NLI model to make a prediction.

Document clustering is the process of grouping similar documents together based on their content. The USE can be used to generate paragraph-level embeddings for documents, which can then be used to cluster the documents together.

Text generation is the process of generating new text that is similar in style and content to a given piece of text. The USE can be used to generate embeddings for the input text,

which can then be fed into a language generation model to generate new text.

In addition to these applications, the USE can also be used for other natural language processing tasks such as machine translation, information retrieval, and named entity recognition. Its versatility and effectiveness make it a valuable tool for a wide range of NLP tasks.

2.3.11 Advantages of the Universal Sentence Encoder

The Universal Sentence Encoder (USE) is a deep learning model developed by Google that generates high-quality embeddings for text data. These embeddings can be used to represent the overall meaning and context of sentences or paragraphs, which makes the USE a powerful tool for a wide range of natural language processing (NLP) tasks.

One of the main advantages of the USE is its ability to capture the overall meaning and context of a piece of text. Unlike traditional bag-of-words approaches, which consider each word independently, the USE takes into account the relationships between words in a sentence or paragraph. This allows the model to capture nuances of meaning that might be lost with other methods.

The embeddings generated by the USE are also of high quality. In particular, they are able to capture the semantic and syntactic relationships between words, which can be useful for a wide range of NLP tasks. The USE is also pre-trained on a large corpus of text data, which makes it well-suited for many applications without the need for additional training.

Another advantage of the USE is that it is a pre-trained and fine-tunable model. This means that it can be used out-of-the-box for many applications, but it can also be fine-tuned on specific datasets to improve performance on particular tasks. For example, a researcher might fine-tune the USE on a dataset of medical documents to improve its ability to identify medical terms and concepts.

Finally, the USE is available in TensorFlow, a popular deep learning framework. This means that it can be easily integrated into existing TensorFlow projects, which makes it a convenient choice for many researchers and developers.

In summary, the Universal Sentence Encoder is a powerful tool for natural language processing tasks. Its ability to capture overall meaning and context, high-quality embeddings, pre-trained and fine-tunable model, and availability in TensorFlow make it a popular choice for many applications.

2.3.12 Limitations of the Universal Sentence Encoder

The Universal Sentence Encoder (USE) is a powerful tool for generating high-quality embeddings that capture the semantic meaning of text at the sentence and paragraph level. However, like any technology, it has its limitations. In this section, we will explore some of the limitations of the USE.

One of the most significant limitations of the USE is that it is limited to the English language. While English is one of the most widely spoken languages in the world, there are many other languages that are equally important in different regions of the world. This limitation can be a significant obstacle for researchers and practitioners who work with text data in languages other than English.

Another limitation of the USE is that it requires a large amount of training data to generate high-quality embeddings. This can be a challenge for researchers and practitioners who work with text data in specialized domains or with limited amounts of data. In some cases, it may be difficult or impossible to generate high-quality embeddings using the USE due to a lack of training data.

Generating embeddings with the USE can be computationally intensive, particularly when working with large datasets. This can be a significant barrier for researchers and practitioners who have limited computational resources. Additionally, the time required to generate embeddings with the USE can limit the speed of some applications, particularly those that require real-time or near-real-time processing of text data.

In conclusion, while the USE is a powerful tool for generating high-quality embeddings that capture the semantic meaning of text, it is not without its limitations. These limitations can be a significant obstacle for researchers and practitioners who work with text data in specialized domains or with limited resources. However, despite these limitations, the USE remains one of the most widely used and effective tools for generating high-quality text embeddings.

2.3.13 Doc2Vec

In the field of natural language processing (NLP), document representation plays a crucial role in capturing the semantic meaning and context of textual data. Document representation techniques aim to convert unstructured text documents into numerical representations that can be effectively processed by machine learning algorithms. Traditional approaches to document representation include methods such as bag-of-words (BoW) and term frequency-inverse document frequency (TF-IDF), which represent documents as high-dimensional vectors based on the occurrence and frequency of words.

However, these traditional approaches have limitations in capturing the contextual information and semantic meaning of documents. They treat each word as an independent unit and do not consider the relationships between words or the overall structure of the document. This can lead to a loss of important information and hinder the performance of downstream NLP tasks such as text classification, clustering, and information retrieval.

Doc2Vec, also known as Paragraph Vector, is a powerful and popular approach for document representation in NLP. It was introduced by Le and Mikolov in 2014 as an extension of the Word2Vec model, which focuses on learning continuous vector representations for individual words. Doc2Vec takes into account the contextual information of words within a document and learns distributed representations, or embeddings, for entire documents.

The main idea behind Doc2Vec is to generate fixed-length numerical representations, or vectors, that capture the semantic meaning of documents in a continuous space. These document vectors are trained to encode the information about the words in the document as well as the document's overall context. By learning such representations, Doc2Vec enables the comparison, clustering, and classification of documents based on their semantic similarities.

Doc2Vec consists of two main architectures: the Distributed Memory (DM) model and the Distributed Bag of Words (DBOW) model. Both models are trained using a variant of stochastic gradient descent, which optimizes the parameters of the model based on the prediction of the surrounding words given the document or vice versa.

The key component of Doc2Vec is the paragraph vector, also referred to as the document vector. This vector represents the entire document and is trained to capture its semantic meaning. The paragraph vector is not constrained to any fixed dimensionality and can be adjusted based on the desired application or the size of the training dataset.

The Distributed Memory model is one variant of Doc2Vec that learns document vectors by considering the context of the target word within the document. It maintains a memory matrix, which acts as a global representation of the document, capturing the overall context. During training, the model takes into account the context words in the window surrounding the target word and the document vector to predict the target word.

The Distributed Bag of Words model is another variant of Doc2Vec that learns document vectors by ignoring the word order and focusing on the overall content of the

document. Unlike the DM model, DBOW does not consider the context of the target word but uses only the document vector to predict the target word.

The DM and DBOW models can be used individually or in combination, depending on the specific requirements of the task at hand. Both models have their own strengths and weaknesses, and their choice depends on the characteristics of the dataset and the nature of the problem.

In the next sections, we will delve deeper into the training process of Doc2Vec, the generation of document embeddings, the applications and use cases of Doc2Vec, as well as its advantages, limitations, and variations. By understanding these aspects, we can gain a comprehensive understanding of the capabilities and potential of Doc2Vec as a document representation technique in the field of natural language processing.

2.3.14 What is Doc2Vec?

Doc2Vec, also known as Paragraph Vector, is a machine learning algorithm that aims to generate distributed representations, or embeddings, for documents in the form of fixed-length numerical vectors. It was introduced by Tomas Mikolov et al. in 2014 as an extension of the Word2Vec model, which focuses on learning continuous vector representations for individual words. Doc2Vec builds upon the success of Word2Vec and extends it to capture the semantic meaning and contextual information of entire documents.

The purpose of Doc2Vec is to overcome the limitations of traditional document representation techniques, such as bag-of-words (BoW) and TF-IDF, which treat documents as collections of independent words and neglect the relationships between them. Doc2Vec, on the other hand, aims to capture the semantic meaning and context of documents by considering the interactions and dependencies between words within the document.

The key motivation behind Doc2Vec is to enable the comparison, clustering, and classification of documents based on their semantic similarities. By representing documents as continuous vectors in a high-dimensional space, Doc2Vec allows for efficient computation of document similarity and retrieval, making it suitable for a wide range of applications in natural language processing.

Doc2Vec consists of two main architectures: the Distributed Memory (DM) model and the Distributed Bag of Words (DBOW) model. Both models are trained using a variant of stochastic gradient descent to optimize the parameters and learn the document vectors.

The DM model maintains a memory matrix, which acts as a global representation of the document, capturing the overall context. It takes into account the context words in the window surrounding the target word and the document vector to predict the target word. This architecture aims to capture the word order and the context within the document.

On the other hand, the DBOW model ignores the word order and focuses on the overall content of the document. It uses only the document vector to predict the target word, treating the document as a bag of words. This architecture is simpler and computationally efficient, making it suitable for larger datasets.

Doc2Vec allows for unsupervised learning of document representations, meaning it does not require labeled data for training. However, it can also be used in a supervised manner, where the document vectors are fine-tuned for specific tasks such as document classification or sentiment analysis.

The document vectors generated by Doc2Vec encode the semantic meaning and context of the corresponding documents. These vectors are dense, continuous, and capture the relationships between words within the documents. By leveraging these document embeddings, various downstream NLP tasks can be performed more effectively, including text classification, clustering, information retrieval, recommendation systems, sentiment analysis, and question-answering systems.

At the core of Doc2Vec is the paragraph vector, also known as the document vector. The paragraph vector represents the entire document and captures its semantic meaning in a fixed-length numerical vector. Unlike traditional document representation techniques, which treat documents as collections of independent words, Doc2Vec considers the document as a whole and learns a continuous vector representation for it.

The paragraph vector is not limited to a predefined dimensionality and can be adjusted based on the desired application or the size of the training dataset. It aims to encode the overall semantic information of the document, taking into account the interactions between words and the context in which they appear.

The Distributed Memory (DM) model is one of the two main architectures of Doc2Vec. It extends the Word2Vec CBOW (Continuous Bag of Words) model by incorporating the document vector as an additional input during training. The DM model aims to capture the contextual information of the target word within the document.

In the DM model, a memory matrix, also known as the paragraph matrix, is introduced. This matrix acts as a global representation of the document and is shared across all the words in the document. The memory matrix is updated during the training process to encode the overall context of the document.

During training, the DM model takes into account both the context words in the window surrounding the target word and the document vector. It learns to predict the target word based on this combined information. By considering the document context, the DM model can capture the relationships between words within the document and generate document vectors that encode the semantic meaning of the document.

The Distributed Bag of Words (DBOW) model is the other main architecture of Doc2Vec. It is a simpler alternative to the DM model that does not consider the word order within the document. Instead, the DBOW model treats the document as a bag of words and focuses solely on the document vector to predict the target word.

In the DBOW model, the document vector is used as the input for the training process, and no additional information about the context or word order is taken into account. This architecture is computationally efficient and can be particularly useful when working with larger datasets.

The DBOW model aims to capture the overall content and meaning of the document without considering the specific interactions between words. Although it may lose some fine-grained information about word order and context, it can still generate meaningful document embeddings that can be useful in various NLP tasks.

The choice between the DM and DBOW models depends on the specific characteristics of the dataset and the requirements of the task at hand. The DM model captures more detailed contextual information but is computationally more expensive, while the DBOW model is simpler and faster but may sacrifice some word order information.

Both the DM and DBOW models are trained using a variant of stochastic gradient descent, where the model parameters are optimized based on the prediction of the surrounding words given the document (DM) or the document vector (DBOW). The training process iteratively updates the document vector and word vectors to improve the prediction accuracy.

By utilizing the paragraph vector and incorporating it into either the DM or DBOW model, Doc2Vec enables the generation of document embeddings that capture the semantic meaning and context of documents. These embeddings serve as powerful representations for documents and can be used in various NLP applications such as text classification, clustering, information retrieval, and recommendation systems.

Doc2Vec, an extension of the popular Word2Vec model, shares some similarities with its word-level counterpart but also introduces unique features tailored specifically for document-level representation. Understanding the similarities and differences between

Doc2Vec and Word2Vec is crucial for comprehending the advancements and nuances of Doc2Vec as a document representation technique.

Both Doc2Vec and Word2Vec are based on neural network architectures. They employ shallow neural networks with a hidden layer to learn distributed representations of text data. This enables them to capture the semantic relationships between words or documents.

Both models aim to learn distributed representations, or embeddings, for textual units. Instead of using discrete and sparse representations, such as one-hot encoding, Doc2Vec and Word2Vec generate continuous vector representations that capture semantic meaning and context.

Both models consider the contextual information of words. Word2Vec captures the context of a target word within a given context window, while Doc2Vec extends this concept to capture the context of words within a document. By considering context, both models aim to learn representations that reflect the semantic relationships between words or documents.

In Word2Vec, the input consists of word sequences or sentences, where the order of words is important for capturing context. In Doc2Vec, the input is a document represented by a unique document vector. The document vector is concatenated or added to the word vectors during training to capture the overall context of the document.

Word2Vec has two training objectives: the Continuous Bag of Words (CBOW) and the Skip-gram models. CBOW predicts a target word given its context, while Skip-gram predicts the context words given a target word. In Doc2Vec, the training objectives are similar, but the models are extended to include the document vector as an additional input for predicting target words. This enables the models to capture document-level semantics.

Word2Vec generates word embeddings, where each word in the vocabulary is associated with a vector. These word embeddings are often used as features in downstream NLP tasks. Doc2Vec, on the other hand, generates document embeddings, where each document is represented by a vector. These document embeddings capture the semantic meaning and context of the entire document and can be utilized for tasks such as document classification, clustering, or retrieval.

In Word2Vec, the dimensionality of the word embeddings is predefined and typically ranges from tens to a few hundred dimensions. In Doc2Vec, the dimensionality of the document embeddings is flexible and can be adjusted based on the desired application or the size of the training dataset. This flexibility allows Doc2Vec to capture the complexity and nuances of documents.

In summary, Doc2Vec is a powerful document representation technique that generates continuous vectors to capture the semantic meaning and context of documents. It addresses the limitations of traditional methods by considering the interactions between words within the document. With its ability to capture document semantics, Doc2Vec has become a valuable tool in NLP, facilitating efficient document comparison, clustering, and classification. It shares similarities with Word2Vec in terms of neural network architecture and contextual information, but introduces unique features like document vectors for capturing document-level semantics. By using document embeddings instead of word embeddings, Doc2Vec allows for the representation and analysis of entire documents, expanding the range of NLP applications.

2.3.15 Training Doc2Vec

Before training the Doc2Vec model, it is essential to preprocess the training data to ensure consistency and optimal performance. The first step in preparing the data is tokenization, which involves breaking down the documents into individual words or tokens. This process can be performed using various techniques, such as using whitespace or punctuation as

delimiters, or employing more advanced tokenization methods like word segmentation or language-specific tokenizers.

Once the documents are tokenized, several preprocessing steps can be applied to enhance the quality of the training data. These steps may include: - Converting all text to lowercase: This ensures that words with different capitalizations are treated as the same token, reducing the vocabulary size and improving generalization. - Removing punctuation: Punctuation marks often do not carry significant semantic meaning and can be safely removed. - Handling stopwords: Stopwords are commonly occurring words like "the," "and," or "is" that do not contribute much to the semantic meaning of the document. Depending on the specific task, stopwords can be removed or retained. - Lemmatization or stemming: These techniques reduce words to their base or root forms to further reduce vocabulary size and capture the essence of the words. Lemmatization tends to produce better results but is computationally more expensive than stemming.

By tokenizing and preprocessing the training data, the documents are transformed into a suitable format for training the Doc2Vec model, ensuring that unnecessary noise is removed and the important semantic information is preserved.

In order to train the Doc2Vec model, a tagged document corpus needs to be constructed. A tagged document corpus consists of individual documents, where each document is represented as a list of words (tokens) and assigned a unique tag or label. The tag can be any string identifier, such as a document ID or a numerical index.

The tagging of documents is crucial for the model to differentiate between different documents during training. The tags serve as the document labels that the Doc2Vec model learns to predict based on the context words or the document vector. It is important to ensure that each document is uniquely tagged to prevent ambiguity or misalignment during training.

The tagged document corpus is typically represented as a list of 'TaggedDocument' objects, where each object contains the document text (list of words) and the associated tag. The 'TaggedDocument' objects can be created using libraries like Gensim, which provide convenient functions for building and managing the corpus.

By constructing a tagged document corpus, the training data is organized in a structured manner, allowing the Doc2Vec model to learn the semantic relationships between words and documents effectively.

Before training the Doc2Vec model, the model parameters need to be initialized. The parameters include the dimensionality of the word and document vectors, the learning rate, and other hyperparameters that control the training process.

The dimensionality of the word and document vectors is a crucial parameter that determines the complexity and expressiveness of the embeddings. Typically, the dimensionality is set based on the size of the training dataset and the desired trade-off between model capacity and computational efficiency. Larger dimensionality allows for more nuanced representations but may require more training data and computational resources.

The learning rate determines the step size taken during the optimization process. It affects the speed of convergence and the stability of the training process. Setting an appropriate learning rate is crucial to ensure that the model parameters are updated effectively without overshooting or converging too slowly.

Other hyperparameters, such as the window size (the number of words considered as context), the number of training iterations (epochs), and the negative sampling parameter, also need to be set based on the characteristics of the training data and the desired performance.

The training of the Doc2Vec model involves optimizing the model parameters to minimize the prediction error. The training process typically employs stochastic gradient descent (SGD) or its variants to update the parameters iteratively.

In each training iteration, a document is randomly selected from the tagged document corpus. The model then predicts the target word(s) based on the context words and the document vector. The prediction error is computed using a loss function, such as softmax or negative sampling, which measures the discrepancy between the predicted and actual target word(s).

SGD updates the model parameters to minimize the loss by calculating the gradient of the loss function with respect to the parameters. The gradient represents the direction and magnitude of the steepest ascent, and SGD takes a step in the opposite direction (descending the gradient) to update the parameters.

During the SGD process, the word vectors, document vector, and other model parameters are updated based on the gradient descent update rule, which involves multiplying the gradient by the learning rate and subtracting it from the current parameter values. The learning rate determines the step size of the update and plays a crucial role in achieving convergence.

The training process continues for a specified number of iterations or until a convergence criterion is met. Convergence criteria can be based on the loss function or the stability of the model parameters. Common convergence criteria include reaching a predefined number of training iterations, achieving a minimum loss threshold, or observing no significant improvement in the loss over multiple iterations.

By training the Doc2Vec model using stochastic gradient descent, the model learns to generate document embeddings that capture the semantic meaning and context of the documents. The iterative optimization process gradually refines the model parameters to improve the prediction accuracy and enhance the quality of the embeddings.

The training process of the Doc2Vec model involves tuning various hyperparameters to optimize the performance and quality of the learned document embeddings. In this section, we will discuss the impact of several key hyperparameters and their significance in training the Doc2Vec model effectively.

The vector size, also known as the dimensionality, determines the length of the word and document vectors generated by the Doc2Vec model. The vector size is a crucial hyperparameter that affects the expressiveness and complexity of the embeddings. Larger vector sizes allow for more detailed representations but may require larger training datasets and computational resources. Smaller vector sizes, on the other hand, may lead to less nuanced embeddings but can be computationally efficient. The choice of vector size depends on the specific application and the available resources.

The window size determines the number of words taken into consideration as the context during training. It defines the local context within which the model learns to predict the target word(s). A smaller window size focuses on capturing more immediate context, while a larger window size considers a broader context. The window size should be set based on the characteristics of the training data. For documents with long-range dependencies, a larger window size may be more appropriate, whereas for documents with more local context, a smaller window size can be effective. Generally, a window size of around 5-10 words is a reasonable starting point.

The minimum word count is a hyperparameter that specifies the minimum frequency threshold for words to be included in the training process. Words that occur less frequently than the minimum word count are usually removed from the training data. This can help reduce noise and improve the quality of the embeddings by focusing on more informative and representative words. However, setting the minimum word count too high can lead to the exclusion of rare or domain-specific words that may carry important semantic information. Finding the right balance is crucial, and it often requires empirical experimentation and domain knowledge.

The learning rate controls the step size taken during parameter updates in the training

process. It determines how quickly or slowly the model converges during optimization. A higher learning rate allows for faster convergence but may lead to overshooting and instability. On the other hand, a lower learning rate ensures more stable updates but may result in slower convergence or getting stuck in suboptimal solutions. Choosing an appropriate learning rate is crucial for successful training. Techniques such as learning rate schedules, adaptive learning rates (e.g., AdaGrad, Adam), or early stopping can be employed to optimize the learning rate during training.

The number of epochs refers to the number of times the entire training dataset is passed through during training. Each epoch consists of multiple iterations where the model updates its parameters using SGD. Increasing the number of epochs allows the model to see more training examples and refine the embeddings further. However, training for too many epochs can lead to overfitting, where the model becomes too specific to the training data and performs poorly on unseen data. It is important to strike a balance between underfitting and overfitting by monitoring the performance of the model on validation or test data and selecting an appropriate number of epochs.

These hyperparameters significantly influence the performance and quality of the Doc2Vec model. It is essential to carefully tune these hyperparameters based on the characteristics of the training data and the specific task at hand. Often, a combination of empirical experimentation and domain expertise is required to achieve the optimal hyperparameter configuration.

2.3.16 Document embedding

One of the primary goals of training the Doc2Vec model is to obtain document vectors, also known as document embeddings. Document vectors capture the semantic meaning and context of the entire document, allowing for various downstream NLP tasks such as document classification, clustering, or information retrieval.

After training the Doc2Vec model, obtaining document vectors for new or unseen documents involves a two-step process.

Inferencing refers to the process of using the trained model to generate document vectors for new documents. To infer document vectors, the model requires the text of the document as input. The document text is tokenized and preprocessed following the same steps used during training, such as tokenization, lowercasing, and removing stopwords. The preprocessed document text is then fed into the trained Doc2Vec model.

Document vector extraction: Once the document text is provided as input, the Doc2Vec model generates a fixed-length vector representation for the document. This document vector encapsulates the semantic information and context of the document. The vector can be extracted directly from the model output or obtained by averaging or pooling the word vectors within the document. The specific method of extracting the document vector may vary depending on the implementation or the task at hand.

By obtaining document vectors, the Doc2Vec model enables the representation of entire documents as dense, continuous vectors, capturing their semantic meaning and context.

Document vectors generated by the Doc2Vec model encode the semantic meaning of the corresponding documents. These vectors capture the distributional properties of words and their interactions within the document, enabling the model to capture subtle nuances and semantic relationships.

The semantic meaning of document vectors can be understood through their proximity in the vector space. Similar documents are expected to have similar document vectors, indicating that they share common semantic characteristics. Conversely, dissimilar documents will have different document vectors, reflecting their distinct semantic properties.

By leveraging the semantic meaning encoded in document vectors, various NLP tasks can be performed. For example, document classification algorithms can use document

vectors as input features to classify documents into predefined categories. Document clustering algorithms can group similar documents together based on their vector representations. Document retrieval systems can use document vectors to measure similarity between query documents and target documents, enabling effective retrieval of relevant documents.

Document vectors obtained from the Doc2Vec model enable similarity measurement between documents. Similarity measurement is a fundamental operation in many NLP applications, such as recommendation systems, information retrieval, and document clustering.

To measure similarity between document vectors, various distance or similarity metrics can be used, such as cosine similarity, Euclidean distance, or Manhattan distance. Cosine similarity is a commonly used metric that calculates the cosine of the angle between two document vectors, indicating the similarity of their orientations in the vector space. A cosine similarity value close to 1 indicates high similarity, while a value close to 0 signifies dissimilarity.

By measuring the similarity between document vectors, it becomes possible to identify documents that are semantically related or similar in content. This can be utilized for tasks such as finding similar documents, recommending relevant content, or identifying duplicate or near-duplicate documents.

In conclusion, document embeddings obtained from the Doc2Vec model provide a powerful representation of documents that captures their semantic meaning and context. These embeddings enable various NLP tasks by allowing similarity measurement, document classification, clustering, and retrieval. The semantic information encoded in document vectors facilitates a deeper understanding of documents and supports the development of advanced NLP applications.

2.3.17 Advantages and Limitations of Doc2Vec

One of the key advantages of Doc2Vec is its ability to capture document-level semantics. Unlike traditional bag-of-words or TF-IDF representations, which treat documents as collections of individual words without considering their relationships, Doc2Vec considers the entire document as a unit. By generating document embeddings, Doc2Vec captures the semantic meaning and context of the entire document, enabling more nuanced and comprehensive representations. This allows for a deeper understanding of documents and facilitates tasks such as document comparison, classification, and clustering.

Another advantage of Doc2Vec is its ability to handle out-of-vocabulary (OOV) words. OOV words are words that are not present in the training vocabulary. Traditional methods, such as bag-of-words, often struggle to handle OOV words, as they rely on pre-defined vocabularies. However, Doc2Vec overcomes this limitation by learning distributed representations for words and documents. Even if a word is not seen during training, the model can still generate a meaningful representation for it based on the contextual information available in the training data. This makes Doc2Vec more robust and adaptable to diverse vocabularies and data sources.

Doc2Vec generates continuous representations for words and documents, as opposed to discrete representations used in traditional methods. Continuous representations capture the subtle semantic relationships between words and documents, allowing for more nuanced analysis and modeling. These continuous representations also facilitate mathematical operations on the vectors, such as vector addition and subtraction, which can be useful in tasks like analogy reasoning or generating document embeddings for unseen combinations of words.

The embeddings learned by the Doc2Vec model are transferable across tasks and domains. Once the model is trained on a large corpus, the learned embeddings can be

used as features for various downstream NLP tasks without retraining the entire model. This transferability is valuable in scenarios where labeled training data is limited or time-consuming to acquire. By leveraging the pre-trained Doc2Vec embeddings, it is possible to achieve good performance on new tasks with relatively small labeled datasets.

Training a Doc2Vec model can be computationally expensive, especially when dealing with large datasets and high-dimensional embeddings. The training process requires iterating over the entire document corpus multiple times, making it time-consuming. Additionally, the dimensionality of the embeddings affects the computational requirements, with higher-dimensional embeddings demanding more resources. It is essential to consider the available computational resources and time constraints when training Doc2Vec models.

The performance of the Doc2Vec model is sensitive to the selection of hyperparameters. Parameters such as vector size, window size, and learning rate can significantly impact the quality of the learned embeddings. Finding the optimal set of hyperparameters often requires empirical experimentation and tuning, which can be time-consuming and resource-intensive. Moreover, the optimal hyperparameter values may vary depending on the characteristics of the training data and the specific task at hand. It is important to carefully choose and fine-tune the hyperparameters to achieve the best performance.

While Doc2Vec generates powerful embeddings that capture semantic meaning, interpreting these embeddings can be challenging. The embeddings are high-dimensional vectors that do not have a direct interpretation in terms of human-understandable features. Understanding the underlying factors or dimensions represented by the embeddings requires additional analysis and visualization techniques. Without proper interpretation, it can be difficult to gain insights into the specific semantic aspects captured by the embeddings.

Doc2Vec treats the entire document as a single unit and generates a single document vector. While this is advantageous in capturing document-level semantics, it can be limiting in certain scenarios where finer granularity is desired. For example, if a document contains multiple distinct topics or sentiments, the single document vector may not be able to capture these nuances separately. In such cases, more advanced techniques, such as topic modeling or sentiment analysis, may be necessary to supplement the Doc2Vec embeddings.

One of the limitations of Doc2Vec is its reliance on large amounts of training data. Doc2Vec models require a substantial corpus of documents to effectively capture the semantic meaning and context. Training on small or insufficient datasets may result in suboptimal embeddings that do not adequately represent the underlying semantics. Insufficient training data can lead to overfitting, where the model memorizes the limited patterns in the data rather than learning generalized representations. To mitigate this limitation, it is recommended to train Doc2Vec models on diverse and representative datasets to ensure robust and meaningful embeddings.

Doc2Vec's performance is highly dependent on the selection and fine-tuning of its hyperparameters. The vector size, window size, minimum word count, learning rate, and number of epochs all significantly impact the quality and effectiveness of the learned embeddings. The optimal hyperparameter values may vary depending on the specific task, dataset, and domain. Determining the ideal values often requires iterative experimentation and evaluation. The sensitivity to hyperparameter tuning can make training Doc2Vec models a time-consuming and resource-intensive process. It is essential to invest effort in carefully selecting and optimizing the hyperparameters to achieve the best results.

While Doc2Vec embeddings capture semantic meaning, they are inherently complex and lack interpretability. The embeddings are high-dimensional vectors with no direct mapping to human-understandable features or concepts. This limits the ability to inter-

pret the specific dimensions or factors represented by the embeddings. Understanding the underlying semantics encoded in the embeddings requires additional analysis and visualization techniques. Techniques such as dimensionality reduction or visualization methods like t-SNE can help gain insights into the distribution and relationships within the embedding space. However, interpreting the embeddings at an individual feature level remains challenging.

The training process of Doc2Vec models involves complex neural networks and optimization algorithms. While this complexity enables the generation of powerful embeddings, it also results in a lack of transparency. Understanding the inner workings of the model and how it generates the embeddings can be challenging. This lack of transparency can make it difficult to diagnose issues or troubleshoot when encountering problems. It also hinders the ability to make informed decisions about model modifications or improvements. Additional research and experimentation are often required to gain a deeper understanding of the training process and improve the effectiveness of Doc2Vec models.

Doc2Vec relies on the notion of a fixed-length context window, where words within a certain proximity contribute to the context. However, this fixed context window may not capture long-range dependencies or global context that extends beyond the window size. In cases where documents have complex structures or dependencies that span across larger sections, Doc2Vec may struggle to capture the complete context. Addressing this limitation may require modifications to the training process or the use of alternative techniques that consider longer-range dependencies.

In conclusion, understanding the advantages and limitations of Doc2Vec is essential for its effective application in natural language processing (NLP) tasks. Doc2Vec offers advantages such as capturing document-level semantics and handling out-of-vocabulary (OOV) words. It allows for comprehensive document representation and analysis. However, there are also limitations to consider. Doc2Vec requires large amounts of training data to generate meaningful embeddings, and it is sensitive to hyperparameter tuning, requiring careful optimization. The interpretability of Doc2Vec embeddings can be challenging, and the training process lacks transparency. Additionally, Doc2Vec has contextual limitations regarding capturing long-range dependencies. By acknowledging these limitations, practitioners can make informed decisions, explore alternative methods, and maximize the benefits of Doc2Vec in their NLP applications.

2.3.18 Top2Vec

In this chapter, we will delve into the concept of unsupervised topic modeling and document embedding, discussing the need for such techniques and introducing Top2Vec as a powerful method to address these tasks.

In the realm of natural language processing (NLP) and text analysis, it is essential to extract meaningful insights from large volumes of unstructured text data. Traditional approaches to topic modeling, such as Latent Dirichlet Allocation (LDA), require prelabeled training data and assume that documents belong to a fixed number of predefined topics. However, in real-world scenarios, the number of topics may be unknown or constantly changing, making it challenging to accurately label and train models on new data. Unsupervised methods aim to overcome these limitations by automatically discovering latent topics and generating document representations without the need for labeled training data.

Top2Vec is an advanced unsupervised technique that was pressented by Angelov in 2020. It combines topic modeling and document embedding into a single framework. It leverages the power of word embeddings, such as Word2Vec or GloVe, to learn distributed representations of words that capture semantic relationships. By representing documents as a collection of word vectors, Top2Vec enables the exploration of topics at various levels

of granularity, from broad themes to specific subtopics. Unlike traditional topic modeling methods that assign documents to fixed topics, Top2Vec allows documents to belong to multiple topics simultaneously, capturing the complex nature of real-world text data. The technique also provides a hierarchical structure, allowing users to navigate topics at different levels and explore the relationships between them. Top2Vec's ability to automatically discover topics and generate document embeddings without the need for labeled data makes it highly scalable and applicable to a wide range of domains and datasets. Furthermore, the technique is capable of handling streaming data, adapting to changing topics over time, and supporting real-time topic inference.

In the upcoming sections, we will dive deeper into the inner workings of Top2Vec, exploring its methodology, algorithms, and practical applications. We will also discuss how to preprocess text data, tune model parameters, and interpret the results generated by Top2Vec. By the end of this book, you will have a comprehensive understanding of Top2Vec and its potential to revolutionize unsupervised topic modeling and document embedding in the field of NLP.

2.3.19 What is Top2Vec

In this section, we will look at the mechanics of Top2Vec—an innovative hybrid approach that combines the strengths of word2vec, doc2vec, and Universal Sentence Encoder (USE). We will define Top2Vec, explore its unique capabilities in generating topic vectors that encapsulate the semantic meaning of both individual words and entire documents, and highlight the numerous advantages it offers, including scalability, interpretability, and the ability to handle large document collections.

Top2Vec can be best described as a hybrid method that draws inspiration from word2vec, doc2vec, and Universal Sentence Encoder (USE). Word2vec is a popular algorithm that learns distributed representations of words, capturing their semantic relationships and contextual information. Doc2vec, an extension of word2vec, extends this concept to learn embeddings for entire documents, allowing for document-level analysis. Universal Sentence Encoder (USE) takes the concept further by providing pre-trained models capable of generating sentence-level embeddings, encapsulating the meaning of entire sentences. Top2Vec combines the power of these techniques, harnessing the strengths of word2vec, doc2vec, and USE to generate topic vectors that capture the semantic essence of both individual words and entire documents.

Top2Vec employs a two-step process to generate topic vectors that encapsulate the semantic meaning of words and documents.

In the first step, word vectors are generated using word2vec or similar embedding algorithms. These word vectors capture the contextual relationships and semantic similarities between words.

The second step involves aggregating the word vectors to create document-level embeddings. This is achieved through a novel technique that combines the word vectors and applies dimensionality reduction, resulting in dense representations that capture the overall theme of each document. By combining the word-level and document-level embeddings, Top2Vec generates topic vectors that encapsulate the semantic meaning of both individual words and entire documents, facilitating a comprehensive understanding of the underlying content.

Top2Vec offers remarkable scalability, making it suitable for handling large document collections. By eliminating the need for manual labeling or predefined topic structures, it significantly reduces the burden of data preparation, allowing for efficient analysis of massive text datasets.

The topic vectors generated by Top2Vec provide interpretable results, enabling users to understand and make sense of the discovered topics. The hierarchical structure of topics

facilitates navigation and exploration at different levels, enhancing interpretability and aiding in extracting actionable insights from the data.

Top2Vec's architecture and algorithms are designed to handle large-scale text collections effectively. It can process vast amounts of data efficiently, ensuring that the analysis remains feasible and computationally tractable.

Top2Vec is capable of adapting to changing topics over time, making it ideal for streaming data or scenarios where topics evolve dynamically. This feature allows for real-time topic inference and ensures that the model remains relevant and up to date.

Top2Vec's unsupervised nature and ability to capture both word-level and document-level semantics make it applicable to a wide range of domains and datasets. It can be effectively employed in fields such as social media analysis, market research, customer feedback analysis, and scientific literature exploration.

In the upcoming chapters, we will delve further into the intricacies of Top2Vec, exploring its implementation, optimization techniques, and advanced applications. By the end of this book, you will possess a comprehensive understanding of Top2Vec and its potential to revolutionize topic modeling and

2.3.20 Architecture of Top2Vec

In this chapter, we will explore the architecture of Top2Vec—a powerful framework that combines key components such as Word2Vec, Doc2Vec, and Universal Sentence Encoder (USE). We will observe the role of each component in generating word and document embeddings and discuss how Top2Vec seamlessly integrates these techniques to create a unified model. Additionally, we will examine the importance of hierarchical softmax in training Top2Vec for efficient computation. Top2Vec Implementation/Architecture

At its core, Top2Vec leverages the principles of Word2Vec, which is a popular algorithm for learning distributed representations of words. Word2Vec employs either the Continuous Bag of Words (CBOW) model or the Skip-gram model. The CBOW model predicts the target word based on its surrounding context words, while the Skip-gram model predicts the context words given a target word. By training on large text corpora, Word2Vec captures semantic relationships and contextual information, generating high-quality word embeddings.

Building upon Word2Vec, Top2Vec incorporates the concept of Doc2Vec, an extension that extends word embeddings to encompass entire documents. Doc2Vec treats documents as continuous bags-of-words and learns paragraph vectors, also known as document embeddings, which represent the semantic meaning of the entire document. By incorporating document-level context, Doc2Vec enables document-level analysis and comparison, adding an extra layer of semantic depth to the model.

Furthermore, Top2Vec integrates the Universal Sentence Encoder (USE), which is a pre-trained model that encodes sentences into fixed-length vectors. The USE model captures the semantic meaning of entire sentences by leveraging a deep neural network architecture and training on a large corpus. By incorporating USE, Top2Vec enhances its ability to capture the semantics of short text snippets or small documents effectively.

The architecture of Top2Vec seamlessly combines the word embeddings generated by Word2Vec, the document embeddings generated by Doc2Vec, and the sentence embeddings generated by USE. This integration allows Top2Vec to create topic vectors that encapsulate the semantic meaning of both individual words and entire documents, providing a comprehensive understanding of the underlying content.

To optimize the training process and enable efficient computation, Top2Vec employs a technique called hierarchical softmax. This technique reduces the computational complexity associated with training models with large vocabularies. Hierarchical softmax organizes the vocabulary into a binary tree structure, where internal nodes represent decision points

between child nodes, and leaf nodes represent individual words or documents. During training, hierarchical softmax traverses the binary tree to compute the probabilities of words or documents more efficiently, significantly reducing computational overhead.

The implementation of Top2Vec involves training the model on a large corpus of text data, utilizing the combined power of Word2Vec, Doc2Vec, and USE to generate word, document, and sentence embeddings. The training process involves iteratively optimizing the model parameters, such as the dimensionality of the embeddings, the learning rate, and the window size, to capture the most meaningful semantic relationships within the data.

Once trained, Top2Vec allows for efficient exploration of topics within the corpus. It offers hierarchical topic structures that enable users to navigate topics at different levels of granularity, from broad themes to specific subtopics. This hierarchical structure aids in interpretability and facilitates the extraction of actionable insights from the text data.

2.3.21 Advantages and limitations of Top2Vec

In comparison to other topic modeling and document embedding techniques, Top2Vec offers a unique set of strengths and weaknesses. Understanding these aspects is crucial for determining when and where Top2Vec excels and where alternative methods might be more appropriate.

One of the key strengths of Top2Vec is its ability to capture both word-level and document-level semantics. By combining word embeddings from Word2Vec, document embeddings from Doc2Vec, and sentence embeddings from USE, Top2Vec provides a holistic view of the underlying content. This comprehensive semantic understanding enhances the accuracy and interpretability of the generated topic vectors.

Top2Vec is highly scalable, making it well-suited for handling large document collections. Unlike some traditional topic modeling techniques that may struggle with scalability, Top2Vec's architecture and algorithms are designed to efficiently process vast amounts of data. It eliminates the need for manual labeling or predefined topic structures, significantly reducing the data preparation burden and enabling efficient analysis of massive text datasets.

Top2Vec generates topic vectors that offer interpretable results, making it easier for users to understand and make sense of the discovered topics. The hierarchical structure of the topics allows for navigation and exploration at different levels of granularity, facilitating the interpretation of broad themes and specific subtopics. This interpretability is particularly valuable in applications where actionable insights need to be extracted from the data.

Top2Vec's architecture is adaptable to changing topics over time. This feature makes it ideal for scenarios with dynamic or streaming data, where topics evolve continuously. By allowing for real-time topic inference, Top2Vec ensures that the model remains relevant and up to date, enabling accurate analysis of evolving text data.

Like many other machine learning techniques, Top2Vec requires a sufficient amount of training data to generate meaningful and accurate embeddings. Insufficient data can lead to suboptimal results, as the model may struggle to capture the semantic nuances present in the text corpus. Adequate data collection and preprocessing are essential to leverage the full potential of Top2Vec.

While Top2Vec offers impressive capabilities, there is a risk of overfitting when training the model on smaller or domain-specific datasets. Overfitting occurs when the model becomes too specialized to the training data and fails to generalize well to new, unseen data. Regularization techniques and careful selection of hyperparameters can mitigate this risk and enhance the model's generalizability.

Due to its sophisticated architecture, Top2Vec may require substantial computational resources during training and inference, especially when dealing with large vocabularies and extensive document collections. Adequate computing power and memory capacity are necessary to ensure efficient processing and timely results. However, advancements in hardware technology and distributed computing frameworks can help mitigate these computational challenges.

While Top2Vec exhibits versatility across domains, it may perform better in certain domains than others. The performance of Top2Vec is influenced by the characteristics of the data it is trained on. Thus, it is important to consider the specific domain and nature of the text corpus when evaluating the suitability of Top2Vec for a particular task. Fine-tuning and customization techniques may be necessary to optimize the model's performance for specific domains.

It is worth noting that the strengths and weaknesses of Top2Vec should be considered in the context of the specific use case and requirements. While it offers numerous advantages in terms of semantic understanding, scalability, interpretability, and adaptability, it is essential to assess its limitations and explore alternative techniques when they align better with the specific task or dataset at hand.

Chapter 3

State of the art in trend detection

3.1 Automatic trend detection: Time-biased document clustering

The first paper that will be discussed is called Automatic trend detection: Time-biased document clustering presented by Behpour et al. (2021). The approach proposed in this article combines traditional document clustering techniques with a time-bias factor that captures the temporal order of documents.

The authors first discuss the challenges of trend detection in large text corpora and the limitations of existing methods, such as keyword extraction and topic modeling. They argue that these methods fail to capture the temporal nature of trends and do not account for changes in the frequency and distribution of keywords over time.

To overcome these limitations, the authors propose a new method that combines document clustering with a time-bias factor. The time-bias factor is defined as the time difference between each document and a reference point, such as the publication date of a key document or the start of a time period of interest. By including the time-bias factor in the clustering process, the authors aim to ensure that documents that are more closely related in time are clustered together.

The authors then describe the implementation of their approach using two real-world datasets: a collection of news articles related to the 2008 financial crisis and a collection of Twitter messages related to the 2014 Ebola outbreak. For both datasets, the authors apply their time-biased clustering approach and evaluate the resulting clusters using various metrics, such as purity, entropy, and normalized mutual information.

The experimental results show that the time-biased clustering approach outperforms traditional document clustering techniques in terms of trend detection. The authors observe that the time-biased clusters capture the temporal evolution of the trends and provide a more accurate representation of the underlying topics. They also note that the time-biased clustering approach is more robust to noise and outliers than traditional clustering methods.

The authors conclude that their time-biased clustering approach is a promising method for trend detection in large text corpora. They argue that the approach can be applied to a wide range of applications, such as tracking the evolution of public opinion on social media or detecting emerging trends in scientific literature. The authors also suggest several directions for future research, such as incorporating domain-specific knowledge and evaluating the approach on different types of text corpora.

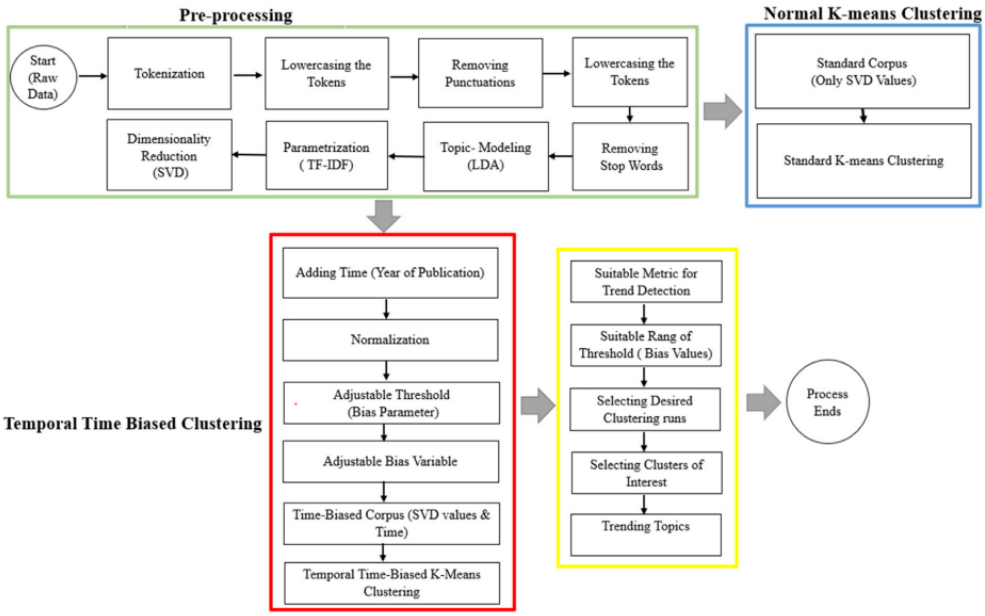


Figure 3.1: Diagram of the trend detection framework. Pre-processing is the first step, which produces vectors for every document. In Temporal (Time-Biased) Clustering, they add the year of publication to the resulting corpus, which introduces an adjustable bias, and perform clustering on this biased representation.

3.2 TwitterMonitor: Trend Detection over the Twitter Stream

The article written by Mathioudakis and Koudas (2010) presents TwitterMonitor, a system that performs trend detection over the Twitter stream. The goal is to identify emerging topics (i.e. trends) on Twitter in real-time and provide meaningful analytics that synthesize an accurate description of each topic. The authors explain the motivation for trend detection over social media streams and the challenges that come with it. The authors describe their approach to trend detection, as well as the architecture of TwitterMonitor. Finally, they lay out their demonstration scenario.

Twitter is currently the major microblogging service with over 11 million active subscribers generating over 6 million tweets per day. The Twitter stream is a document stream that contains a wealth of information and offers significant opportunities for exploration as well as challenges. One of the first challenges is to automatically detect and analyze the emerging topics that appear in the stream and to do so in real-time. Trends are typically driven by emerging events, breaking news, and general topics that attract the attention of a large fraction of Twitter users. Trend detection is of high value to news reporters and analysts, as they might point to fast-evolving news stories. Trend detection is also important for online marketing professionals and opinion tracking companies, as trends point to topics that capture the public's attention.

The authors present TwitterMonitor, a system that performs trend detection in two steps and analyzes trends in a third step. First, it identifies 'bursty' keywords, i.e. keywords that suddenly appear in tweets at an unusually high rate. Subsequently, it groups bursty keywords into trends based on their co-occurrences. In other words, a trend is identified as a set of bursty keywords that occur frequently together in tweets. After a trend is identified, TwitterMonitor extracts additional information from the tweets that belong to the trend, aiming to discover interesting aspects of it.

The architecture of TwitterMonitor is depicted in fig:twitter$_architect.Thesystemcomprisesthreemaincom$

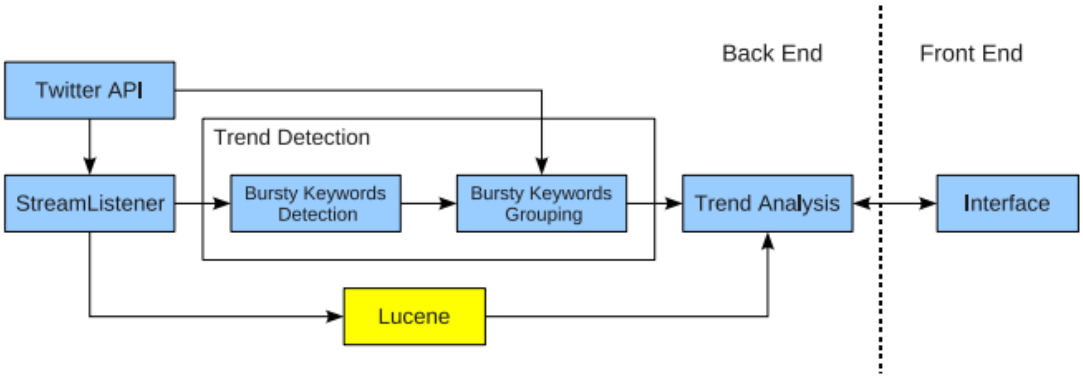


Figure 3.2: TwitterMonitor architecture.

$(1) the Trend Detection component, (2) the Trend Grouping component, and (3) the Trend Analysis component.$
$occurrences. The Trend Analysis component extracts additional information from the tweets that belong to the$

The authors explain in detail each of the three components of TwitterMonitor. The Trend Detection component employs a simple algorithm that looks for sudden increases in the frequency of keywords in the Twitter stream. The Trend Grouping component uses a graph-based approach to group the bursty keywords into trends. The Trend Analysis component identifies the most relevant tweets in a trend and extracts additional information from them, such as sentiment analysis, author demographics, and temporal trends.

Finally, the authors present a demonstration scenario of TwitterMonitor in action. They show how the system can be used to detect trends and analyze them in real-time. They also demonstrate how users can interact with the system by ordering the identified trends using different criteria and submitting their own descriptions for each trend.

3.3 Collective dynamics in knowledge networks: Emerging trends analysis

Various approaches to studying research fronts and emerging trends in the field of bibliometrics are discussed in the article by Liu, Jiang, and Ma (2013). The concept of research fronts, as defined by Price and Persson, is explored, with subtle differences noted between the two definitions. The article discusses various clustering methods used to analyze the research front of a field, including co-citation networks and keyword collections. The article also highlights the importance of considering the effects of relevant subjects, fields, and clusters when studying research fronts. The article concludes by discussing text mining as a method for identifying emerging trends and the difficulties associated with this approach due to the polysemous nature of human language. Overall, the article provides a comprehensive overview of the various methods used to study research fronts and emerging trends in bibliometrics.

The article discusses a study that aims to identify the key factors that drive the evolution of knowledge in complex networks. The authors focus on the concept of "knowledge clusters," which they define as a set of more closely related knowledge nodes than those outside the cluster. Knowledge clusters are dynamic entities that are constantly evolving, and understanding how they change over time is essential for understanding the evolution of knowledge in complex networks. The study proposes a four-step research design to investigate the evolution of knowledge clusters. The first step involves identifying the subjects of knowledge evolution, which are the knowledge clusters themselves. The second step involves acquiring knowledge clusters by clustering all the knowledge nodes and

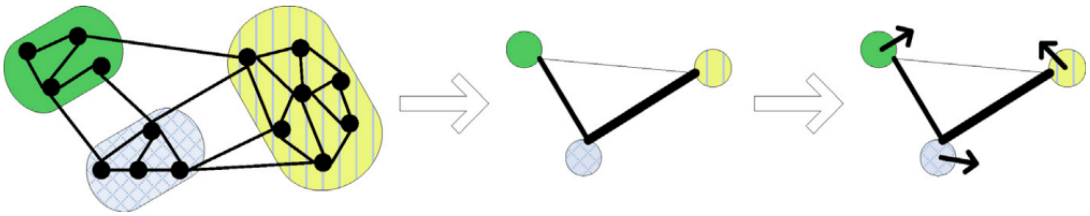


Figure 3.3: Knowledge clusters.

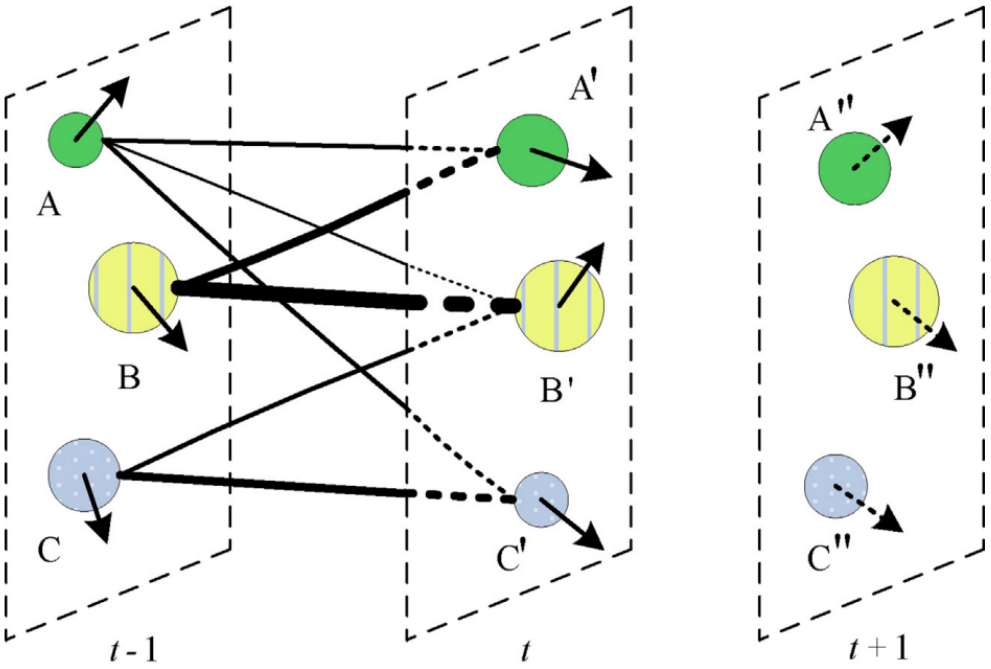


Figure 3.4: Time-slice map of knowledge clusters.

describing each block with a single node called the cluster node. The third step involves determining the direction of a knowledge cluster at a specific time by summing up the keyword vectors of all the articles in the cluster. Finally, the fourth step involves investigating the changing rules of cluster states by constructing a collective dynamics model of emerging trends in knowledge networks based on two hypotheses.

The collective dynamics model proposed in the study is based on two main hypotheses. The first hypothesis (H1) states that without external influence, the development trend of a knowledge cluster over the next time window remains the same as that over the current time window. The second hypothesis (H2) states that the development direction of a knowledge cluster is influenced by its closely related clusters in proportion to the strength of the ties between the two clusters multiplied by the difference between their keyword vectors.

The study also discusses the empirical data used to test the proposed model. The data is based on citation networks, and the authors construct knowledge clusters by clustering all the knowledge nodes in the network. They then use the keyword vectors of the knowledge nodes within the clusters as knowledge development states to investigate the changing rules of cluster states.

The study's results show that the proposed collective dynamics model can effectively capture the formative process of innovative trends in knowledge networks. The model is also able to predict the direction of knowledge clusters over time, which can be useful for

identifying emerging trends and predicting future developments in knowledge networks.

Overall, the study provides a valuable contribution to the field of knowledge evolution studies by proposing a new research design and collective dynamics model for investigating the evolution of knowledge clusters in complex networks. The study's technical details and empirical data analysis provide a comprehensive and detailed understanding of the proposed model, making it a useful resource for researchers and practitioners in the field.

The article discusses the collective dynamics method and its superiority over the baseline method in predicting the development trends of knowledge clusters. The experiments showed that the collective dynamics method achieved an average degree of 8% higher in the accuracy value and 12% higher in the coverage value than the baseline method. The collective dynamics method is also superior to the baseline method in predicting the development trends of knowledge fields with more uncertainties, with the accuracy value obtained 59% higher than the baseline method, while the coverage value is 84% higher. The article also discussed the effect of the scale of keywords on the prediction results, and the collective dynamics method achieved optimal results when the number of selected keywords is relatively small. The method has better prediction results for small knowledge clusters than for large ones, and the collective dynamics method is still obviously superior to the baseline method in predicting the development trends of small knowledge clusters.

Furthermore, the article highlights that the collective dynamics approach emphasizes the mutual influences between correlated fields, which is suitable for analyzing the DCN. The paper concludes that analyzing the collective dynamical behavior in the DCN composed of knowledge clusters is essential for exploring how to research trends form in the interaction and coordination between knowledge clusters. The study reveals the predicted value of the keyword vector of the entire cluster during the next stage, rather than the specific development subject specified by a general structural analysis.

3.4 Detecting Emerging Trends from Scientific Corpora

The article [8] discusses the emerging trend detection (ETD) problem in text mining, specifically in the context of scientific articles. ETD is defined as detecting topic areas that are growing in interest and utility over time. The article presents a model for detecting emerging trends that is richer in topic representation and more appropriate for evaluating emerging trends than existing models. The model is designed to adapt to different kinds of scientific corpora and can be modified to meet user needs.

The ETD process is viewed in three phases: topic representation, identification, and verification. In the topic representation phase, each topic is represented by a set of temporal features. These features are extracted from document databases using text-processing methods in the feature extraction phase. In the topic verification phase, the features associated with each topic are monitored over time, and the topic is classified using interest and utility functions. The effectiveness of an ETD model depends on how well a topic is represented in computers, how well the features associated with a topic are extracted from documents, and how appropriately the interest and utility functions are constructed.

While many ETD models have been proposed, most are poor in representing research topics and inappropriate for determining and ranking interest and utility. The article addresses these limitations by proposing a rich representation scheme for topics using specific features of research articles, methods for extracting these features from documents, and interest and utility measures for evaluating emerging trends. The model's evaluation is given in the experimental evaluations section, which shows that the model promises to achieve significant results in emerging trend detection.

In summary, the article presents a model for detecting emerging trends in scientific articles that is richer in topic representation and more appropriate for evaluating emerging

trends than existing models. The model can adapt to different kinds of scientific corpora and can be modified to meet user needs. The article proposes a rich representation scheme for topics using specific features of research articles, methods for extracting these features from documents, and interest and utility measures for evaluating emerging trends. The experimental evaluations show that the model promises to achieve significant results in emerging trend detection.

In this section, the authors describe the topic representation and identification process used in their proposed emerging trend detection model. The model takes a set of scientific articles D as input and associates each topic with some temporal features extracted from D. The goal is to evaluate the growth in interest and utility using two functions f and g, and to determine whether each topic is an emerging trend. The output is a set of emerging trends selected by the model.

To represent each topic, the authors use a time series that consists of six parameters, each associated with a specific year in the trial period. The first parameter measures how often the topic is mentioned in the year, while the second parameter is the weight of citations to the topic in that year, where the topic is cited for referring to a theoretical basis, using methods, or making comparisons. The third parameter is the number of citations to the topic in that year, while the fourth parameter is the influence of the topic on other topics in the year. The fifth parameter is the weight of author reputations of the topic in the year, and the sixth parameter is the weight of sources (journals/proceedings) talking about the topic in the year.

The authors note that their model differs from existing ETD models, which often use n-grams, term frequencies, and term co-occurrences associated with date tags, author names, citations, etc. Instead, their model considers each topic in relation to others, examining its change in interest and utility over time. To achieve this, each topic is organized in a concept hierarchy and associated with many features extracted from scientific articles. This gives the model a richer representation scheme for topics and allows for a more reasonable computation of growth in interest and utility.

To identify how often a topic is mentioned in a specific year, the authors measure the relevance of the topic to each article published in that year. This process, called topic identification, is an important part of automatic text processing techniques such as information retrieval, text categorization, and text summarization. The authors note that there are three main groups of methods for topic identification: statistical methods, knowledge-based methods, and hybrid methods. Statistical methods infer the topic in the text from term frequencies, term locations, and term co-occurrences, while knowledge-based methods use external knowledge sources such as dictionaries and thesauri. Hybrid methods combine both statistical and knowledge-based approaches. The authors do not specify which method they use for topic identification in their model.

In this article, the authors present their model for detecting emerging trends in scientific databases. They developed a prototype system that uses a database of 9000 full-length articles in PDF format from their university library. WordNet was used to detect synonymy and hyponymy relationships between words. Two experiments were designed to evaluate the effectiveness of the topic identification method and the accuracy of the citation type detection method. Evaluations of interest/utility functions and the entire ETD process are ongoing.

In the first experiment, the authors performed tf-idf ranking and selected 1000 topics into the concept hierarchy T. They then identified topics from the full text and from the abstract of each article to compute hits, mistakes, and misses. Two measures from Information Retrieval, Recall and Precision, were used to evaluate the algorithm's performance. The authors achieved Recall and Precision values of 0.52 and 0.58, respectively, for 100 randomly selected papers for testing. When they added the keywords provided by the

authors to the set of keywords extracted using tf-idf, the accuracy improved significantly to 0.82 in Recall and 0.87 in Precision.

The second experiment evaluated whether their method achieved higher accuracy compared to Nanba and Okumura's method when running under the same conditions. They used the same definition of citation types as theirs and achieved higher accuracy than Nanba's method using HMMs and MEMMs based on concept representation. The authors noted that Nanba's method still had problems with synonymy and hyponymy, which is why their method using concept representation resulted in higher accuracy.

In conclusion, the authors constructed a model for emerging trend detection in scientific databases with computational methods for all model components, most of which are implemented in the prototype system. Their model has a richer representation scheme for topics, and their methods for topic identification and citation type detection achieved impressive results compared to other works. The authors' contribution is significant, as they evaluated the growth in interest and utility separately, which can classify emerging trends by different criteria and clarify the development of research topics in the published literature.

3.5 A graphical decomposition and similarity measurement approach for topic detection from online news

This paper [25] presents a new approach to detect topics from online news articles. The authors explain that with the explosion of information on the internet, it has become increasingly difficult to manually categorize and classify news articles. Therefore, there is a need for automated methods to detect topics from these articles.

Traditional methods, such as LDA and clustering, require prior knowledge of the number of topics in the data. This is a problem because the number of topics is often unknown and can vary depending on the data. Furthermore, these methods do not take into account the relationships between topics, which can lead to a fragmented view of the data.

Second, current network-based methods, such as co-occurrence networks and topic correlation networks, suffer from limitations as well. Co-occurrence networks are often sparse and noisy, and may not capture the semantic relationships between words. Topic correlation networks rely on the assumption that topics are independent, which may not hold true in some cases.

Recent approaches such as non-negative matrix factorization and probabilistic topic modeling, have shown promising results. However, these methods are often computationally expensive and may not scale well to large datasets.

Authors of this article present a novel approach for topic detection from online news articles using a graphical decomposition and similarity measurement technique. To address the challenges mentioned earlier, the authors propose a two-stage approach to topic detection.

In the first stage, they construct a co-occurrence network of keywords from the news articles, which supposedly reveals the relationship between keywords. Each part of the graph has a different density. Now the step of dividing a graph into several subgraphs by community detection is done. Words inside each subgraph are highly related, while the relationship between subgraphs is relatively weak. Each subgraph is called a capsule vertex. Connecting those subgraphs requires the usage of the equation below

$$sim(C_i, C_j) = \frac{V_{C_i} * V_{C_j}}{||V_{C_i}|| * ||V_{C_j}||}, \tag{3.1}$$

where V_{C_i} and V_{C_j} represent the vector of C_i and C_j, $|| * ||$ represents the L2 norm.

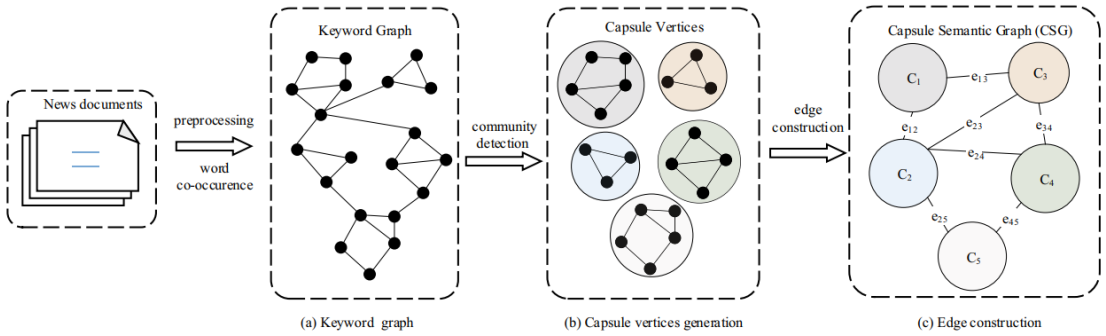


Figure 3.5: The graphical representation of the approach for topic detection proposed in paper [25]. Where (a) is a keyword graph construction stage, (b) is a community detection step, (c) final step of creating a graph from communities using similarity measures.

The CSG is an undirected weighted graph, where the weights of the edges represent the similarity between capsule vertices.

In the second stage, the article presents a method for generating topics from documents using incremental clustering. The similarity between pairs of news documents is measured based on the definition of a graph kernel between pairs of Capsule Semantic Graphs (CSGs).The graph kernel takes into account both the capsule vertices and the relationship between them, and it is defined as the sum of the kernel function for comparing two vertices and the kernel function for comparing two walks. The normalization factor is introduced to account for the length of the compared documents. The similarity value between two documents lies in the interval [0, 1].

$$k(d_1, d_2) = \frac{\sum_{v_1 \in V_1, v_2 \in V_2} k_{node}(v_1, v_2) + \sum_{e_1 \in E_1, e_2 \in E_2} k_{walk}(e_1, e_2)}{norm}, \tag{3.2}$$

where

$$k_{node}(v_1, v_2) = max(V_{v_1}^T * V_{v_2}, 0), \tag{3.3}$$

where V_{v_1} and V_{v_2} are the vectors of capsule vertices v_1 and v_2

$$norm = ||A_1 + D_1||_F * ||A_2 + D_2||_F \tag{3.4}$$

, where $||||_F$ is Fobenious norm, A_1 and A_2 are adjacency matrices, D_1 and D_2 are diagonal matrices. To generate topics, the article applies single-pass clustering to the news documents dataset. The algorithm sequentially processes the input documents and generates clusters incrementally. A news document is added to the most similar cluster if the similarity between this document and the centroid of the cluster exceeds a pre-defined threshold. Otherwise, the document is regarded as the seed of a new cluster. The algorithm returns the topic clusters, and each cluster represents a topic. The initial similarity threshold is set based on the experience, and the optimal threshold is determined through several preliminary experiments. The threshold setting is different for different datasets.

The performance of CSG-SM is evaluated using three standard datasets CEC, 20newsgroup, and THUCNews - and is compared with six baseline methods including term-based methods, probabilistic methods, and graph analytical methods. The results show that CSG-SM outperforms the baseline methods in terms of precision, recall, and F1 score for all three datasets. The normalized detection cost function is also used to evaluate the performance of the system, and CSG-SM outperforms the baseline methods on this metric as well.

The baseline methods used for comparison include GAC, KNN, LDA-VEM, LDA-GS, KG, and KG+. GAC and KNN are term-based methods, while LDA-VEM and LDA-GS are probabilistic methods. KG and KG+ are graph analytical methods. The results show that CSG-SM outperforms all of these methods in terms of precision, recall, and F1 score.

The paper acknowledges that several challenges still exist in topic detection, such as tracking the evolution of topics over time and detecting fine-grained topics. Additionally, the paper mentions the challenge of combining news with multiple data forms, such as social media data and multimedia data, for topic detection. The authors suggest investigating these problems in future research.

Overall, the paper presents a promising approach to topic detection in news articles, which overcomes the limitations of traditional approaches and achieves better performance.

3.6 Detecting research topic trends by author-defined keyword frequency

The paper [11] presents a novel approach to detecting research topic trends. The proposed method is called Author-Defined Keyword Frequency Prediction (AKFP), and it aims to predict the frequency of keywords used in academic publications over time. The AKFP task takes the features of consecutive m years of the author-defined keywords (AKs),$X(x_1, x_2, ..., x_m)$, as input variables and the AK frequency in the following n-th year, $Y(y_n)$, as output variables to find the complex functional relationship $Y = f(X)$.

To achieve this, the authors propose using a neural network model, specifically a Long Short-Term Memory (LSTM) neural network described in Section 2.2.1. LSTMs are effective in fitting the sequence property of time series data, and they avoid the problem of gradient vanishing and gradient explosion, making them suitable for this task.

The neural network framework employed in this study consists of an input layer, a hidden layer, and an output layer. The input layer consists of a single-layer feedforward neural network (FNN) that transforms the initial low-dimensional feature representations into the high-dimensional feature space using an activation function eq:sigma that converts the input into a nonlinear output.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.5}$$

The multi-layer LSTM is utilized to deal with time series data, where i, f, and o are the input, forget, and output gates, respectively, and h_j, c_j represent the short-term and long-term memory of the LSTM in time j. Finally, the output of the hidden layer is taken as the input of the output layer, which still employs a single-layer FNN to transform the output into the dimension needed.

The loss function used is the mean square error (MSE), and the model is trained using historical data of AKs. The goal of the AKFP is to quantitatively predict the frequency of use in any life-cycle stage of the AK and reveal the developing pattern of topics in a specific field.

The study aims to test the feasibility of short-, medium-, and long-term prediction of keyword frequency. The training set was constructed with the sliding window method, and the data set was divided into a training set, verification set, and test set in an 8:1:1 ratio.

Four machine learning approaches were used to evaluate the AKFPs' performance, including LSTM, XGBoost, Random Forest, KNN, and Logistic Regression. The four machine-learning approaches were implemented using the algorithm library encapsulated in scikit-learn, and the optimal parameter values were determined using the random search

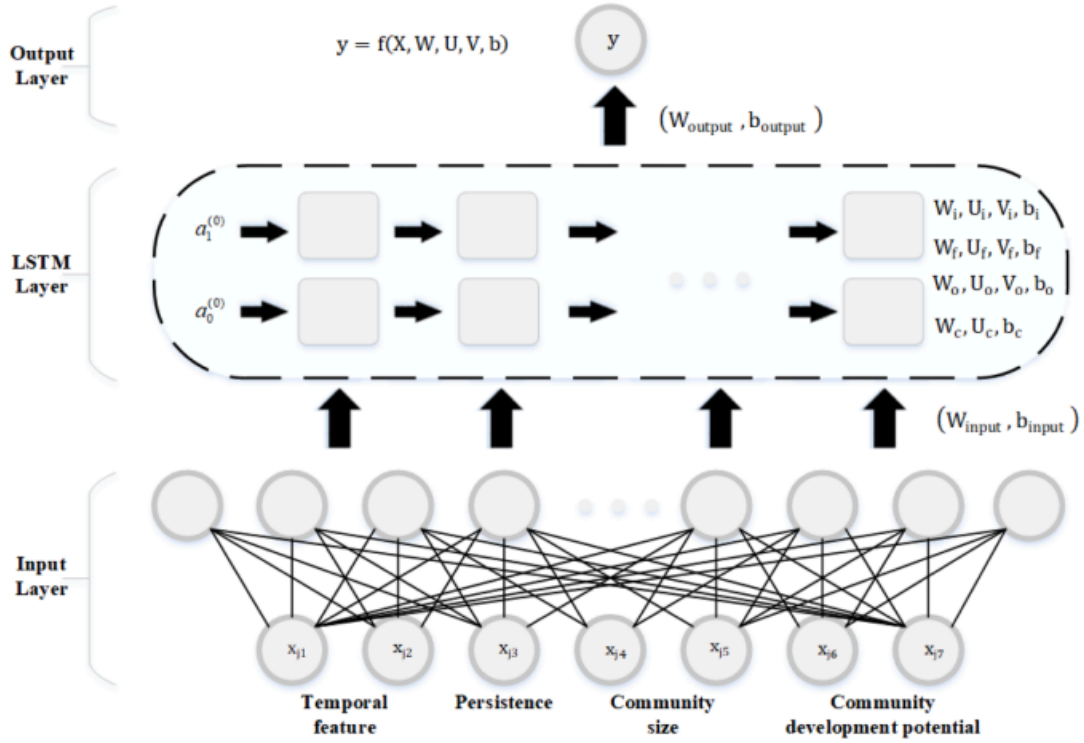


Figure 3.6: The keyword frequency prediction model defined in [11]

method. The proposed method was evaluated using the mean square error (MSE), mean average error (MAE), and coefficient of determination (R2) criteria.

The study selects 12 keywords as a case study and shows that the predicted curves keep track similar to the actual black curve, although occasionally there is a lag phenomenon. For keywords with lower frequency, the prediction curves are more oscillatory due to the randomness of actual word frequency.

The study uses three different time spans, and the results show that the LSTM model outperforms all baseline models on the test set, although it has slightly less effect on the training set than XGB and RF.

The study also shows that the longer the time span, the more challenging the task of predicting keyword frequency becomes, requiring a deeper neural network. The MSE of the optimal model also increases with the expansion of the time span. The study uses the sliding window method to predict keyword frequency, and the results show that the accuracy of the corresponding AKFP decreases as the time span increases.

The theoretical implications of the study include the proposal of AKFP as a prediction task that pursues the tradeoff between timeliness and accuracy, exploration of the factors affecting the future word frequency, and the demonstration of the better generalization ability of neural network algorithms than some common machine learning algorithms. The study also provided practical guidelines on how researchers may employ the AKFP to fit the developing pattern of topics in a field, and the limitations of the study, such as the use of keywords to represent research topics, the "black box" of the neural network, and the need for further improvement of AKFP. Overall, the AKFP is a simple and effective approach to track research trends and can be used as a tool to assist in emerging topic detection.

Chapter 4

Proposed Method

4.1 Introduction

The research trend detection task is an important problem in the field of data analysis and machine learning. The objective of this task is to identify emerging trends and patterns in large datasets of research publications, which can help researchers stay up-to-date with the latest developments in their field and inform future research directions. Despite the importance of research trend detection, there are still challenges associated with this task. One major challenge is the sheer volume and complexity of research publications, which can make it difficult to identify meaningful trends and patterns. Another challenge is the lack of a standardized methodology for trend detection, which can lead to inconsistent and unreliable results across different studies.

In order to address these challenges, we developed our own method for research trend detection. Our method is designed to be flexible and accurate, and it leverages a combination of machine learning and statistical techniques to identify and track research trends over time. The motivation for developing our own method stems from a desire to improve the accuracy of research trend detection and to provide a standardized framework for researchers to use in their own studies. By developing a method that is both reliable and easy to use, we hope to contribute to the broader goal of advancing scientific knowledge and discovery.

4.2 Method description

In this section, we present the method we developed to address the research trend detection task, including details on the data preprocessing steps, algorithmic details, and how our approach addresses the challenges associated with this task.

After reviewing the state-of-the-art methods, we decided to try top2vec as our primary tool. Since it detects topics without the need to specify the number of topics, as well as, it works well with large datasets. It uses one of the two well-established embedding algorithms (doc2vec, universal-sentence-encoder).

To validate the topic detection algorithm, we decided to test it on some datasets containing only the titles of the articles. For that, we used "dblp" website, which provides a ".xml" dump of all articles uploaded to it. We decided to train 1 model on the whole dataset to check if everything works correctly. Before training, minor preprocessing steps were implemented to filter the in English. To observe the results, we decided to plot the topics most similar to keywords and resulting topics to plot as a word cloud of words that represent this topic. The test was conducted using doc2vec and USE.

Presented results in Figures 4.2 and 4.1 allow us to say that top2vec captures the general topics and some intricate specifics of the field presented in the dataset.

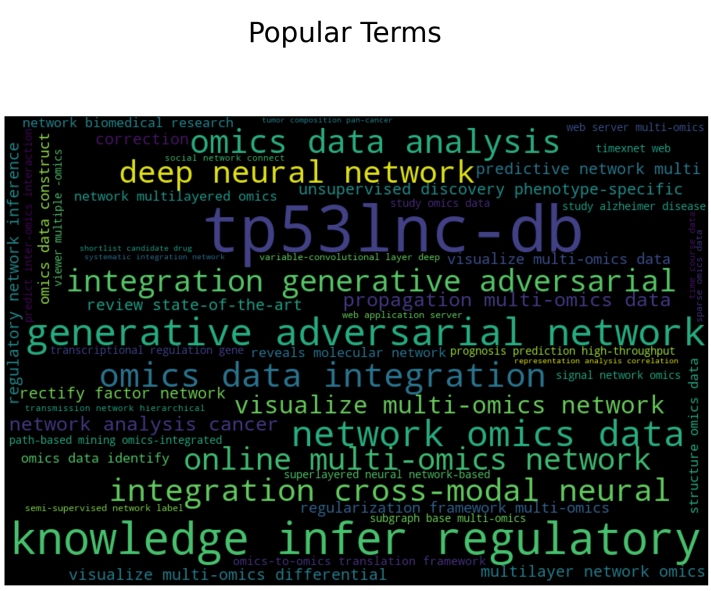


Figure 4.1: Word cloud of a topic with the highest similarity score for the keyword "biology."

Figure 4.2: Word cloud of a topic with the highest similarity score for the keyword "malware."

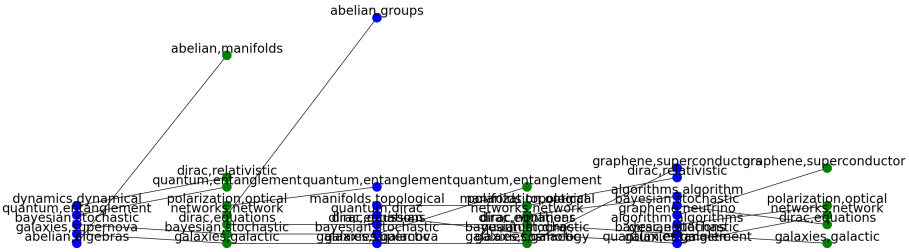


Figure 4.3: Bipartite graph, formed by comparing topics from 6 top2vec models, trained on different time slots.

One way to track the topics' behavior through time is to train separate models for every time window (year, month) and use techniques to connect most similar topics.

The latter experiments will be conducted mainly with USE embedding because doc2vec will not produce consistent embedding spaces, meaning that every model, even trained on the same dataset, will yield different embeddings for the same document. Unfortunately, it is very crucial for us since there is no straightforward way to compare those embeddings or the documents they represent together. There is a walk-around, which requires using an older implementation of doc2vec and pre-trained word2vec. This will make the document vector representation more consistent since all word vectors will be the same for every model.

The following experiment took one year of articles and split them into six subsets every two months. The current dataset, dblp, does not contain the metadata that would allow us to break the data in the way we described above, so we found another dataset from Kaggle arXive [23], which contains the exact date of publication for every article.

We trained six top2vec models with USE as an embedding method using this data. We then iterated through each in time order, computing for every two consecutive models the maximum cosine similarity measure between topics of the models t and $t + 1$. Based on this, we construct a graph where nodes are topics from model m_t, and edges are created with the node from model m_{t+1} with the maximum cosine similarity. We plot this graph as bipartite to see the connection it forms. In the resulting bipartite graph, we used only edges with cosine similarity greater than 0.6.

The result is presented as multiple trees representing the evolution of topics through time. This evolution can be named a generalization since the above procedure catches how topics merge into one through time.

To observe the activity of individual trends for every two months of data, we calculated how many documents every time point had. We then plotted the activities of every trend and observed their behavior over time. The computed trends represent the generalization of the research articles because we started comparing the similarity between the first two neighboring models.

Exciting results will emerge if we start from the end instead of computing from the beginning of the dataset timeline. Since we will see how the topics from the latest time model will merge into one from the beginning, this operation results in the research article's trends specification.

Our method overcomes several challenges associated with research trend detection. First, by using only the titles of research papers, we reduced the complexity of the data and improved computational efficiency. Second, splitting the dataset into two-month intervals allowed us to track trends over time, providing insights into their emergence, evolution, and decline. Third, the use of top2vec models and graph representations facilitated the identification and visualization of trends in an interpretable manner.

In conclusion, our research trend detection method offers a novel and practical ap-

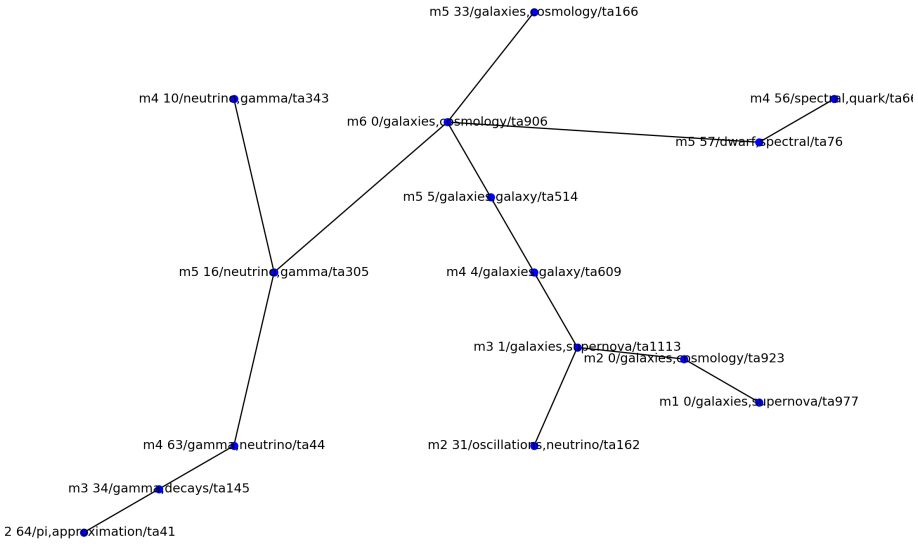


Figure 4.4: Trees that show how topics generalized through time, formed by comparing topics from 6 top2vec models, trained on one year of data splitted on 6 different time slots.

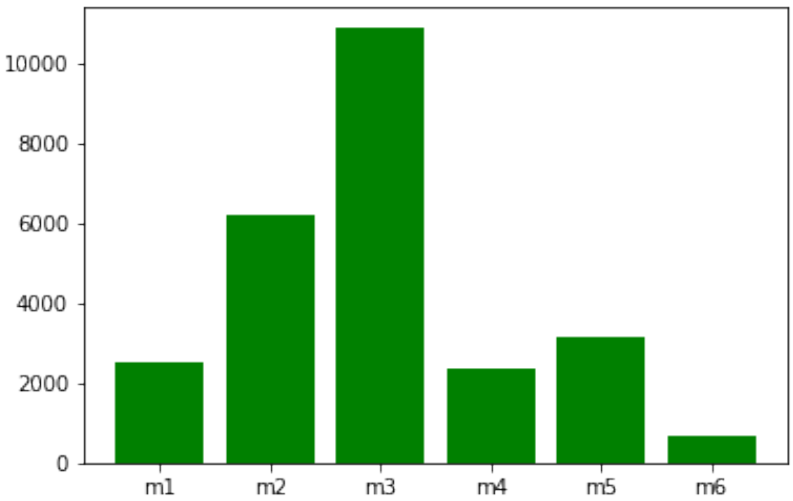


Figure 4.5: Plot the shows forward trend activity over time.

proach to this task, addressing fundamental challenges and providing valuable insights into emerging trends in research publications. The combination of top2vec models, the USE embedding method, and graph representations enables efficient analysis of extensive collections of research papers and facilitates the identification of trends.

4.3 Implementation of the reviewed method

In this section, we discuss the implementation and finding of one of the methods for topic detection presented in Section 3.5.

The article written by Xiao, Qian, and Qin in 2021 describes in detail the method they created. Unfortunately there is not mention of any successful implementation of it.

Through out our research we were using python programming language. Since, it already contains a lot of implemented statistical and algorithmic solutions. So we decided to go with this language for the implementation.

There are a few issues with the description that we should mention. The similarity kernel function 3.2, which should compare two documents graph representation between each other and yield a result in range between 0 and 1. After many experiments we could not confirm this statement. Even though in the denominator of the function there is a normalization factor, which should prevent any overflow it seems like it is not enough. Depending on the number of nodes in CSG we could see values that exceeded the range.

Some changes were done in the implementation. In the first part of the method description authors constructed the word co-occurrence graph based on occurrences of individual words in a sentence. Instead we decided to use the mutual information measure.

Mutual information (MI) measures the degree of dependence or association between two variables. Mutual information estimates how much information is shared between them. It is defined as the amount of information obtained about one variable when the value of the other variable is known, compared to the amount of information received when the value of the other variable is unknown. Mathematically it is defined as:

$$I(X;Y) = H(X) - H(X|Y), \tag{4.1}$$

where $H(X)$ and $H(Y)$ are the marginal entropies of X and Y, respectively, and $H(X|Y)$ and H(Y—X) are the conditional entropies of X given Y and Y given X, respectively. $H(X,Y)$ is the joint entropy of X and Y. Intuitively, MI measures how much the knowledge of one variable reduces the uncertainty about the other variable. If the variables are independent, then the value of MI is 0. If they are perfectly dependent, then the value of MI is equal to the entropy of one of the variables (e.g., $H(X)$ if X and Y are perfectly dependent).

Second part of the method described how to use the word graph to create a CSG by using using a community detection algorithm defined in the paper, which is based on finding maximum betweenness. Maximum betweenness of a node is a measures of how many shortest path go through this node. Based on this certain edges were eliminated to create separate communities.

For our implementation we decided to go with more newer approach for community detection called Louvain algorithm. We gave more description in Section 2.1.1.

Last part of the method, shows a new approach for graph comparison introducing the graph kernel function. As well as showing the clustering algorithm that will use the graph kernel metrics to cluster the data.

From our point of view the clustering algorithm lacks the description of a centroid recomputation, which is crucial part for this algorithm. Since, it will directly influence the formation of new clusters as well as adding a new document to already existing one.

Given that we could not implement the clustering algorithm. We decided to substitute it with Agglomerative Hierarchical Clustering. This bottom-up approach starts by considering each data point as a separate cluster and then progressively merges the closest clusters based on a chosen similarity measure. It creates a hierarchy of clusters that can be cut at different levels to obtain other numbers of clusters.

For this algorithm to work we need to create a similarity, distance matrix, which will contain a measure of distance or in our case similarity between every document. Unfortunately computing this matrix is very time consuming, which will result in some experiments scale decrease.

Chapter 5

Results

5.1 Clustering algorithms comparison

This section describes the experimental setup. For the next experiment, we have used a 20 newsgroups dataset from Pedregosa et al., creators of scikit-learn library, as well as arXiv dataset [23] available on Kaggle.

In our first experiment, we will check the difference between top2vec clustering and the method proposed by Xiao, Qian, and Qin in 2021. In Section 4.3, we described our way of implementing the method and the challenges we have faced during this process.

	Calinski-Harabasz Index	Davies-Bouldin Index	Silhouette Score
top2vec	0.0167	3.9887	46.2261
method by Xiao, Qian, and Qin	-0.0283	14.8472	2.8813

Table 5.1: Comparison results of two clustering methods is done on 20 newsgroup dataset.

In the custom method 4.3, we have used "scipy" library implementation with a scalar parameter set to 20 and criterion to "maxclust". For the top2vec model we used universal-sentence-encoder (USE) as our embedding method, we have not changed any other parameters. In the other two experiments, we will use top2vec with the USE only.

To compare these two algorithms, we used a few evaluation scores measuring different clustering performance aspects.

The first one is Silhouette Score. The silhouette score measures the clustering quality by assessing how well each sample fits within its own cluster compared to other clusters. It ranges from -1 to 1, where a higher value indicates better-defined and well-separated clusters. A score close to 1 suggests that the samples are assigned to the correct clusters, while a score close to -1 indicates that the samples may have been assigned to the wrong clusters.

The second one is Davies-Bouldin Index. The Davies-Bouldin index measures the average similarity between each cluster and its most similar cluster. It is calculated based on the within-cluster scatter and the between-cluster separation. The index ranges from 0 to infinity, where a lower value indicates better clustering. A lower index suggests that the clusters are well-separated and distinct from each other.

The last one is Calinski-Harabasz Index. The Calinski-Harabasz index, also known as the Variance Ratio Criterion, measures the ratio between cluster and within-cluster dispersion. It evaluates the compactness and separation of clusters. A higher index value

indicates better-defined and well-separated clusters. The index is higher when clusters are dense and well-separated and lower when clusters overlap or are poorly separated.

These evaluation scores provide different perspectives on the performance of clustering algorithms. However, it's important to note that these scores have limitations. They may not always capture the complete picture of cluster quality, and their interpretation can be subjective depending on the dataset and problem domain. Therefore, it's advisable to consider multiple evaluation metrics and use domain knowledge to assess the clustering results comprehensively.

	Calinski-Harabasz Index	Davies-Bouldin Index	Silhouette Score
top2vec	0.0167	3.9887	46.2261
method by Xiao, Qian, and Qin	-0.0883	1.1237	0.9940

Table 5.2: Comparison results of two clustering methods on arXiv dataset from Kaggle.

From the results in Table 5.2, we can say that the Silhouette Score for Top2Vec is positive, indicating that the clustering has some degree of separation between the samples. However, the score is quite low, suggesting that the clusters may not be well-defined or distinct.

The Davies-Bouldin Index for Top2Vec is relatively high, indicating that the clusters may have overlapping or poorly separated regions. A higher index value suggests less optimal clustering.

The Davies-Bouldin Index for Top2Vec is relatively high, indicating that the clusters may have overlapping or poorly separated regions. A higher index value suggests less optimal clustering.

As for the method by Xiao, Qian, and Qin, The Silhouette Score for the custom method is negative, indicating that the clustering results are not well-defined and the samples may be incorrectly assigned to clusters.

The Davies-Bouldin Index for the custom method is relatively low, suggesting good separation and distinction between the clusters. A lower index value suggests better clustering.

The Calinski-Harabasz Index for the custom method is relatively low, indicating that the clusters may not be well-separated and exhibit clear patterns or structures.

Top2Vec performs slightly better than the custom method, with a positive Silhouette Score and a higher Calinski-Harabasz Index, indicating better separation and well-defined clusters. The custom method has a negative Silhouette Score and a lower Calinski-Harabasz Index, suggesting less optimal clustering and lower separation between clusters.

5.2 Trend detection discussion

In this experiment of the diploma thesis, we focused on extracting and analyzing a plot of trend activity over time to demonstrate the effectiveness of our approach in capturing meaningful trends.

To conduct this experiment, we utilized the processed dataset and applied our trend detection pipeline, as described in Section 4.2. By leveraging the pipeline's capabilities, we were able to identify and track trends over the selected time period.

The plot obtained from this experiment, Figure 5.1, presents a visualization of the trend activity over time. Each point on the plot represents a specific time period, while the vertical axis denotes the intensity or activity level of the identified trend during that

period. This graphical representation 5.1 includes labeled lines for each trend, showcasing the top three TF-IDF words extracted from documents associated with each trend.

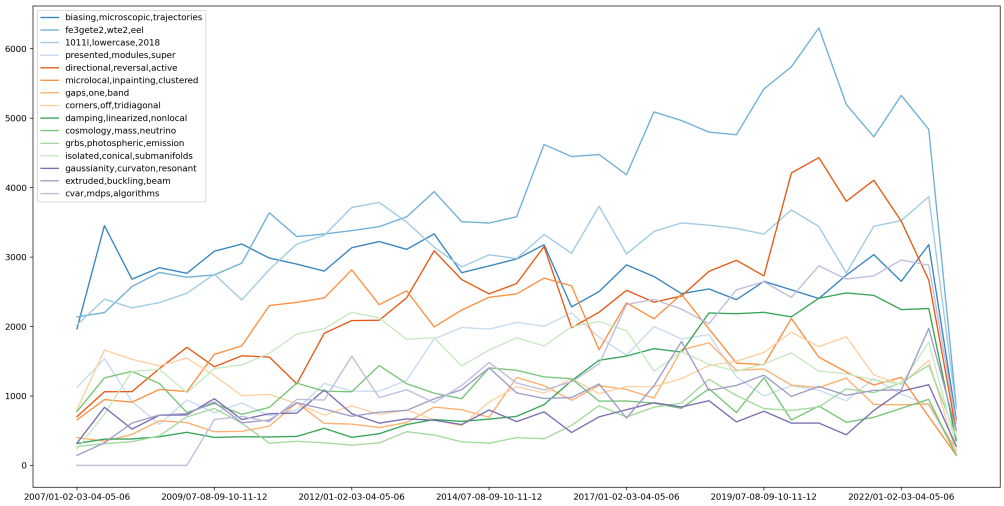


Figure 5.1: Forward direction of the trend, the data is split for every 6 months.

By carefully examining the plotted trend activity, we can observe distinct patterns and fluctuations, indicating the presence of significant trends in the research domain. Peaks in the plot signify periods of heightened activity, suggesting increased interest and focus on specific research topics. Conversely, valleys or plateaus may indicate periods of relative stability or reduced attention to particular areas of study.

Among these trends, a specific one, with the words "Fe3GeTe2" and "WTe2," caught our attention due to their potential connection with magnets.

Fe3GeTe2 and WTe2 are materials that have garnered significant interest in the field of magnetism. By examining the plotted trend activity and the associated TF-IDF word labels, we can gain deeper insights into their relationship with magnetic properties and potential applications.

Fe3GeTe2, known as a ferromagnetic material, exhibits intriguing magnetic properties due to its unique crystal structure and spin configurations. The presence of Fe (iron) suggests a potential connection with ferromagnetism, while Ge (germanium) and Te (tellurium) are elements known for their semiconducting properties. The appearance of "Fe3GeTe2" as a prominent word in the trend's label implies a substantial interest in this material and its magnetic characteristics within the research community.

Similarly, WTe2, or tungsten ditelluride, has attracted considerable attention due to its exotic electronic and magnetic properties. It is known to exhibit a variety of phases, including a nontrivial semimetallic state and superconductivity under specific conditions. The inclusion of "WTe2" as a significant word in the trend's label suggests ongoing research efforts to explore its magnetic and electronic properties and uncover potential applications in areas such as spintronics or quantum computing.

Analyzing the trend activity plot and the TF-IDF word labels provides a comprehensive overview of the research community's interest in Fe3GeTe2 and WTe2 as materials with magnetic properties. This graphical representation underscores the significance of these materials in the field of magnetism and motivates further exploration and investigation into their unique characteristics and potential technological applications.

This experiment demonstrates the utility of graphical representations in capturing and visualizing trends effectively. It enables the identification of emerging areas of study, the tracking of evolving trends, and the assessment of their impact and significance.

Overall, our pipeline presents meaningful trends through graphical representations. By utilizing these plots in conjunction with other analytical techniques, researchers can gain valuable insights into the dynamics of the research landscape and make informed decisions regarding resource allocation, collaboration opportunities, and future directions of study.

5.3 Impact of Time Window Size

In continuation of the previous experiments, we employ the arXiv dataset for this investigation. The primary objective of this experiment is to assess the influence of varying the time window size on the effectiveness of the proposed method outlined in Section 4.2.

Understanding the relationship between the window size and the resulting trends' number and activity is of significant interest. The time window serves the purpose of dividing the timeline into evenly sized chunks, enabling the identification of topics within each chunk. Moreover, it facilitates the exploration of topic relationships across adjacent chunks, offering insights into their temporal behavior. The choice of window size plays

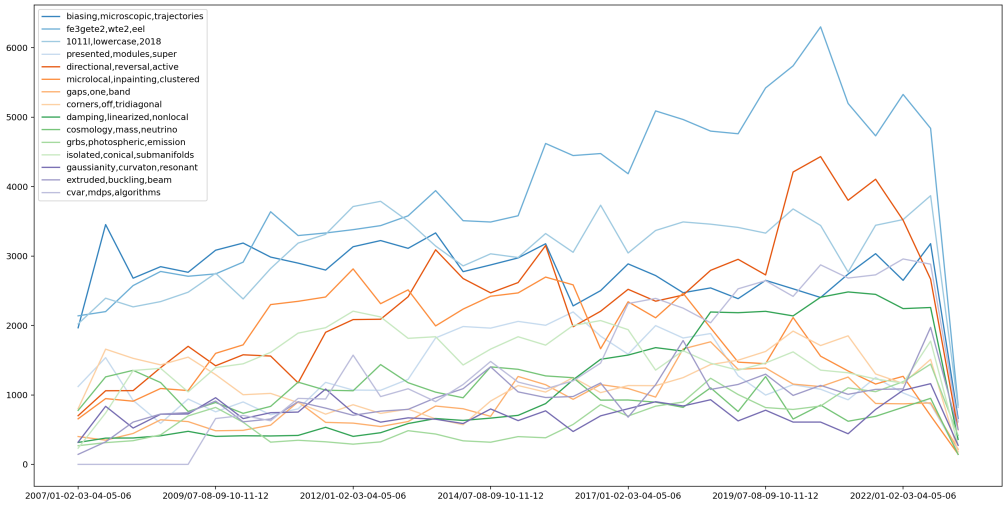


Figure 5.2: Forward direction of the trend, the data is split for every 6 months.

a pivotal role in determining the number of documents encompassed within each chunk. If a chunk spans an excessive number of documents, the subsequent clustering process may yield overly abstract topics instead of specific and focused ones. This abundance of abstract topics can lead to a decrease in the number of detected trends as all topics become connected to a single overarching abstract theme.

Additionally, it is important to note that selecting an excessively small window will have an impact as well. A smaller window size may result in a higher level of noise, Figures 5.4 and 5.5, within the trends detected by the proposed method. This increased noise can hinder the accurate identification of actual trends as the signal-to-noise ratio diminishes. Consequently, the observed trends may predominantly reflect the noise present in the dataset rather than capturing meaningful and significant research directions. Hence,

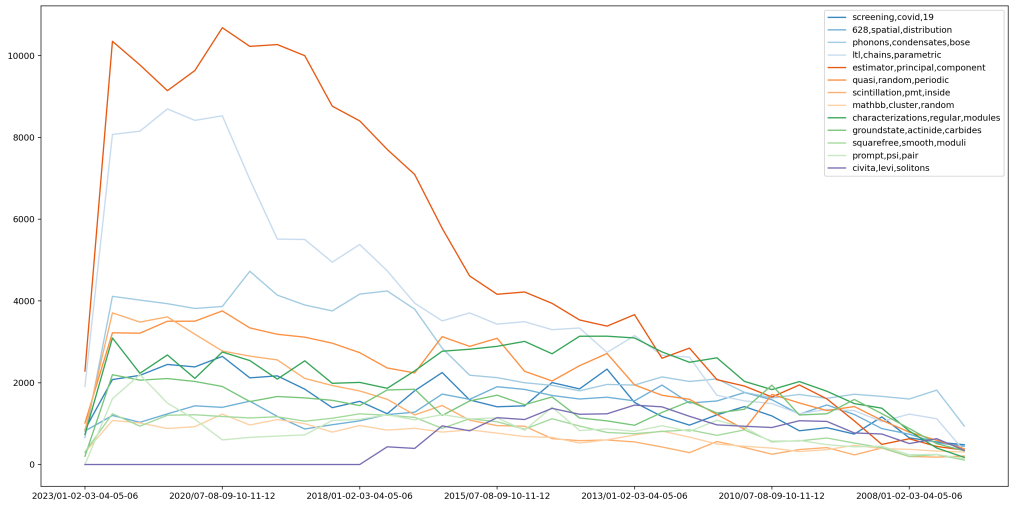


Figure 5.3: Backward direction of the trend, the data is split for every 6 months.

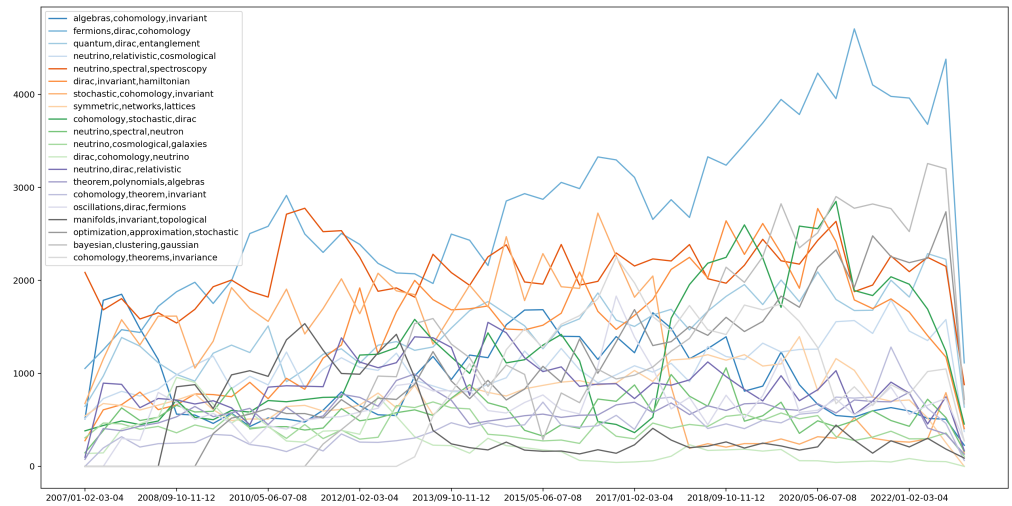


Figure 5.4: Forward direction of the trend, the data is split for every 4 months.

striking the right balance in choosing the window size is crucial to ensure that the proposed method captures genuine trends while minimizing the impact of noise in the analysis.

Therefore, it is crucial to select an appropriate window size to achieve meaningful trend detection. The window size should strike a balance between capturing sufficient contextual information within each chunk and avoiding an excessive amalgamation of documents that could dilute the specificity of identified trends. Fine-tuning the window size parameter is vital to ensure the pipeline's efficacy in identifying distinct and representative trends within the research landscape.

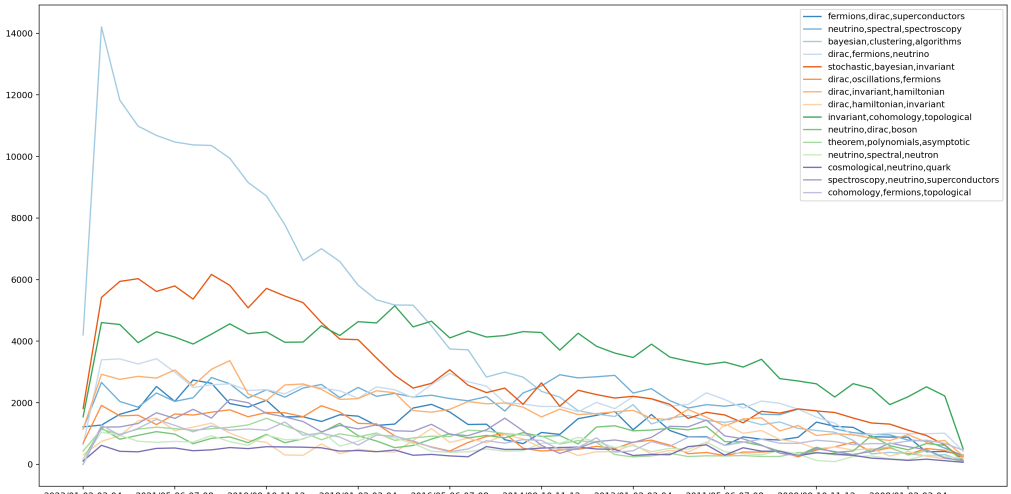


Figure 5.5: Backward direction of the trend, the data is split for every 4 months.

Chapter 6

Conclusion

In conclusion, this diploma thesis focused on the topic of research trend detection and presented a pipeline designed to observe these trends using article titles as input. Throughout the research process, several important findings and challenges emerged, shedding light on the complexities of trend detection in the academic domain.

One of the primary challenges encountered during this study was the absence of a definitive measure for trend detection. Research trends can be subjective, making it difficult to establish a universally applicable criterion for their identification. Nevertheless, by leveraging various techniques and methodologies, this thesis aimed to provide a reliable framework for trend detection, offering valuable insights into the ever-evolving landscape of scientific research.

It is worth noting that the performance of the pipeline introduced in this thesis may be influenced by the selection of parameters and the dataset used. Different parameter settings or alternative datasets might yield varying results, potentially leading to misbehavior or skewed trend observations. Therefore, it is crucial to exercise caution and conduct thorough validation and fine-tuning of the pipeline to ensure its optimal functionality and accuracy.

Future research should focus on refining the existing methodologies, incorporating additional data sources, and investigating alternative approaches to mitigate the limitations highlighted in this study.

In conclusion, this diploma thesis has provided valuable insights into the complexities of research trend detection. While acknowledging the absence of a definitive measure and the potential pitfalls of the proposed pipeline, it serves as a stepping stone for future endeavors aimed at understanding and forecasting the dynamic nature of scientific progress. By continuing to refine and expand upon the methodologies presented here, researchers can contribute to the advancement of knowledge and facilitate informed decision-making in various domains impacted by research trends.

Bibliography

[1] Dimo Angelov. "Top2Vec: Distributed Representations of Topics". In: (2020). arXiv: 2008.09470 [cs.CL].

[2] Wojciech Banaszczyk. "On a certain class of positive definite functions and measures on locally compact Abelian groups and inner-product spaces". In: *Journal of Functional Analysis* 282.1 (2022), p. 109261. ISSN: 0022-1236. DOI: https://doi.org/10.1016/j.jfa.2021.109261. URL: https://www.sciencedirect.com/science/article/pii/S0022123621003438.

[3] Sahar Behpour et al. "Automatic trend detection: Time-biased document clustering". In: *Knowledge-Based Systems* 220 (2021), p. 106907. ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2021.106907. URL: https://www.sciencedirect.com/science/article/pii/S0950705121001702.

[4] Yoshua Bengio et al. "A Neural Probabilistic Language Model". In: *J. Mach. Learn. Res.* 3 (2003), pp. 1137–1155.

[5] Yasuyuki Hamura and Tatsuya Kubokawa. "Bayesian predictive density estimation for a Chi-squared model using information from a normal observation with unknown mean and variance". In: *Journal of Statistical Planning and Inference* 217 (2022), pp. 33–51. ISSN: 0378-3758. DOI: https://doi.org/10.1016/j.jspi.2021.07.004. URL: https://www.sciencedirect.com/science/article/pii/S0378375821000719.

[6] Huang (Steeve) Kung-Hsiang. *Word2Vec and FastText Word Embedding with Gensim*. Feb. 2018. URL: https://towardsdatascience.com/word-embedding-with-word2vec-and-fasttext-a209c1d3e12c.

[7] Philip Lawrence et al. "Measuring Respiratory Function". In: *Encyclopedia of Respiratory Medicine (Second Edition)*. Ed. by Sam M Janes. Second Edition. Oxford: Academic Press, 2022, pp. 42–58. ISBN: 978-0-08-102724-0. DOI: https://doi.org/10.1016/B978-0-08-102723-3.00243-2. URL: https://www.sciencedirect.com/science/article/pii/B9780081027233002432.

[8] Minh-Hoang Le, Tu Bao Ho, and Yoshiteru Nakamori. "Detecting Emerging Trends from Scientific Corpora". In: 2006.

[9] Quoc V. Le and Tomas Mikolov. *Distributed Representations of Sentences and Documents*. 2014. arXiv: 1405.4053 [cs.CL].

[10] Xiang Liu, Tingting Jiang, and Feicheng Ma. "Collective dynamics in knowledge networks: Emerging trends analysis". In: *Journal of Informetrics* 7.2 (2013), pp. 425–438. ISSN: 1751-1577. DOI: https://doi.org/10.1016/j.joi.2013.01.003. URL: https://www.sciencedirect.com/science/article/pii/S1751157713000059.

[11] Wei Lu et al. "Detecting research topic trends by author-defined keyword frequency". In: *Information Processing Management* 58.4 (2021), p. 102594. ISSN: 0306-4573. DOI: https://doi.org/10.1016/j.ipm.2021.102594. URL: https://www.sciencedirect.com/science/article/pii/S0306457321000935.

[12] Benjamin Mahieu et al. “A multiple-response chi-square framework for the analysis of Free-Comment and Check-All-That-Apply data”. In: *Food Quality and Preference* 93 (2021), p. 104256. ISSN: 0950-3293. DOI: https://doi.org/10.1016/j.foodqual.2021.104256. URL: https://www.sciencedirect.com/science/article/pii/S0950329321001397.

[13] Michael Mathioudakis and Nick Koudas. “TwitterMonitor: Trend Detection over the Twitter Stream”. In: *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*. SIGMOD ’10. Indianapolis, Indiana, USA: Association for Computing Machinery, 2010, pp. 1155–1158. ISBN: 9781450300322. DOI: 10.1145/1807167.1807306. URL: https://doi.org/10.1145/1807167.1807306.

[14] Tomas Mikolov et al. “Efficient Estimation of Word Representations in Vector Space”. In: *International Conference on Learning Representations*. 2013.

[15] Tomás Mikolov et al. “Efficient Estimation of Word Representations in Vector Space”. In: *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2013. URL: http://arxiv.org/abs/1301.3781.

[16] Youssef Mourchid, Mohammed El Hassouni, and Hocine Cherifi. “A New Image Segmentation Approach using Community Detection Algorithms”. In: vol. 8. Dec. 2015. DOI: 10.1109/ISDA.2015.7489194.

[17] Hidetsugu Nanba and Manabu Okumura. “Towards Multi-paper Summarization Using Reference Information”. In: *International Joint Conference on Artificial Intelligence*. 1999.

[18] Christopher Olah. *Understanding LSTM Networks*. 2015. URL: http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[19] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[20] Nuran Peker and Cemalettin Kubat. “Application of Chi-square discretization algorithms to ensemble classification methods”. In: *Expert Systems with Applications* 185 (2021), p. 115540. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2021.115540. URL: https://www.sciencedirect.com/science/article/pii/S0957417421009477.

[21] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. “GloVe: Global Vectors for Word Representation”. In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1532–1543. URL: http://www.aclweb.org/anthology/D14-1162.

[22] A. Saavedra and V. Opfer. *A Global Cities Education Network Report. A Global Cities Education Network Report.* New York, Asia Society., 2012.

[23] arXiv.org submitters. *arXiv Dataset*. 2023. DOI: 10.34740/KAGGLE/DSV/4983956. URL: https://www.kaggle.com/dsv/4983956.

[24] Basuki Wibawa Uwes Anis Chaeruman and Zulfiati Syahrial. “Determining the Appropriate Blend of Blended Learning: A Formative Research in the Context of Spada-Indonesia.” In: *American Journal of Educational Research* 6.3 (1905), pp. 88–195. DOI: http://dx.doi.org/10.12691/education-6-3-5..

[25] Kejing Xiao, Zhaopeng Qian, and Biao Qin. “A graphical decomposition and similarity measurement approach for topic detection from online news”. In: *Information Sciences* 570 (2021), pp. 262–277. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2021.04.029. URL: https://www.sciencedirect.com/science/article/pii/S002002552100356X.

Appendix A

Appendix title