technische universität dortmund

fakultät statistik

Master Thesis

Topic detection in natural language processing using clustering and semantic analysis techniques

Alp Yalçın

Enrolment Number: 230888

Course of Studies: MSc. Data Science

Registration Date:

01.03.2023

Submission Date:

01.09.2023

Academic Advisors:

Prof. Dr. Markus Pauly

Dr. Rolf Bardeli

Contents

List of symbols **III**

List of figures **III**

List of tables **IV**

1 Introduction **1**

2 Comparative Approaches and Different Paths in Textual Analysis **2**

3 Methods **3**

3.1 Dataset explanation 3

3.1.1 DBPEDIA dataset 3

3.1.2 Text preprocessing 6

3.2 Sentence Embedding 10

3.3 Clustering methods and dimensionality reduction 22

3.3.1 Principal Component Analysis (PCA) 23

3.3.2 K-Means Clustering 26

3.3.3 HDBSCAN 28

4 First Exploration and Baseline Analysis on Topic Detection **34**

4.1 Clustering Methods 34

4.2 Semantic Graphs for L1 Topic Classes 37

5 Improvements on implementation and more granular analysis on topic detection **47**

5.1 Advancing Semantic Analysis: The Louvain Method for Community Detection 47

5.2 Advancing Topic Detection: Hierarchical Clustering Combined with HDBSCAN 49

5.3 In-depth Analysis and Refinement of Hierarchical Clusters 51

5.3.1 Agent label Subcluster analysis 53

5.3.2 Species label Subcluster analysis 58

6 Enhanced Interactive Tools for Topic Detection Analysis 61

6.1 Local Analysis: Cluster Selection with Hierarchical Clustering and HDBSCAN 61

6.2 Nested Hierarchical Clustering: Subtopic Analysis with Dual Thresholds . 64

7 Conclusion 66

References 68

Appendices 71

List of Figures

1 L1 label classes for training data 4
2 Text lengths for each classes 5
3 Label in the sentence 6
4 Scaled dot-product attention and Multi-head attention [17] 12
5 Transformers Architecture [17] 15
6 Input Representation [13] 19
7 Pre-training of BERT [13] 20
8 Neural Network architecture for embeddings 22
9 Probability Density Function for each classes in DBPedia 29
10 Calculating PDF methods [24] 30
11 Hiearchical Structure of HDBSCAN [24] 30
12 HDBSCAN clustering 35
13 KMeans clustering 36
14 Process of acquiring semantic layers 37
15 Example of semantic layers for the word 'institution' 38
18 Embedding space colors for l1 class 40
16 Agent class 3 levels of hypernym graphs 44
17 Subgroups for Agent, Place and Event classes 45
19 Visualization of TF-IDF Scores for Selected Community Groups within the Embedding Space 46
20 Semantic graph for Place class with Louvain community 49
21 Hierarchical Clustering Combined with HDBSCAN 52
22 Text Pairs for subclusters 10 and 12 56
23 Analysis of text pairs for cluster 10 and cluster 12 57
24 Species Subclusters in the Embedding Space 58
25 Text pair similarity analysis for subclusters 1 and 2 59
26 User Interface for Cluster Selection: Interactive sliders for threshold and cluster selection enable users to control the number of topics and the analysis of semantic graphs. 62
27 Local Analysis Tool: An illustration of the interactive tool that allows users to perform a specific examination within the embedding space, selecting text observations and analyzing neighboring texts. 63
28 Dual Threshold Hierarchical Clustering: A visual representation of primary clustering (left) and nested subclustering (right) based on selected Threshold 1 and Threshold 2 values 64

List of Tables

1 Synsets and hypernyms explanation table 9
2 Explanation of Scaled Dot Product Attention Mechanism in BERT 13
3 K-Means Clustering Algorithm 27
4 Place Class Subclusters 53
5 Agent Class Subclusters 55
6 Species Class Subclusters 58
7 Work, Event, SportSeason and Topical Concept,Device,UnitOfWork Class Subclusters 71

1 Introduction

In the digital era, the exponential growth in textual data has necessitated effective methods for detecting and classifying topics within large, unclassified text datasets. This task is crucial across various fields, including social media analytics, customer feedback analysis, and automated news categorization. However, traditional approaches, like Term Frequency-Inverse Document Frequency (TF-IDF), struggle to fully capture the semantics of text.

This thesis seeks to address the challenge of semantic understanding in topic detection tasks, aiming to capture the rich, context-dependent semantics of language. The proposed solution is an integrated approach that employs advanced clustering algorithms, specifically Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN), and semantic network analysis built upon synonyms and hypernyms, to navigate the semantic complexities of large text corpora.

The research aims to develop a comprehensive topic detection algorithm by merging clustering and semantic analysis techniques. This hybrid approach uses sentence embeddings, generated by a pretrained model mpnet-base [1], to transform the text into a high-dimensional vector space suitable for detailed semantic analysis. The research also introduces an innovative application of the Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) [2] algorithm for text data clustering, offering a nuanced understanding of the topics and subtopics within the corpus. Moreover, the study intends to create user-friendly interactive tools to fine-tune the analysis parameters for hierarchical clustering, thereby providing a flexible solution for topic detection.

The thesis unfolds as follows: Chapter 2 reviews another master thesis project that explores the same topic from a different perspective and shares a common starting point of analysis. Chapter 3 explicates the adopted methodologies, including sentence embeddings, clustering, and dimensionality reduction techniques. Chapter 4 presents the results of the initial experiment, focusing on clustering methods and the creation of semantic graphs. Chapter 5 discusses enhancements to the initial implementation and an in-depth analysis of hierarchical clusters. Chapter 6 introduces enhanced interactive tools designed for detailed topic analysis. Finally, Chapter 7 concludes the study, summarizing the findings, delineating the contributions and limitations, and suggesting potential avenues for future research.

2 Comparative Approaches and Different Paths in Textual Analysis

In this chapter, the dynamic landscape of methodologies for textual data analysis is explored, encompassing both established paradigms and the divergent paths that have emerged from them. The foundational standpoint for our investigations is found in the work of Pandya [22], whose master's thesis, titled 'Topic-aware approaches to Natural Language Processing'.

Pandya's work [22] presents a comparison between conventional TF-IDF representation and transformer-based BERT models, yielding insights into the encoding of corpus documents. It highlights poignant concerns regarding the complexities of semantic interpretation within natural language, shedding light on challenges such as differentiating homonyms like "Apple" in divergent contexts. The limitations of these established techniques underscore the significance of the current study, which strives to transcend these challenges through advanced semantic analysis techniques, effectively capturing the nuanced contextual meanings of words.

While the focus of Pandya's work predominantly gravitates toward the supremacy of supervised Neural Network (NN)-based methods, this study embarked upon takes a distinct trajectory, emphasizing the potential of unsupervised techniques and semantic network analysis. By applying these methods, a balance between model performance and computational efficiency is sought

Furthermore, paper [22] highlighted the interpretability challenge in deep learning models and proposed the Disentangling Invertible Interpretation Network (DIIN) [27] as a solution for textual data. The importance of interpretability is recognized in the current study, echoing this effort to improve it. However, the approach taken extends this concept by emphasizing the use of explainable machine learning techniques, which allow for a more transparent understanding of model decisions and foster trust in the findings.

While Pandya's academic paper Pandya [22] serves as a valuable cornerstone, this master thesis ventures towards a different perspective on the same problem. It builds on the foundation laid by Pandya by addressing the identified limitations and aiming to develop a method that is not only effective but also balanced in terms of efficiency and transparency.

3 Methods

The Python programming language was employed for all computations and analyses in this Master's thesis project. Python's extensive ecosystem of libraries significantly facilitated data manipulation, analysis, and visualization. The libraries used included `numpy` [3], `pandas` [4], `nltk` [5], `matplotlib` [6], `sklearn` [7], `re` [8], `transformers`, `tensorflow` [9], `hdbscan` [2], `networkx` [10], `pylab`, `community.community_louvain`, `scipy`, `collections`, `ast`, `IPython.display`, and `ipywidgets`.

3.1 Dataset explanation

It is imperative to identify an appropriate dataset for studying topic detection in natural language processing tasks. The dataset must be sufficiently intricate, encompassing full-length texts, in order to facilitate the discovery of a comprehensive topic detection solution applicable in real-world scenarios. Accordingly, text observations within the dataset should display semantic diversity and possess varying lengths.

In order to satisfy all these criteria, the DBPEDIA dataset from Kaggle [11] [12] has been chosen for this Master's thesis project. Various applications and methods will be investigated using this dataset.

3.1.1 DBPEDIA dataset

The DBpedia dataset from Kaggle offers taxonomic hierarchical classes for 337 739 Wikipedia articles. This dataset is divided into three distinct subsets: training, validation, and testing, containing 240 942, 36 003, and 60 794 observations respectively. It is comprised of three levels L1, L2 and L3 with 9, 70, and 219 classes. Each level contains increasingly detailed topic names for texts. For instance, an example of the taxonomic classes assigned to one selected observation is provided below:

'Un Millón de Rosas is the nineteenth studio album released by American band La Mafia. It was released on January 30, 1996 by Sony Music Entertainment. The album peaked at number one on the Billboard Regional Mexican Albums chart and also reached the top ten on the Billboard Top Latin Songs chart. Un Millón de Rosas earned them the Grammy Award for Best Mexican/Mexican-American Album at the 39th Grammy Awards. At the 9th Lo Nuestro Awards, it received a nomination for Regional Mexican Album of the Year.'

For this particular observation, the L1 label is 'Work', indicating that the text relates to

the ‘Work’ topic. The L2 label provides a more specific topic name, ’MusicalWork,’ while the L3 label, ’Album,’ further refines the topic name for the selected text. As can be seen from this example, labels are structured in a hierarchical order, which proves beneficial for semantic analysis. Predominantly, the L1 labels will be the focus for modeling applications in this task to identify more specific and granular topics for each text. The L2 and L3 labels are used to analyze how the new topics identified from our analysis relate to the original labels assigned in the dataset.

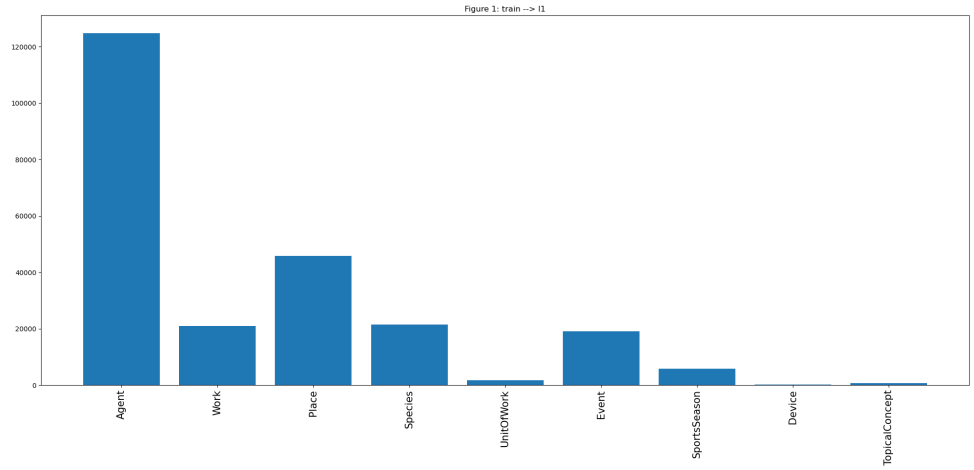


Figure 1: L1 label classes for training data

The training, validation, and testing datasets are structured to preserve consistent proportions of class densities within each. By adopting this approach, balanced representation of each class is ensured across all three datasets, which is critical for the reliability of the modeling analysis. The DBPedia dataset exhibits highly imbalanced label classes, as can be seen in the Figure 1.

In the training dataset, the ’Agent’, ’Place’, ’Species’, ’Work’, and ’Event’ classes encompass 124 978, 45 877, 21 472, 21 013, and 19 106 text observations respectively. Conversely, the ’SportSeason’, ’UnitOfWork’, ’TopicalConcept’, and ’Device’ classes have considerably fewer text observations, with counts of 5 883, 1 761, 784, and 248 respectively.

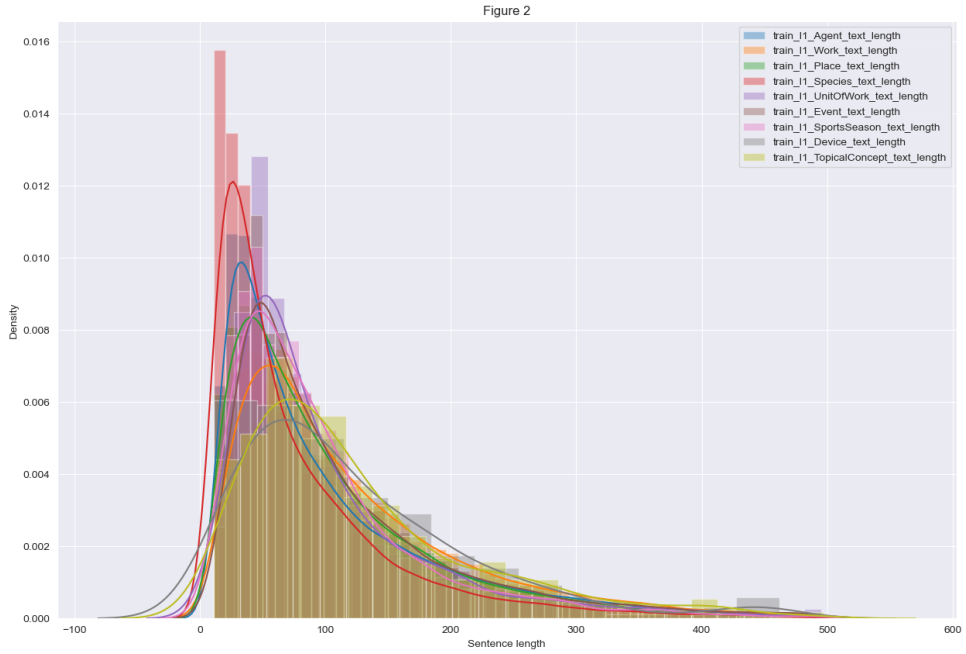


Figure 2: Text lengths for each classes

To discern potential patterns or disparities in text lengths based on their respective labels, distribution plots for each class are utilized. The distribution of text lengths for the classes is depicted in Figure 2. Upon examination of label-specific text lengths, the 95th percentile of text lengths for each label ranges from 241 to 322 words. The 'Species' class has the shortest percentile score, while the 'TopicalConcept' class has the longest, as can also be observed from the figure.

Additionally, one significant consideration that arises when examining the class names and text observations concurrently is the presence or absence of these class names within the text observations. As depicted in Figure 3, L1 labels are found in merely 12% of the texts, whereas L2 label names appear in 14.4% of the text observations, and L3 label names exist in 36.6% of the texts. These results suggest that the likelihood of identifying topic names within the text sentences increases when considering more specific levels of topic names.

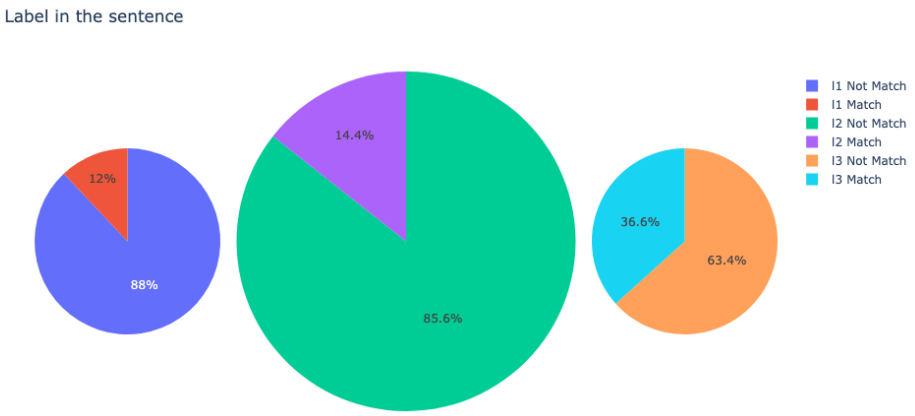


Figure 3: Label in the sentence

3.1.2 Text preprocessing

Text preprocessing is a crucial aspect of this project, significantly contributing to semantic analysis and topic detection. It entails the transformation of text data into a format more suitable to analysis and machine learning algorithms, thereby improving the accuracy and efficiency of NLP tasks. As it is explained in the previous section, the DBPedia dataset exhibits imbalanced classes and varying text lengths, making the analysis similar to real-world applications. This section of the thesis details the several steps implemented in the text preprocessing phase of the analysis.

In a nutshell, different text preprocessing methods are implemented on the dataset in order to obtain better data for semantic analysis and topic detection applications. Text observations are tokenized, punctuation marks and possessive apostrophes are removed, words are converted to lowercase, stopwords are removed from the text groups, lemmatization for each word is applied, frequency of words within each text sentences and topic groups are calculated, and synsets and hypernyms for each words are also calculated for improving the analysis quality, respectively. In the last part of the data preprocessing chapter, construction of semantic graphs are also explained briefly, but it is explained more detailed in next chapters.

Several Python libraries are utilized in the text preprocessing phase to perform various tasks and enhance the quality of the text data. These libraries offer robust functionalities and tools suitable for different aspects of text preprocessing.

One of the most frequently used tools in the preprocessing phase is the Natural Language Toolkit (NLTK) [5] , a platform for constructing Python programs to work with human language data. According to the NLTK homepage, it offers 'easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning. [5]

The NLTK tool is used several times in the text preprocessing parts of the DBPedia dataset. First of all, NLTK is employed for the tokenization, `word_tokenize` function from nltk.tokenize is used to split the text into individual words or tokens. Punctuation marks, such as periods, commas, question marks, and parentheses, often do not contribute significantly to the semantic meaning of the text. Removing these punctuation marks can simplify the text and reduce noise, making it easier to focus on the essential words and their relationships.

Another tool which is used for the punctuation removal is Python's built-in Regular Expression (re) [8] module which provides powerful functionalities for pattern matching and text manipulation. After tokenization of the texts, the 're.compile()' function is utilized to create a pattern object that represents a set of characters to be matched and removed from the text data. After this, another function called re.sub() is used to substitute the matched characters with an empty string. It replaces the occurances of the pattern in each word of the tokenized data with an empty string, effectively removing the punctuation marks.

Also, re.escape() function is used to escape special characters, for instance double quotation, before performing the substitutions. After using Regular Expression(re) preprocessing tools, punctuation marks and possessive apostrophes are removed from the tokenized text.

After punctuation removal, all text words are converted to lowercase which is crucial to achieve consistency and avoid duplicate entries based on case differences. By converting text to lowercase, it is assured that words with the same spelling but different capitalization are treated as identical, eliminating any dicrepancies that may arise due to case sensitivity.

As a next step of text preprocessing, stop words removal is applied. Stop words are commonly occuring words in a language that often do not carry significant semantic meaning and can be safely removed from text during preprocessing. Examples of stop

words include ‘the’, ‘and’ ‘is’ and other frequently used words in every texts which do not contain any semantic meaning for the analysis. Removing stop words helps reduce noise and focus on the more words bearing topic information in the text. In order to remove stop words in every texts in the dataset, NLTK corpus library’s ‘stopwords’ module is utilized to obtain English stop words. NLTK provides a pre-defined list of stop words for various languages, including English. This list contains commonly occuring words that can be safely removed from text without affecting its overall meaning.

After removing the stopwords from the texts, lemmatization for each words is applied for transforming words to its base form. Lemmatization is the process of reducing words to their base form, known as lemmas. It helps establish the canonical representation of words, allowing for better analysis and interpretation of text data. NLTK stem module is utilized as a tool for lemmatization process in the project. WordNetLemmatizer is an algorithm provided by NLTK, specifically designed for English words.

Afterwards, frequency distribution analysis technique is used to examine the occurence and frequency of words in a corpus. It helps identify the most common words and provides insights into the overall characteristics of the text data. NLTK Probability modules FreqDist class is used for analyzing the frequency of words in each text.

Most common words information in each classes is critical for the topic detection and semantic analysis tasks since this common words and relation between these common words in each classes is used for extracting the similarities and possible topic names for obtained clusters.

Last but not least, semantic graph is constructed to connect the words as well as synsets and hypernyms of words are calculated with using NLTK corpus WordNet library. A semantic graph can be defined as a graphical representation employed to visually portray the intricate web of semantic relationships and connections that exist among individual words or concepts. In the context of our study, this tool plays a fundamental role in revealing the hierarchical structure and interconnected nature of words. It aids in facilitating an exploration of semantic similarity and relatedness, thereby contributing to a deeper comprehension of topic detection within word groups.

Construction of a semantic graph involves utilizing WordNet, a lexical database in NLTK that provides a vast collection of synsets, which are sets of synonyms that share a common meaning. Synsets serve as the building blocks or establishing connections in the

semantic graph.

For example, when the word ‘cat’ is considered in wordnet, the sysent for ‘cat’ encompasses different senses or meanings such as ‘feline mammal’ and ‘gossip’. Each synset is assigned a unique identifier, and within WordNet, sysets are organized in a hierarchical structure. Hypernyms play a crucial role in representing the hiearchical relationships within WordNet. A hypernym is a word that represents a more general category or concept. For instance, the hypernym of ‘cat’ would be ‘mammal’, which is a broader category encompassign various species, including cats.

Table 1 shows 3 synsets of the word ‘cat’ and hypernyms of these synsets with definitions. Analysing texts with word synsets and hypernyms allows to obtain more comprehensive connections between the words in the texts and connections between these words critical for the semantic analysis in natural language processing tasks.

Table 1: Synsets and hypernyms explanation table

Word	**Definition**
cat.n.01	Feline mammal usually having thick soft fur and no ability to roar: domestic cats; wildcats
Hypernyms	Feline.n.01 – Any of various lithe-bodied roundheaded fissiped mammals, many with retractile claws
cat.n.03	A spiteful woman gossip
Hypernyms	Gossip n.03 – A person given to gossiping and divulging personal information about others woman.n.01 – An adult female person (as opposed to a man)
cat.v.01	Beat with cat-o’-nine-tails
Hypernyms	Flog.v.01 – Beat severely with a whip or rod

In linguistic notation, ’v(verb)’ represents the grammatical category of a word that typically describes an action or occurrence, as exemplified in Table 1 under the ’cat.v.01’ entry. Conversely, ’n(noun)’ denotes a grammatical category for words that signify people, places, things, or ideas, as demonstrated by the synsets ’cat.n.01’ and ’cat.n.03’ in the same table.

3.2 Sentence Embedding

In natural language processing (NLP), sentence embedding refers to the process of converting sentences into compact numerical representations. Models like mpnet-base-v2 [1] use advanced techniques to capture the meaning and context of sentences in fixed-length vectors. These representations encode the semantics and context of sentences, allowing for efficient analysis and comparison.

Pretrained models in NLP

Pretrained models in Natural Language Processing (NLP) refer to models that have been trained on large-scale unlabeled text data to learn general language representations. These models are typically trained using unsupervised learning methods, where they learn to capture contextual information, semantic representations, and language patterns from the vast amount of available text [13] [14].

Pretrained models are trained on massive text corpora, allowing them to capture intricate contextual information and semantic representations. The pretraining process typically involves training the models on extensive amounts of unlabeled data, enabling them to acquire comprehensive language knowledge. Common pretraining objectives include predicting masked words in a sentence or predicting the next sentence in a sequence. Subsequently, these pretrained models are fine-tuned on task-specific datasets with labeled examples, resulting in state-of-the-art performance across a range of Natural Language Processing (NLP) tasks [13] [15].

Fine-tuning takes place by further training the models using task-specific labeled data. This process enables the models to adapt their learned representations to handle specific tasks, such as sentiment analysis, text classification, question answering, and more.e. Fine-tuning enhances the models' performance on these specific tasks by leveraging the knowledge gained during pretraining [13] [16]. As part of this thesis project, a specific pretrained model named BERT is employed for the purpose of topic detection and semantic analysis.

Pretrained models, such as BERT (Bidirectional Encoder Representations from Transformers), utilize the transformer architecture as the underlying framework. These pretrained models leverage the transformer architecture to learn contextualized word representations by training on large-scale unlabeled text data [15].

Transformers

The transformer architecture is commonly employed in pretrained models for Natural Language Processing (NLP). It was introduced in the paper titled "Attention Is All You Need" in 2017 by researchers at Google [17]. The authors proposed the transformer architecture as an alternative to traditional recurrent neural networks (RNNs) for capturing dependencies and relationships in sequential data. Unlike RNNs, the transformer does not rely on sequential processing, making it highly parallelizable and capable of capturing long-range dependencies effectively [18] [19].

Attention Mechanism

At its core, the transformer incorporates a self attention mechanism, which allows the model to assess the importance of different words in a sentence and capture their relationships more efficiently. The self-attention mechanism in the transformer enables the model to dynamically focus on and consider various positions within the input sequence. It constructs context-aware representations by assigning attention scores to each word token based on its relevance to other words in the sentence. This process captures the intricate relationships and dependencies between words. This dynamic and adaptive weighting of words allows the model to capture both local and global dependencies, enhancing its anthropomorphising of the text [17, 20].

The self-attention mechanism achieves these goals through a process of calculating attention scores for each word token in parallel. For each word token, attention scores are computed with respect to all other word tokens in the sequence. These scores reflect the importance of other tokens in relation to the current token. The calculation of attention scores is guided by learned weights. These learned weights enable the model to focus on specific words that contribute most to the context of the current token. By combining the weighted representations of all word tokens, the model creates contextually rich embeddings that capture the nuances of semantic relationships within the sequence. This transformative approach to attention significantly enhances the model's ability to understand and represent complex linguistic structures [17, 20].

For instance, consider the sentence: "A dog ate the food because it was hungry." The self-attention mechanism helps the model understand that the pronoun "it" refers to the dog and not the food by relating the word "it" to all the words in the sentence. By capturing the relationships between words, the model can make accurate interpretations based on the given context [17, 20].

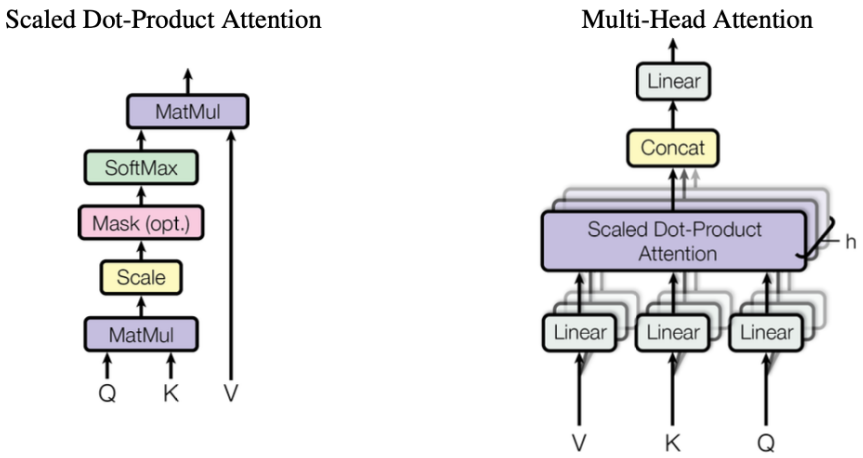


Figure 4: Scaled dot-product attention and Multi-head attention [17]

The attention mechanism is a crucial component of transformers that allows the model to capture the relationships and dependencies between words in a sentence. BERT employs a self-attention mechanism, specifically the scaled dot-product attention, which calculates the attention weights for each word by considering its relevance to other words in the sentence. This attention mechanism allows BERT to assign different weights to different words based on their contextual importance and captures the contextual relationships between words effectively[17].

Scaled dot-product attention

Scaled dot-product attention is a key component of the attention mechanism used in BERT. In the left part of the Figure 4, scaled dot-product attention mechanism is shown step by step. In the scaled dot-product attention mechanism, three key components are involved: the query, the key, and the value. These components are obtained by projecting the input embeddings into lower-dimensional subspaces using learnable linear transformations. Query represents the word for which attention weights are calculated where key and value represent other words in the sentence[17].
The scaled dot-product attention mechanism in BERT consists of four key steps explained in Table 2.

Multi-head attention mechanism

Table 2: Explanation of Scaled Dot Product Attention Mechanism in BERT

Step	**Description**
Step 1: Calculation of Dot Product	In this step, the dot product is calculated between the query vector (Q) and the key vector (K^T). The dot product measures the similarity between the vectors, indicating how related they are to each other.
Step 2: Scaling	To ensure stable gradients during the training process, the dot product matrix ($Q.K^T$) is divided by the square root of the dimension of the key vector. This scaling helps to prevent the values from becoming too large or too small, ensuring effective learning.
Step 3: Normalization with Softmax	The preceding similarity scores are then normalized using the softmax function. Applying the softmax function to the scores brings them into the range of 0 and 1, and the sum of the scores becomes 1. This normalization process helps to obtain attention weights that reflect the relative importance of each word.
Step 4: Computation of Attention Matrix	In the final step, the attention matrix (Z) is computed by multiplying the softmax-normalized score matrix (softmax(QK^T/d_k)) with the value matrix (V). The attention matrix contains the attention values for each word in the sentence. It is obtained by summing the values of the vectors, weighted by the attention scores.
Attention Formula	$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$

BERT employs a multi-head attention mechanism to enhance the computation of the attention matrix. Unlike a single attention head, this approach calculates multiple attention matrices simultaneously. Each attention head focuses on capturing distinct aspects and nuances of the input data. In the subsequent normalization layer, attention values are normalized within each head, ensuring consistent patterns within that head. This normalization does not enforce uniformity across all heads; rather, it guarantees reliable attention patterns within each head.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h) \cdot W_O$$
$$\text{where} \quad W_O \in \mathbb{R}^{h \times d_{\text{model}}}$$

The output of each attention head, denoted as $\text{head}_1, ..., \text{head}h$, is concatenated and further transformed by the weight matrix $W_O \in \mathbb{R}^{h \times d\text{model}}$. Following this, BERT integrates a normalizing layer to maintain consistent patterns within each head, and a subsequent feed-forward layer for additional processing and transformation [17]. The multi-head attention mechanism is depicted in Figure 4 on the right.

The utilization of multiple attention heads provides a distinct advantage over a single head, as it enriches the accuracy and depth of the attention matrix. By focusing on different facets of the input, each attention head captures diverse contextual representations, incorporating various perspectives and encompassing both local and global dependencies. This multifaceted methodology enhances BERT's ability to discern intricate interrelationships among words, resulting in thorough and finely nuanced contextualized representations [17].

Transformers Architecture

The transformers architecture consists of an encoder and decoder structure. The encoder processes the input sequence and generates a context representation $(x_1, ..., x_n)$, which is further transformed into a sequence of continuous representations $z = (z_1, ..., z_n)$.

Given the context representation z, the decoder takes it as input and generates the output sequence $(y_1, ..., y_m)$ of symbols one element at a time. The model operates in an auto-regressive manner, meaning that it consumes the previously generated symbols as additional input when generating the next symbol in the sequence.

In summary, the encoder-decoder architecture of transformers allows the model to process the input sequence and create a meaningful context representation, which is then

used by the decoder to generate the output sequence step by step, taking into account the previously generated symbols for auto-regressive generation [19][17].

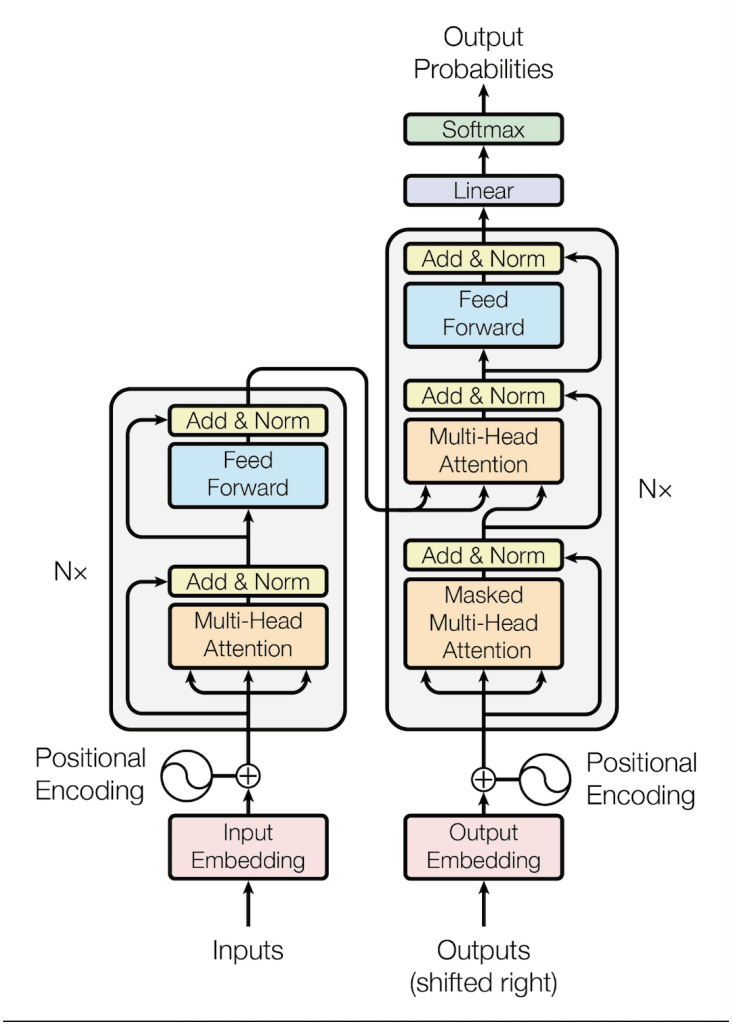


Figure 5: Transformers Architecture [17]

In the Figure 5 from 'All you need is an attention' [17] paper, transformers architecture is visualized.

The transformers architecture is structured around an encoder and decoder framework. The encoder part of the architecture consists of several key components. In the beginning of the process, the input sequence is transformed into sequence of word embeddings. These embeddings capture the semantic information of each word in the input sequence. Then, input embeddings are passed through multiple parallel self-attention layers in the encoder. Each self-attention mechanism allows each word in the sequence to attend the other words,

capturing the relationships and dependencies between words at different positions [17] [21].

After the self attention layer, residual connections are added. This helps the model to learn both the input embeddings and the attention outputs. The outputs are then normalized to stabilize the learning process.

After this, outputs are passed through a feed-forward neural network. This network includes multiple layers of linear transformations and non linear activation functions which allows the model to learn complex representations of the input sequence. Last but not least, residual connections and layer normalization are again applied to the output of the feed-forward layer [17] [21].

The decoder takes the outputs from the encoder as the initial hidden states for the decoder. The decoder incorporates positional embeddings to consider the order of words in the output sequence. Additionally, the decoder employs masked and multihead attention mechanisms, allowing it to focus on previously generated words while simultaneously averting attention towards upcoming words. Residual connections and layer normalization are applied throughout the decoder layers [17] [21].

The decoder also employs multihead attention to capture relevant information from the encoder outputs. A feed-forward neural network is applied to the normalized outputs. Finally, a linear transformation followed by softmax activation generates output probabilities for each word in the vocabulary as the next word in the output sequence [17] [21].

The BERT model utilizes the bi-encoder architecture for embedding sentences. As a result, the encoder part of this architecture is particularly relevant for this project, as it plays a crucial role in capturing the contextual information and generating sentence embeddings.

BERT

The BERT (Bidirectional Encoder Representations from Transformers) model employs the encoder architecture of the transformer model in a bidirectional manner. This approach allows BERT to consider not only the context to the left but also the context to the right of each word during training. In other words, BERT comprehends the entire sentence from both directions, gathering a holistic understanding of the relationships between words.

As explained in the attention mechanism section, BERT adopts the encoder archi-

tecture of the transformer model in a bidirectional fashion. This trait helps BERT to analyze the sentence in both forward and backward directions. As a result, the model tries to capture the intricate interdependencies among words, encompassing the contextual nuances from all angles. Unlike traditional word embedding methods like Word2Vec, which consider words in isolation, BERT takes into account the contextual information of each word by leveraging the power of the transformer model. Context-based embedding, as implemented in BERT, goes beyond traditional word embeddings by considering the full context in which words appear. This allows BERT to capture complex linguistic patterns and semantic relationships.

For example, to understand the differences between context-based and context-free embedding models, consider the following sentences:

- Sentence A: "She saw a bat flying in the sky."
- Sentence B: "He swung the bat and hit the ball."

In Sentence A, the word "bat" refers to a flying mammal. However, in Sentence B, the word "bat" refers to a sports equipment used in baseball. The difference in meaning is clear based on the context of each sentence. A context-based model like BERT would consider the context of the entire sentence to generate word embeddings that capture the different meanings of "bat" based on the surrounding words. This enables the model to differentiate between the two meanings and better represent the intended meaning of the word in each context. On the other hand, context-free models do not perform well in differentiating the meaning of a word in different contexts. These models assign a fixed embedding to each word regardless of its context, treating the word as if it has the same meaning in all instances.

The transformer's self-attention mechanism plays a crucial role in BERT's context-based embeddings. BERT leverages the self-attention mechanism of the transformer architecture to differentiate words based on their contexts. Self-attention allows BERT to capture the dependencies and relationships between words in a sentence. By attending to the entire input sequence, BERT can assign different weights to different words based on their relevance to each other [20] [17].

In this thesis project, two different tokenization processes are explained. It is important to note that the tokenization used in the text preprocessing stage is independent of the tokenization employed in the BERT model. These two processes serve distinct purposes and are not directly related.

The tokenization used in the BERT model is specifically designed for the purpose of encoding sentences and generating sentence embeddings. On the other hand, the tokenization performed in the preprocessing stage serves different objectives. It is employed for tasks such as creating semantic graphs and analyzing word frequency for topic detection analysis. This preprocessing tokenization is distinct from the BERT model's tokenization and serves specific purposes in understanding the text and extracting relevant information.

BERT uses the WordPiece tokenizer, which follows a subword tokenization scheme. The WordPiece tokenizer reduces vocabulary size and improves results for rare words by splitting them into subwords. When processing a given text, the tokenizer first checks if a word is directly present within its WordPiece vocabulary. If the word exists in the vocabulary, it is treated as a standalone token. However, if the word is absent from the vocabulary, it is systematically broken down into subwords. [22] [17]
This breakdown process involves attempting to identify the longest subword within the input word that is included in the vocabulary. This identified subword is then isolated as a token. Finally, the remaining portion of the original word is subjected to the same procedure. This iterative process continues until the entire word is represented through a sequence of tokens.

One tokenized sentence is provided as an example for clarification.

Tokens = [let, us, start, pre, ##train, ##ing, the, model]

In this example, the word "pretraining" is not in BERT's vocabulary, thus this word is split into subwords. If the word "pretraining" is not part of BERT's vocabulary, the tokenizer will strive to identify the longest subsequences within it that are included in the vocabulary. In this case, it might identify "pre," "train," and "ing" as subwords that exist in the vocabulary. Therefore, "pretraining" would be fragmented into these subwords, and these subwords would be treated as individual tokens, as demonstrated in the example.

Transformer Building Blocks

The three fundamental building blocks of a transformer encoder can be described as follows: (1) Input representation, (2) Attention mechanisms, and (3) Pre-training [17]

The input representation phase involves transforming the input text into a numeri-

cal representation that captures its sequential nature. In the Figure 6 input representation is explained visually. The input representation includes three embedding layers: Token embedding, Segment embedding, and position embedding. In the case of BERT, the input consists of two sentences. The [CLS] token is added only at the beginning of the first sentence. The [SEP] token is used to separate the sentences and indicate the end of each sentence. [17] [21]

Before inputting the tokens into BERT, they are converted into embeddings using an embedding layer called token embeddings. It is important to note that the values of token embeddings are learned during the training process. [17] [21] [13]

Additionally, the input representation includes segment embeddings and positional embeddings. The segment embedding is used to distinguish between the two given sentences, providing information about whether the tokens belong to the first sentence or the second sentence. The positional embedding layer is used to provide information about the position of the tokens since the transformer does not use any recurrence mechanism and processes all the words in parallel. [20] [17] [13]

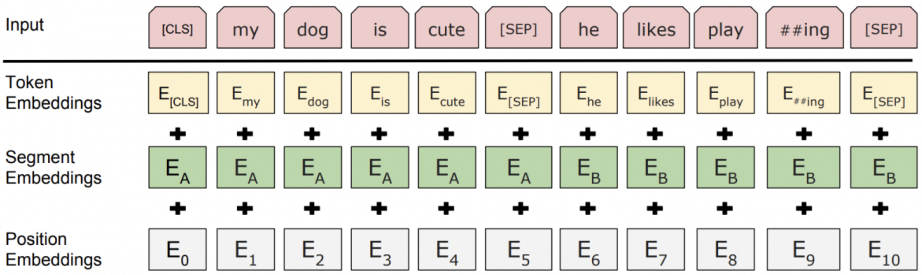


Figure 6: Input Representation [13]

Two main pretraining tasks used for the BERT model are Masked Language model(MLM) and Next Sentence Predictions (NSP). In the masked language modelling task, a certain percentage of the input tokens are randomly masked with a [MASK] token. The model's objective is to predict the original masked tokens based on the surrounding context. By learning to fill in the masked tokens, BERT gains an understanding of the relationships between words and their context. This MLM task helps the model to capture syntatic and semantic information. [17] [13]

In next sentence prediction task, BERT is trained to predict whether two given sentences appear consecutively or if they are randomly sampled from different documents. This

task enables the model to grasp the relationship between sentences and learn to capture sentence-level semantics. The NSP task helps BERT understand the coherence and connectivity between sentences, allowing it to generate meaningful representations for text pairs. [13]

In Figure 7, the pre-training procedure is explained, where a sentence's WordPiece token is denoted by T_i. Random tokens are masked for both Sentence A and Sentence B, and vector C is used for Next Sentence Prediction.

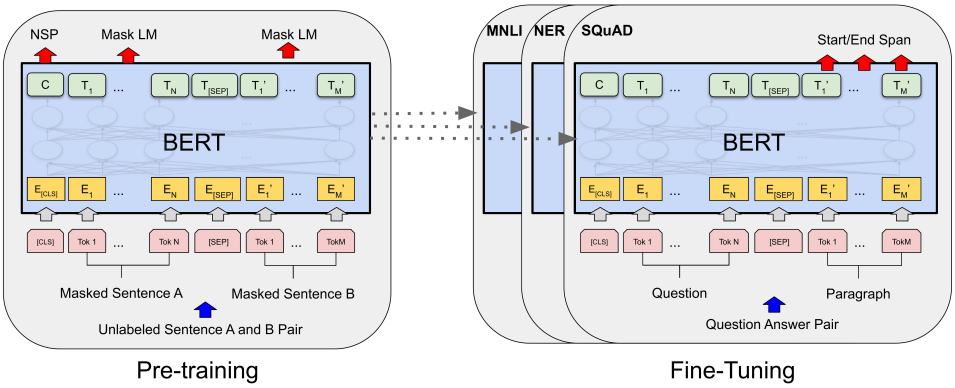


Figure 7: Pre-training of BERT [13]

Implementation Using the MPNet-Base-v2 Model

A sentence transformers model called 'all-mpnet-base-v2' ([1]) is used for text embeddings for the DBPedia dataset.
The MPNet-Base-v2 model is built upon the BERT model, incorporating its underlying principles and structures. It is a variant of the BERT model designed specifically for sentence-level tasks, leveraging the transformer architecture to learn powerful sentence embeddings. Unlike BERT, which primarily focuses on monolingual data, MPNet-Base-v2 extends this approach by incorporating multilingual training. It is pretrained on a diverse multilingual corpus, thereby capturing cross-lingual semantic information and providing robust language representation capabilities across multiple languages.

In the initial stage of this project, we generate sentence embeddings using the MPNet-Base-v2 model. These embeddings, represented as 768-dimensional vectors, are prepared separately for three distinct datasets: training, validation, and testing.

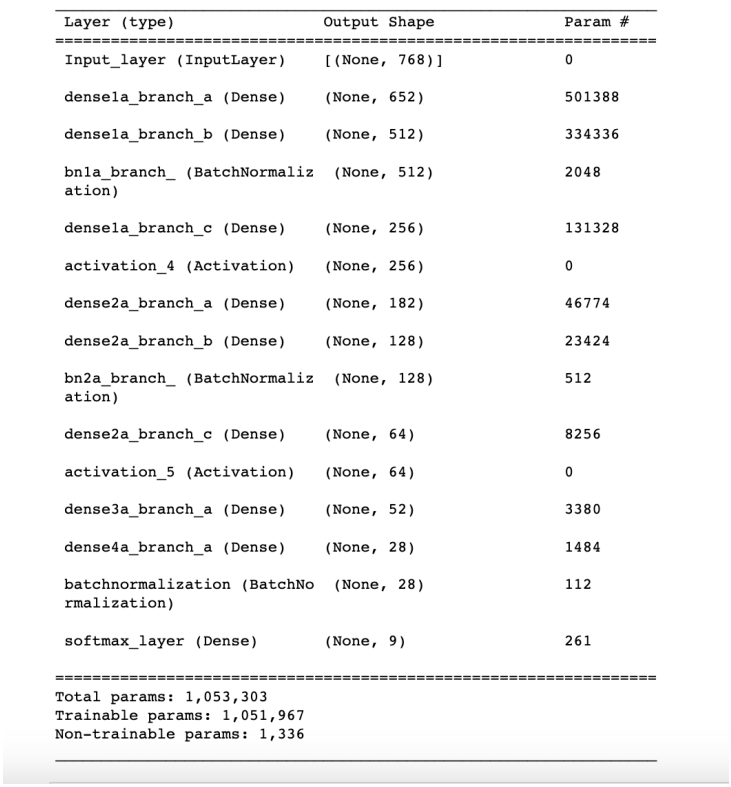
Subsequently, the labels for each sentence are prepared using a label encoding method, which converts categorical labels into a numerical form suitable for machine learning models. The processed embeddings and label information are then integrated into a

pandas DataFrame for each of the datasets. Each DataFrame comprises the sentence embeddings, original text, and hierarchical labels (l1, l2, l3).

Neural Network Architecture for Embeddings

After having sentence embeddings for each text observation with the 'MPNet-Base-v2' sentence transformer pretrained model, a neural network architecture comes from Pandya's master's thesis [22] is defined and implemented on the embeddings to decrease the dimensionality and making the embeddings 'topic aware'. The network is a type of feedforward neural network, with multiple dense layers, including batch normalization layers for stable learning and activation layers to introduce non-linearity. As shown in Figure 8, each dense layer serves a specific function. Some layers aim to reduce the dimensionality of the input data, others introduce non-linearities via activation functions, and batch normalization layers are included to stabilize the learning process and reduce internal covariate shift.

This neural network model is trained using the processed training dataset, with the validation dataset employed to tune the model parameters and prevent overfitting. However, the predictions from this model are not directly used. Instead, the output from the final batch normalization layer is used as new, lower-dimensional embeddings for the text data. This modified model is employed to transform the original 768-dimensional embeddings into new, 28-dimensional embeddings. These new embeddings are designed to be more representative and to capture the topic aware features of the text data in a lower-dimensional space more efficiently. The new embeddings, generated for all three datasets, are then used for topic detection and semantic analysis approaches as explained in the following sections of the thesis.



Layer (type)	Output Shape	Param #
Input_layer (InputLayer)	[(None, 768)]	0
dense1a_branch_a (Dense)	(None, 652)	501388
dense1a_branch_b (Dense)	(None, 512)	334336
bn1a_branch_ (BatchNormalization)	(None, 512)	2048
dense1a_branch_c (Dense)	(None, 256)	131328
activation_4 (Activation)	(None, 256)	0
dense2a_branch_a (Dense)	(None, 182)	46774
dense2a_branch_b (Dense)	(None, 128)	23424
bn2a_branch_ (BatchNormalization)	(None, 128)	512
dense2a_branch_c (Dense)	(None, 64)	8256
activation_5 (Activation)	(None, 64)	0
dense3a_branch_a (Dense)	(None, 52)	3380
dense4a_branch_a (Dense)	(None, 28)	1484
batchnormalization (BatchNormalization)	(None, 28)	112
softmax_layer (Dense)	(None, 9)	261

Total params: 1,053,303
Trainable params: 1,051,967
Non-trainable params: 1,336

Figure 8: Neural Network architecture for embeddings

3.3 Clustering methods and dimensionality reduction

Unsupervised learning, a distinct class of machine learning, is employed in this study as a pivotal analytical approach, enabling the uncovering of inherent patterns within the datasets. This category of machine learning methods leverages algorithms to identify patterns in datasets containing data points that are neither classified nor labeled. The goal is to model the underlying structure or distribution of the data, gaining more understanding about it without any guidance in the form of target outputs. Two fundamental areas of unsupervised learning, namely clustering and dimensionality reduction, are integral to this project. [23]

Clustering involves dividing the dataset into groups, or clusters, of similar instances. This division can unveil insights about the dataset by highlighting hidden patterns that might not be immediately noticeable. Following the generation of new sentence embeddings from the batch normalization layer of the neural network in this project, the K-Means and HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) clustering algorithms were specifically deployed to analyze and model the data.[23]

Dimensionality reduction is a technique used to decrease data complexity while preserving as much crucial information as possible. This is achieved by mapping high-dimensional data to a lower-dimensional space, a process that can enhance computational efficiency, reduce storage space requirements, and mitigate the 'curse of dimensionality', which can adversely affect the performance of machine learning models. Within the scope of this research, PCA was utilized post-clustering as a tool for data visualization. It aided in simplifying the visualization of clusters by transforming the high-dimensional sentence embeddings into a 2-dimensional space.

It is noteworthy that the evaluation of the results produced by unsupervised learning methods can be complex due to the absence of universally accepted mechanisms for result validation. This differs from supervised learning where prediction performance on unseen data can be used to validate the efficacy of the model. Nonetheless, unsupervised learning, often utilized in exploratory data analysis, remains instrumental in revealing concealed data patterns and structures.

Having outlined the general ideas and challenges of unsupervised learning, clustering, and dimensionality reduction, the next parts of this thesis will provide a detailed exploration of PCA, K-Means, and HDBSCAN. [23]

3.3.1 Principal Component Analysis (PCA)

PCA, a fundamental tool in unsupervised learning, traditionally serves two main purposes: to generate derived variables for supervised learning problems, and to facilitate data visualization.

The key objective of PCA is to derive a low-dimensional representation of a dataset that captures as much variation as possible. Although each observation exists in a p-dimensional space, not all these dimensions are equivalently interesting. PCA seeks dimensions (principal components) that are as informative as possible, measured by the amount that observations vary along each dimension. These dimensions are linear combinations of the p features.[23]

Consider a dataset with p features. Let $X_1, X_2, ..., X_p$ represent the values of these features for a given observation (i.e., they are scalars). The principal component analysis attempts to find a linear combination of these feature values to create new dimensions called principal components.

The first principal component of a set of features $X_1, X_2, ..., X_p$ is the normalized linear combination of the features that has the largest variance. The constraint on these loadings is that their sum of squares equals one; otherwise, arbitrarily large values could lead to an arbitrarily large variance.[23]

$$Z_1 = \phi_{11}X_1 + \phi_{21}X_2 + ... + \phi_{p1}X_p \quad (1)$$

For an $n \times p$ data set X, each variable in X is centered to have a mean of zero (meaning the column means of X are zero). This is a preparatory step to compute the first principal component. The objective is to find the linear combination of the sample feature values that has the largest variance, subject to the constraint that the sum of squares of loadings equals one. This optimization problem can be represented by:

$$z_{i1} = \phi_{11}x_{i1} + \phi_{21}x_{i2} + ... + \phi_{p1}x_{ip} \quad (2)$$

$$\underset{\phi_{11},...,\phi_{p1}}{\text{maximize}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{p} \phi_{j1}x_{ij} \right)^2 \right\} \text{ subject to } \sum_{j=1}^{p} \phi_{j1}^2 = 1. \quad (3)$$

From (2), the objective in (3) can be written as $\frac{1}{n}\sum_{i=1}^{n} z_{i1}^2$. Since $\frac{1}{n}\sum_{i=1}^{n} x_{ij} = 0$, the average of the $z_{11}, ..., z_{n1}$ will be zero as well. Hence, the objective that we are maximizing in (3) is just the sample variance of the n values of z_{i1}. We refer to $z_{11}, ..., z_{n1}$ as the scores of the first principal component. Problem (3) can be solved via an eigen decomposition. [23]

Once the first principal component (Z_1) has been determined, the second principal component (Z_2) can be computed. The second principal component is the linear combination of $X_1, ..., X_p$ that has the maximum variance out of all linear combinations uncorrelated with Z_1. The scores of the second principal component ($z_{12}, z_{22}, ..., z_{n2}$) are derived from the formula:

$$z_{i2} = \phi_{12}x_{i1} + \phi_{22}x_{i2} + ... + \phi_{p2}x_{ip} \quad (4)$$

where ϕ_2 is the second principal component loading vector, with elements $\phi_{12}, \phi_{22}, ..., \phi_{p2}$. Constraining Z_2 to be uncorrelated with Z_1 is the same as ensuring the direction ϕ_2 is

orthogonal (perpendicular) to the direction ϕ_1. The problem similar to the one solved for ϕ_1 is solved for ϕ_2, with the additional requirement that ϕ_2 is orthogonal to ϕ_1. [23]
In the usual process, after computing the principal components, they can be plotted against each other to provide a low-dimensional view of the data, facilitating easier interpretation and visualization. However, in this study, PCA is applied differently, as it's used post-clustering.[23]

Z_2 is the linear combination of features that has maximal variance among all linear combinations that are uncorrelated with Z_1. The constraint that Z_2 be uncorrelated with Z_1 equates to ϕ_2 being orthogonal to ϕ_1.

Note: the principal component directions $\phi_1, \phi_2, \phi_3, ...$ are the ordered sequence of eigenvectors of the matrix $X^T X$, and the variances of the components are the eigenvalues. There are at most $\min(n-1, p)$ principal components.[23]

After clustering high-dimensional sentence embeddings using K-Means and HDBSCAN algorithms, PCA was used to transform these embeddings into a 2-dimensional space for visualization purposes.

This approach facilitated a more straightforward interpretation of the underlying data structure and aided in visually inspecting the clustering results. The resulting plots offered a clear depiction of the data partitioning performed by the clustering algorithms, thus illustrating the effectiveness of these methods in delineating distinct clusters within the data.

In this research, PCA was used after the application of clustering, offering a different way to interpret complex data. This method has helped in better understanding the results of the used clustering methods, namely K-Means and HDBSCAN.

Clustering Methods

In previous sections, sentence embeddings for text data are calculated using the M' Net-Base-v2' pretrained model. a neural network architecture is used to reduce the dimensions from 768 to 28. After these steps, this section introduces two popular clustering algorithms: K-Means and Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN). Additionally, the silhouette score, which is used to validate cluster quality and compare the results of these two clustering models, is also discussed

3.3.2 K-Means Clustering

K-means clustering is also called a partitioning algorithm. The main goal of partitioning algorithms is to construct a partition of a database D of n objects into a set of k clusters, minimizing an objective function.

K-means clustering is a straightforward yet powerful method for segregating a dataset into K clusters. The process commences with the predetermination of the number of clusters, K, following which the K-means algorithm assigns each data point to one of these K clusters. This leads to the creation of cluster sets, represented as $C_1, C_2, \ldots, C_K$, which encompass the indices of the observations within each cluster. Two conditions govern these sets:

1. Each observation is a member of one of the K clusters ($C_1 \cup C_2 \cup \ldots \cup C_K = \{1, \ldots, n\}$).

2. The clusters do not intersect, meaning an observation cannot belong to multiple clusters ($C_k \cap C_{k'} = \emptyset$ for all $k \neq k'$).

The primary objective of K-means clustering is to achieve the smallest possible within-cluster variation. Observations have to be partitioned into K clusters such that the total within-cluster variation, summed over all K clusters, is as small as possible.

$W(C_k)$ is a measure of within-cluster variation.To find the best cluster assignments, minimizing the overall within-cluster variation across all clusters is aimed:

$$\underset{C_1,\ldots,C_K}{\text{minimize}} \left\{ \sum_{k=1}^{K} W(C_k) \right\}$$

Here, equation calculates the average squared Euclidean distance between all pairs of observations in the same cluster. In order to calculate this within-cluster variation, $W(C_k)$, squared Euclidean distance is used. Essentially, for a given cluster, we measure how "spread out" the data points are within that cluster. The mathematical representation of this concept is:

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2$$

where $|C_k|$ denotes the number of observations in the kth cluster.

Thus, the within-cluster variation for the kth cluster is calculated as the sum of the

squared Euclidean distances between all observation pairs within the same kth cluster, normalized by the total number of observations present in that cluster.
With this definition of within-cluster variation, optimization problem can be expressed as:

$$\underset{C_1,\ldots,C_K}{\text{minimize}} \left\{ \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 \right\}$$

In simpler terms, the K-means algorithm tries to find clusters such that the average distance between data points in the same cluster is minimized.

The algorithm for solving this equation, which partitions the observations into K clusters to meet the minimization objective, is explained below. The primary goal of this algorithm is to provide a locally optimal solution for the K-Means optimization problem.

The K-Means algorithm's dependence on initial centroid placements can lead it to converge to a local minimum instead of a global one. Running the algorithm multiple times with different initializations may increase chances of finding a better solution, but a global minimum is never guaranteed due to the nature of the clustering problem.

Table 3: K-Means Clustering Algorithm

Step	**Description**
1	Randomly initialize k centroids by selecting k data points from the dataset.
2	Compute the centroids of the clusters based on the current partition. The centroid is the arithmetic mean of all the points in the cluster, serving as the cluster's center of gravity.
3	Assign each data point to the cluster whose centroid is nearest, based on Euclidean distance or other distance metrics.
4	Update the centroids based on the newly assigned data points and repeat Step 2. Terminate when centroids no longer change between iterations.

The K-means algorithm ensures a consistent reduction in the within-cluster sum of squares. In Step 2, by recalculating the cluster centroids, we ensure the least sum of squared deviations for each feature. During Step 3, reassigning observations based on

these updated centroids can only reduce the overall distance to centroids. Thus, the value of the objective function never increases between iterations. As a result, the algorithm concludes when it reaches a local minimum for the objective function.

Since the K-means algorithm finds a local rather than a global optimum, the algorithm may need to run multiple times from different random initial configurations.

Implementing K-Means clustering is easy and relatively efficient. However, there are some drawbacks too. It is applicable only when the mean is defined, and the number of clusters (k) needs to be specified in advance. Also, the K-Means algorithm is sensitive to outliers, and clusters are forced to have convex shapes.

3.3.3 HDBSCAN

HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) is an extension of the DBSCAN algorithm that introduces a hierarchy into the clustering process. It offers a more flexible and granular approach to identifying clusters by leveraging the concept of density.

Density in clustering refers to the concentration of data points within a specific region. Dense regions exhibit a high concentration of points, indicating strong intra-cluster similarity, while sparse regions have a lower density, suggesting areas with fewer data points and weaker inter-cluster relationships.

The HDBSCAN algorithm builds upon the foundation of DBSCAN, a density-based clustering algorithm. Unlike traditional clustering methods such as K-Means, DBSCAN does not require the pre-specification of the number of clusters and can discover clusters of arbitrary shapes.

The core idea of DBSCAN is density reachability and density connectivity. Starting from an arbitrary point, if there are at least a minimum number of points ('MinPts') within a given radius ('eps'), that point becomes the core point of a new cluster. DBSCAN then iteratively adds all directly reachable points within the 'eps' distance to this cluster. The process continues until no new points can be added, forming clusters based on density. Points that do not belong to any cluster are considered noise or outliers.

HDBSCAN takes this density-based clustering concept further by introducing a hierarchical clustering approach. It constructs a hierarchy of clusters by considering

different density levels and progressively expanding clusters from regions of higher density to lower density areas. This hierarchy captures the relationships between clusters at different levels of granularity.

The utilization of the hierarchy in HDBSCAN enables a more comprehensive exploration of the data structure. By analyzing the hierarchy, the density and sparseness of different regions can be analyzed within the dataset. The hierarchy allows for adjustable threshold values, providing flexibility in detecting clusters and subclusters at different density levels.

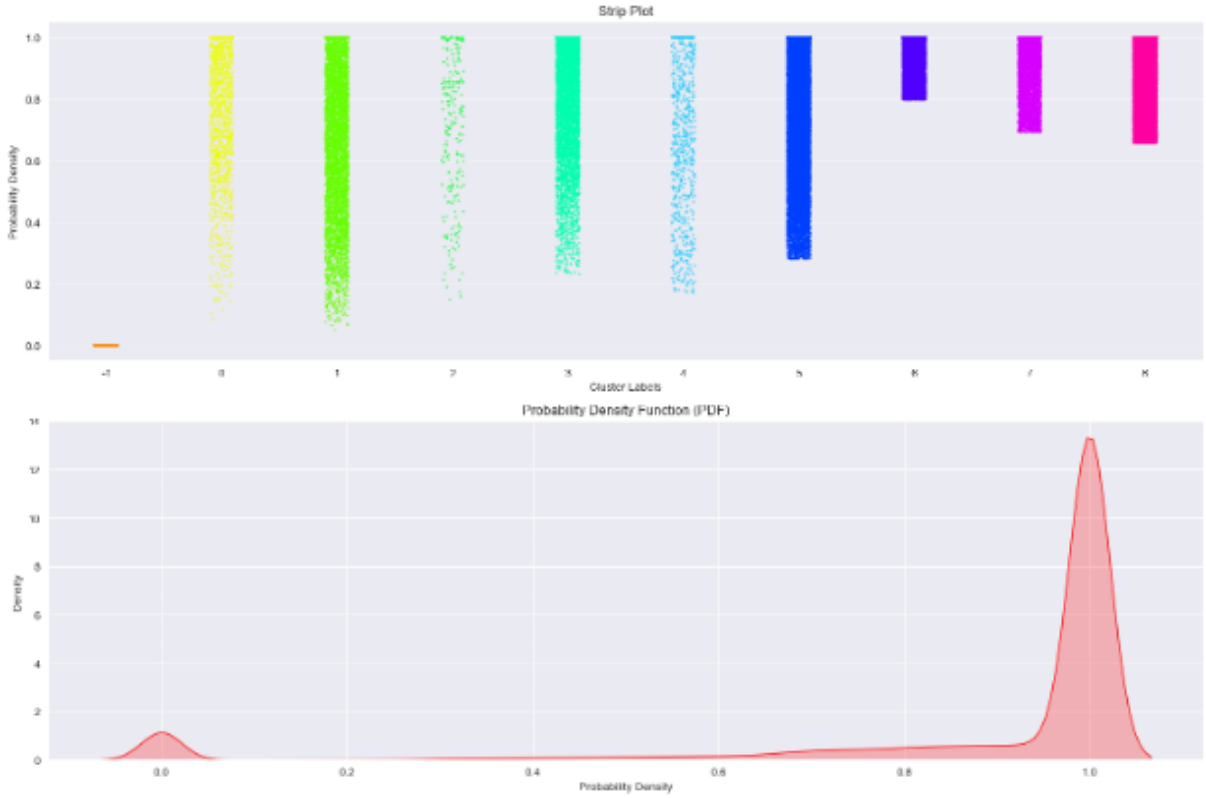


Figure 9: Probability Density Function for each classes in DBPedia

Figure 9 displays the probability density function (PDF) of the HDBSCAN algorithm applied to the DBPedia dataset, along with strip plots for each of the nine topic classes.

Calculating the PDF is a crucial part of density-based clustering algorithms like HDBSCAN. However, for large datasets like DBPedia, the computation of the PDF presents challenges due to its high dimensionality and the sheer number of data points. There are two notable methods for overcoming this challenge:

1.Fixed ϵ-radius Hypersphere Method: Here, the local density around a data point

is gauged by counting how many neighboring points lie within a predefined distance, termed the ϵ-radius. Denser regions have more points within this radius, while sparser ones have fewer. However, the method's effectiveness can vary based on the chosen radius and the dataset's size, making it sensitive to both scale and data magnitude [24] [23].

2. Variable ϵ-radius Method: This strategy doesn't use a set distance. Instead, it adjusts the ϵ-radius for each point based on the distance to its (K) th nearest neighbor, where (K) is a preset number. In denser areas, the radius needed to encompass (K) neighbors is smaller, signaling high density, whereas it's larger in less dense areas. HDBSCAN employs this approach, as it more fluidly adapts to the dataset's intrinsic densities [24] [23].

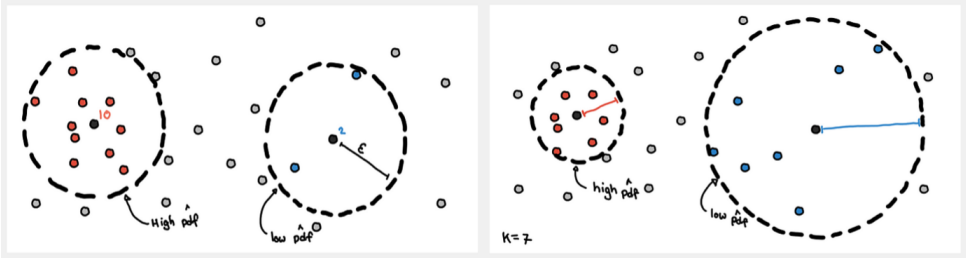


Figure 10: Calculating PDF methods [24]

In the Figure 10, the left visualization depicts the "Fixed ϵ-radius Hypersphere" approach, while the right visualization represents the "Variable ϵ-radius" approach.

After calculating the PDF, the next step involves extracting the hierarchical structure of the data.

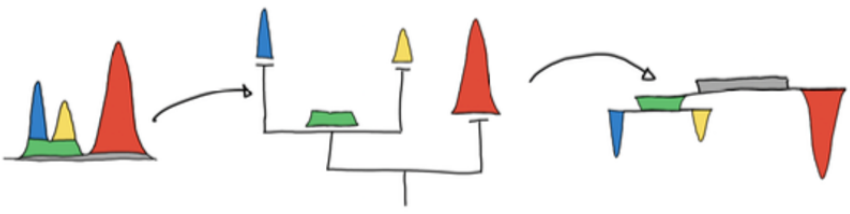


Figure 11: Hiearchical Structure of HDBSCAN [24]

Figure 11, shows the hiearchical structure of the data. To capture and represent these cluster relationships effectively, a hierarchy tree is employed. This tree structure offers a generalized representation that extends to higher dimensions. It serves as a natural

abstraction, facilitating traversal and manipulation of the data. Conventionally, hierarchy trees are depicted top-down, with the root node representing a single cluster at the top and subsequent nodes representing merged or connected clusters as the tree branches downward. [24] [23]

Excess of Mass (EoM)

The Excess of Mass (EoM) algorithm is employed for HDBSCAN modeling in this project to select the final set of clusters from the hierarchical representation produced by the algorithm. EoM plays a role in the cluster selection procedure within HDBSCAN, aiming to identify clusters based on their density distribution as they emerge within this representation.

The method starts by estimating the dataset's density distribution, creating a landscape that portrays the proximity and concentration of data points. A predetermined density threshold then serves as a criterion to highlight significant regions in this landscape. Those surpassing the threshold are identified as potential clusters due to their pronounced data point concentration.

Central to EoM is its emphasis on clusters demonstrating a significant "mass" of data points. By underscoring this density, the method prioritizes the most populated regions while potentially sidelining less dense areas.

In summary, the Excess of Mass method is a strategic technique utilizing density estimations to differentiate and select pertinent clusters from the hierarchical structures generated by HDBSCAN.

Hiearchy in Clustering and Dendogram

In Chapter 5 , an enhancement to the initial topic detection analysis is proposed by integrating hierarchical clustering with the HDBSCAN algorithm. This combined approach seeks to provide a deeper understanding of the data's structure by superimposing a hierarchical perspective onto the density-based clusters obtained from HDBSCAN. The addition of hierarchy offers the opportunity to delve into different granularities, uncovering the nuances of density variations within the dataset.

In the context of hierarchical clustering, Ward's method, also known as the minimum variance method, optimizes the clustering process by seeking to minimize the total within-cluster variance. Specifically, during the hierarchical merging process, the pair of clusters that results in the smallest increase in this variance when merged is prioritized for merging.

It's worth noting that while HDBSCAN incorporates a form of hierarchical clustering based on density, the traditional hierarchical clustering method presented in Chapter 5 emphasizes merging based on dissimilarity measures, such as the variance within clusters when using Ward's linkage.

Dendrograms serve as visual tools for representing the results of hierarchical clustering. These diagrams offer a lucid depiction of how data points or clusters are merged at each level of the hierarchy. Within a dendrogram, the height of vertical lines signifies the degree of dissimilarity between clusters or data points being merged. Longer vertical lines suggest a higher degree of dissimilarity, indicating that clusters or data points are quite distinct from each other.

Silhuette Score

Following the application of the KMeans and HDBSCAN algorithms for data clustering, an evaluation of the resultant clusters' quality becomes necessary. This assessment can be accomplished using the Silhouette score, a measurement that offers insights into the homogeneity of data within the clusters and the extent of separation between different clusters. [25] [26]

The Silhouette score calculation incorporates two important metrics: the mean intra-cluster distance $a(i)$ and the mean nearest-cluster distance $b(i)$.

The mean intra-cluster distance $a(i)$, captures the internal cohesion of the clusters. Specifically $a(i)$ is determined by computing the average distance between a single data sample (i) and all other data points in the same cluster. It's given by:

$$\text{intra-cluster distance, } a(i) = \frac{1}{|C_A| - 1} \sum_{j \in C_A, i \neq j} d(i, j)$$

Where, C_A represents the number of points in cluster A, and $d(i, j)$ is the distance between data points points i and j. A smaller $a(i)$ value suggests better cohesion within a cluster, indicate of high-quality clustering. [25]

The mean nearest-cluster distance, denoted as the $b(i)$ score, on the other hand, gauges the dissimilarity between a data sample and its nearest cluster to which it doesn't belong. It is calculated as the minimum average distance from the data sample (i) to samples in any other cluster. A larger $b(i)$ score suggests that the clusters are well-separated, implying a high-quality clustering. [25]

$$\text{mean dissimilarity, } b(i) = \min_{J \neq I} \frac{1}{C_J} \sum_{j \in C_J} d(i,j)$$

In this formula, C_J is the mean of the distance from i to all points in cluster C_J.
From these two equations, the silhouette value for data point i can be computed as:

$$\text{silhouette value, } s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, \text{ if } |C_A| > 1$$

where $-1 \leq s(i) \leq 1$.

It is worth noticing that this approach provides better results when data points within assigned clusters are following a standard normal distribution. [25]

The Silhouette score for each sample is then calculated as the difference between the mean nearest-cluster distance ($b(i)$) and the mean intra-cluster distance ($a(i)$), divided by the maximum of these two distances. The overall Silhouette score of a dataset is obtained by taking the average of the Silhouette scores for each sample.
This score provides a comprehensive measure of the quality of the clustering, with higher scores indicating well-separated and cohesive clusters. [25] [26]
[27] [28]

4 First Exploration and Baseline Analysis on Topic Detection

4.1 Clustering Methods

After implementing the neural network architecture to obtain embeddings, the topic detection task proceeds with the application of unsupervised clustering methods, namely HDBSCAN and KMeans. These clustering techniques aim to explore the underlying structure within the data and group similar embeddings together to identify distinct topics or patterns. The performance of these methods is evaluated using the silhouette score, which measures the quality of clustering results.

Firstly, the HDBSCAN algorithm is implemented on the obtained embeddings to perform unsupervised clustering for topic detection. The algorithm is instantiated with three key parameters that have been selected to achieve meaningful clustering results:

- **Minimum Cluster Size:** This parameter controls the minimum number of data points required to form a cluster. For this task, it is chosen as 15, meaning that a cluster must consist of at least 15 data points to be considered valid.
- **Metric:** This parameter specifies the distance metric used to measure the similarity between data points. In this case, Euclidean distance metric is utilized, which calculates the straight-line distance between two points in Euclidean space.
- **Cluster Selection Method:** This parameter determines the method for selecting the final set of clusters from the hierarchical representation produced by the algorithm. The 'Excess of Mass' (EoM) method is employed for this modeling, which identifies the clusters that have the most substantial support in the data, effectively helping to find the most prominent clusters while filtering out smaller, less significant ones.

Once the HDBSCAN algorithm is applied to the training data, the embeddings are reduced to two dimensions using Principal Component Analysis (PCA) to facilitate visualization. The resulting 2D embeddings are then used to visualize the clustered data points and outliers in a scatter plot. Outliers identified by HDBSCAN are represented in gray, while the clustered data points are colored based on their assigned cluster labels using the 'hsv_r' color map.

To objectively evaluate the performance of HDBSCAN in generating clusters, the

silhouette score is calculated. The silhouette score measures the coherence and separation of the clusters. In order to have a benchmark for comparison, the silhouette score is also computed for the original labels, which represent the ground truth topic annotations. This enables a quantitative evaluation of how well HDBSCAN's unsupervised clustering results align with the true underlying topics.

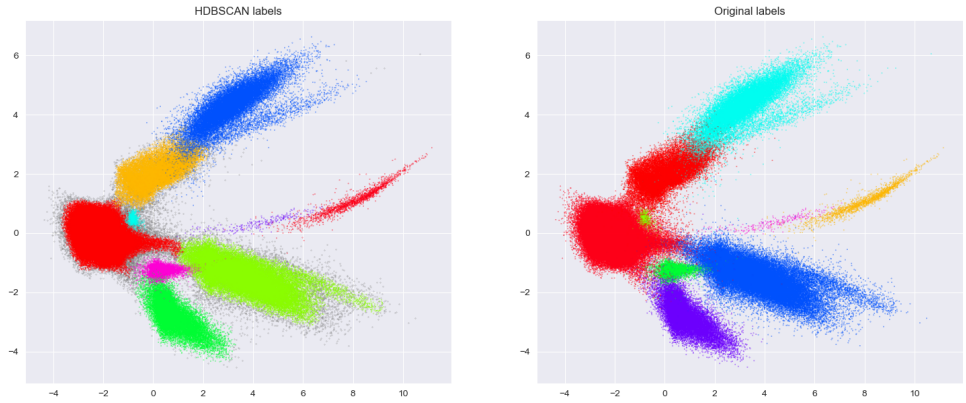


Figure 12: HDBSCAN clustering

Secondly, the KMeans algorithm was implemented for unsupervised clustering on the obtained embeddings. In this particular case, the algorithm was set with a single primary parameter: the number of clusters (`n_clusters`), which was specified as 9. This choice was informed by prior knowledge of the number of Level 1 (L1) classes for the DBPedia L1 dataset, which is an atypical situation. Typically, for topic detection tasks, the number of classes is not pre-established. Therefore, the selection of the 'number of clusters' parameter must be undertaken with considerable care and precision.

Following the execution of KMeans, the PCA method was employed to reduce the dimensionality of the data to a 2D space for visualization. Lastly, the silhouette score was calculated to quantitatively evaluate the quality of the clustering results obtained from KMeans.

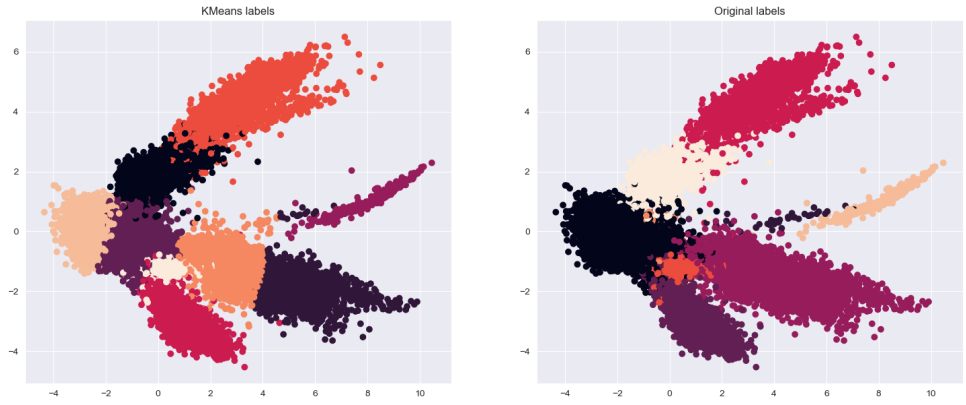


Figure 13: KMeans clustering

The silhouette score obtained from the KMeans implementation was approximately 0.48, which is less than the score of 0.68 from the original labels, and also below the 0.63 achieved with the HDBSCAN method.

The lower silhouette score from the KMeans algorithm suggests that the clusters formed may exhibit less coherence and separation compared to those generated by the HDBSCAN algorithm. Notably, a higher silhouette score represents a model with better-defined clusters. Thus, by comparing these silhouette scores, a quantitative benchmark is provided to evaluate the effectiveness of HDBSCAN and KMeans in creating meaningful clusters for the topic detection task.

However, the decision to favor HDBSCAN over KMeans wasn't solely based on silhouette scores. In addition to silhouette scores, other factors further highlight the potential superiority of HDBSCAN over KMeans for topic detection tasks. HDBSCAN's ability to automatically determine the optimal number of clusters, handle clusters of various shapes and sizes, and its robustness to noisy data and outliers offer considerable advantages over KMeans. HDBSCAN, unlike KMeans, can handle irregular and non-convex clusters, and it assigns outliers to a separate cluster, minimizing their impact on clustering results.

Additionally, HDBSCAN reveals hierarchical data structures by identifying not only individual clusters but also subclusters and cluster hierarchies. Its density-based approach enables the identification of clusters with varying densities, making it suitable for diverse datasets. Finally, HDBSCAN effectively manages high-dimensional data by leveraging density-based concepts, an area where KMeans could face challenges due to the "curse of dimensionality."

Taking into consideration all the previously mentioned points and the comparative quality of clustering results achieved by both the HDBSCAN algorithm and KMeans, a decision has been reached to continue with HDBSCAN for further analysis. Owing to its superior performance and unique characteristics, the HDBSCAN algorithm will be adopted as the selected modeling algorithm in the subsequent parts of this research.

4.2 Semantic Graphs for L1 Topic Classes

In this part of the project, semantic graphs are constructed for each of the nine labels following the clustering of the DBPedia text observations. These graphs help to study the connections between words, their synsets, and their hypernyms. A comprehensive explanation of the text preprocessing conducted for this project, and the connectivity of semantic relationships between words, is provided in 'Section 3.1.2. Text Preprocessing'.

The first step in this process is to group all tokenized words that share the same predicted classes, thereby generating nine lists, one for each class. For instance, all observations predicted as the 'Agent' class are grouped together. The 50 most frequent words from this list are selected as the top words for the 'Agent' class.

The process of acquiring these semantic layers is visualized in Figure 14. It starts with the initial extraction of the most frequent words and their synonyms, which are then subjected to hypernym calculations to form three levels of semantic content.

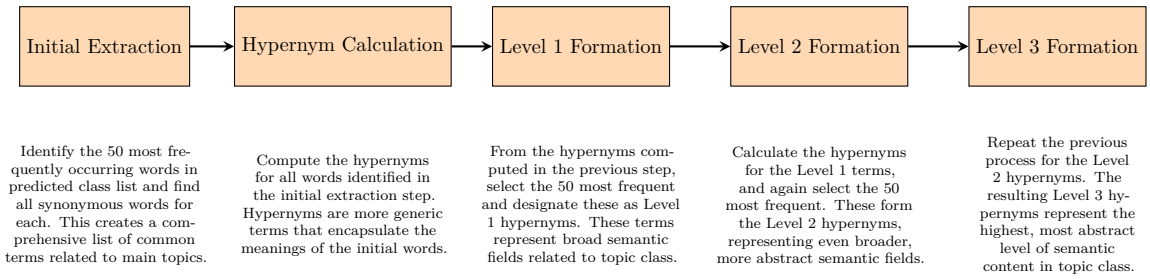


Figure 14: Process of acquiring semantic layers

Once the 50 most common words are chosen, the synsets of these words are also added to the class list. This process results in a collection of frequently encountered words and their synsets, which are saved as a list for the subsequent creation of semantic graphs.

Subsequently, to generate semantic graphs across three levels, the hypernyms of the words in these lists are calculated, and these hypernym words are integrated into the semantic graphs. The construction of these semantic layers is illustrated using the word 'institution' in Figure 15.

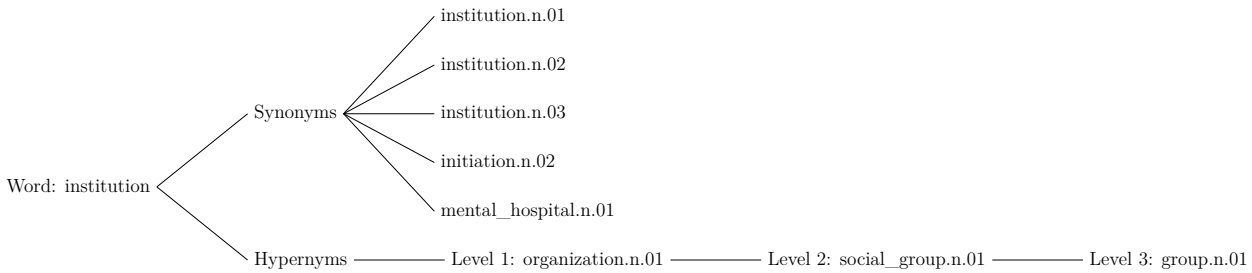


Figure 15: Example of semantic layers for the word 'institution'

Three levels of hypernym graphs are produced in succession, resulting in three graphs for each of the nine classes, each with different levels of granularity. The ensuing section explores the analysis of these semantic connections.

Figure 16 presents three hypernym graphs for the 'Agent' class, each illustrating a different level of granularity. Graph (a) (see Figure 16a) provides more granular, specific word connections for texts associated with the 'Agent' topic. In contrast, Graph (c) (see Figure 16c) exhibits connections between more abstract hypernym words for the same topic.
Each of these graphs offers unique advantages for analysis. The more abstract graph, created from the Level 3 hypernym list (Figure 16c), provides a broader understanding of the topic. Conversely, the Level 1 hypernym graph, depicted in Graph (a) (Figure 16a), offers detailed insight into the connections between the most frequently occurring words for the specific topic.

By analyzing all three semantic graphs, one can identify the topic name and also gain a general understanding of the theme by observing these hypernym connections. Such a multi-level analysis caters to both macro and micro analytical needs, thus facilitating a comprehensive understanding of the topic.

Calculating word similarities for each classes

Upon the completion of the three-tier semantic graph generation, pairwise word similarities within each graph are calculated. This step, crucial to understanding the semantic relationships among the words and hypernyms of each class, is accomplished using WordNet's `path_similarity` function.

The hypernym hierarchy, a foundational structure within WordNet, is a directed graph where nodes represent words and edges represent hypernym relationships (i.e., "is-a" relationships). In simple terms, this hierarchy captures the broader-to-narrower relationships

between words. For instance, "canine" might be a hypernym of "dog", while "animal" is a hypernym of "canine".

The `path_similarity` function gauges semantic similarity by identifying the shortest path between two words within this hypernym hierarchy. It quantifies this path in terms of the fewest hypernym relationships that need to be traversed to connect the two words. The resulting similarity score ranges between 0 and 1. A score of 1 signifies that the words share the highest degree of semantic similarity, while a score closer to 0 indicates less similarity.

For the scope of this study, a similarity threshold is set at 0.49. This threshold was determined based on preliminary testing, wherein multiple threshold values were assessed. The value of 0.49 consistently yielded the most meaningful clusters, providing clear differentiation between semantically similar and dissimilar word pairs. The procedure involves the computation of the `path_similarity` for each unique pair among the 50 most common hypernyms for each level of granularity. It is ensured that a word is not paired with itself, and that the `path_similarity` score is not `None`, thereby mitigating potential computational errors.

In the event that the `path_similarity` score for a word pair surpasses the predetermined threshold, the pair is classified as highly similar. Such word pairs are consequently deemed as belonging to a 'community group of words' within their specific class. This is a critical step, as the establishment of these community groups offers an in-depth understanding of the thematic composition and embedded patterns within the texts of a topic.

Detection of these community groups via path similarity scores is performed across each class and for all three levels of granularity, leading to a comprehensive analysis of 27 semantic node graphs. The utilization of these community groups, rich in semantically similar words, provides a robust foundation for understanding the subtle themes and patterns within the texts of each topic.

Figure 17 presents the obtained subgroups for three distinct classes: Agent (Figure 17a), Location, and Event. All three semantic node graphs shown in the figure are generated using level 1 hypernyms. Nodes sharing the same color in these graphs represent word groups with similarity scores exceeding a certain threshold within their respective class. For instance, in Figure 17a representing the 'Agent' class, the words 'organisation',

'unit', 'association', 'activity', and 'occupation' form a group with a pairwise similarity score exceeding the threshold value.

Figure 17 provides insights into various subtopics within each class, circled for clarity. Each of the nine classes features approximately 5-6 subtopics. For example, the Agent class (Figure 17a) includes subtopics such as Creativity, Compete, Organization, Action, and Living Organization, reflecting the semantic relevance of the associated texts.

Similarly, subtopics within the 'Place' class (Figure 17b) include 'Navigation/Direction', 'Natural Places', 'Structure', and 'Region/Location', reflecting logical subdivisions within the 'Place' category.

Finally, the 'Event' class (Figure 17c) is subdivided into subtopics such as Creativity, Location, and Time, offering a more detailed understanding of the related texts. These subtopics provide a more granular understanding of the text, thereby aiding in the understanding of its overall theme.

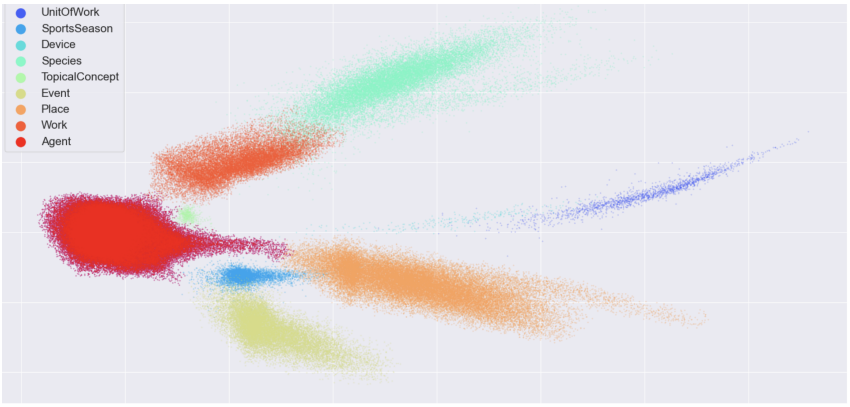


Figure 18: Embedding space colors for l1 class

Figure 18 illustrates the embedding space of HDBSCAN clustering for each of the nine classes. The observations belonging to the 'Agent' class are highlighted in red, those associated with the 'Place' class are delineated in orange, and the observations pertaining to the 'Event' class are depicted in mustard color.

Consequently, it is anticipated that the clusters of words, derived from the 'Agent' semantic graph (as shown in (Figure 17a),), would exhibit higher TF-IDF scores for the observations located within the red-colored area in Figure 18. This expectation emanates from the hypothesis that these high-scoring words bear more relevance within the 'Agent'

class, resulting in a closer spatial proximity of 'Agent' observations within the embedding space.

As an initial example, 'Compete' community group is considered, which is derived from the 'Agent' class. The 'Compete' community group comprises the words 'gamble', 'bet', 'play', and 'compete', as seen in Figure 17a, where the relevant node group is circled on the left. The 'Compete' group is defined in a list and the TF-IDF scores are calculated for all text observations. If the TF-IDF score is high, then the corresponding text embeddings in the embedding space are colored green; if the score is low, the corresponding embeddings are colored red. If there is no relationship (a TF-IDF score of 0) between a 'Compete' community group and a text observation, then the corresponding observations are colored black.

As demonstrated in Figure 19a, text observations belonging to the 'Agent' class cover the most substantial green area in the embedding space. This result is expected, given that the 'Compete' community group originates from the 'Agent' class's semantic graph. However, some smaller areas from other classes, such as 'Place' and 'SportSeason', are also colored green. This indicates that these classes' text observations share some semantic similarity with the 'Compete' words. The presence of the 'Compete' semantic group in the 'SportSeason' class is intuitive, as texts related to sports events are likely to contain the 'compete' subtopic.

Another example is the 'Water' community group, which is extracted from the 'Place' semantic graph Figure 17b. This group includes words such as 'stream', 'body of water', 'lake', 'water', and 'waterfall'. The same semantic analysis conducted for the 'Compete' group was applied to this node group. The expectation here is for text observations from the 'Place' class to be colored green, while other areas would primarily be colored black. As illustrated in Figure 19b, the 'Place' class's embedding space area is indeed predominantly green, with most other areas appearing black, save for a small green section belonging to the 'Agent' class.

The final example involves the 'Navigation' community group, which is a common community group extracted from both the 'Place' and 'Event' semantic graphs. The 'Navigation' community includes the words 'cardinal compass point', 'compass point', 'direction', and 'position'. This time, it is expected that the 'Place' and 'Event' observations would yield high TF-IDF scores, while the other areas would remain black. Figure 19c shows that multiple class regions, including 'Agent' and 'SportSeason', have high TF-IDF

scores for the 'Navigation' community. The 'Agent' class has the most densely green area in the embedding space, while the 'Place' and 'Event' classes exhibit sparser green observations. Therefore, the 'Navigation' community group does not provide the expected results in the embedding space.

This analysis reaffirms that the HDBSCAN algorithm performs admirably on the DBPedia dataset, effectively clustering text observations and grouping these texts into specific topic groups. Our examples of the 'Compete', 'Water', and 'Navigation' community groups serve as evidence of this proficiency. Moreover, the semantic node graphs' ability to identify subtopics from the most common words and their hypernyms, as demonstrated in our 'Water' and 'Compete' examples, facilitate the extraction of more informative meanings from texts with a shared abstract topic.

However, the value of this method is not limited to specific examples or datasets. The underlying methodology and logic have broad applicability. This technique involves:

Semantic representation: Text observations are embedded into a space, where their relationships can be visualized and interpreted. This aids in understanding the inherent meanings and associations within textual data, regardless of its source or context.

Community Group identification: From this semantic representation, community groups, or clusters of related terms, are identified. This allows for a higher-level understanding of themes or topics present in the data. For instance, words like 'stream', 'lake', and 'waterfall' might be categorized into a broader 'Water' theme. This principle can be applied to any dataset, identifying key themes or motifs that emerge from the data.

Expectation versus reality analysis: Once these community groups are identified, one can set expectations based on intuition or prior knowledge. This method allows for a direct comparison between these expectations and the actual results. For instance, we might expect that the 'Water' community group is predominantly associated with a 'Place' category. But the actual distribution in the embedding space might reveal unexpected associations, indicating that our understanding of the data's semantics may need to be expanded or adjusted.

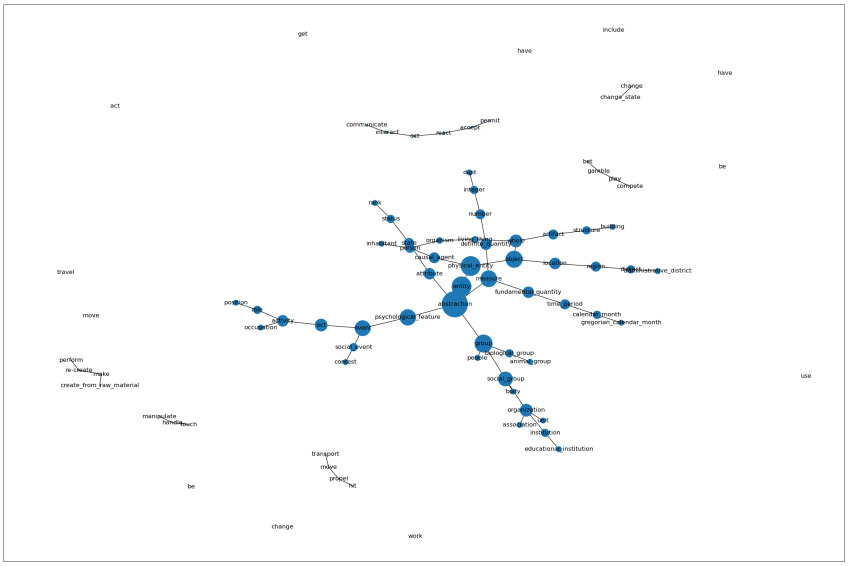
Inter-class relationship: The methodology is not static. It encourages repeated analysis and refinement. For example, after analyzing the 'Water' community group, one can apply the same semantic analysis to another group, comparing and contrasting results.

This iterative process allows for a deep and multi-faceted understanding of the data.

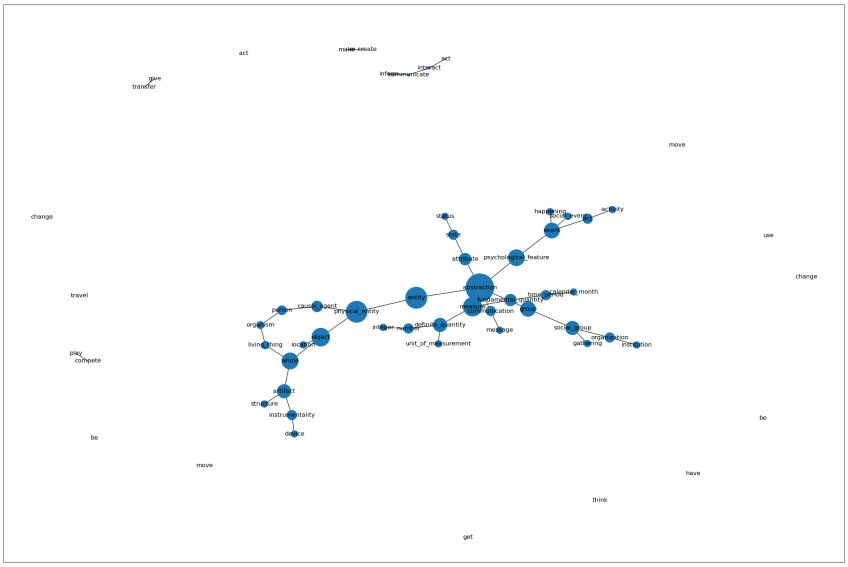
In general, this process is helpful for anyone seeking to dissect large volumes of textual data, identify inherent themes, and understand relationships within the data. Whether it's analyzing customer feedback, categorizing academic literature, or interpreting user-generated content on a social media platform, the same steps can be applied.

As the analysis broadens to encompass nine different topic classes, discerning subtopics from the node graphs for this specific dataset becomes increasingly intricate. This complexity is emphasized by instances where classes share subtopics, potentially affecting the extraction of detailed information for individual classes. A prime example is the 'Navigation' community group, where the TF-IDF scores diverged from expectations, and the anticipated density of green areas in the embedding space was not achieved. Such overlaps and shared themes among classes underline the imperative for an enhanced, more granular approach to designate text observation areas as subtopics for focused analysis.

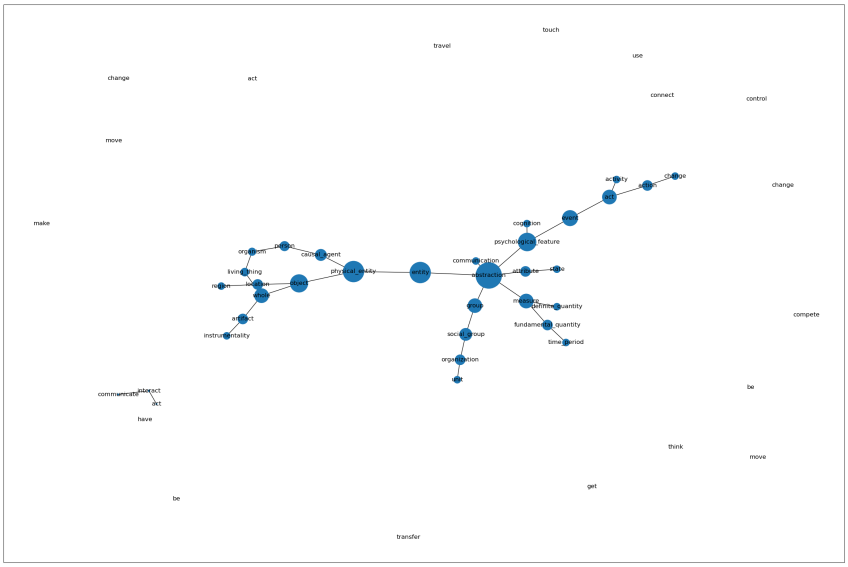
Moving forward, the quest for a more autonomous and practical solution for subtopic detection across diverse datasets calls for an evolved strategy in creating semantic graphs and community node groups. The intricate interdependencies and overlaps spotlighted in this study amplify this need. Consequently, the subsequent chapter will delve deeper into potential advancements in topic detection, drawing upon the insights obtained from this research



(a) Agent hypernym level 1 semantic graph

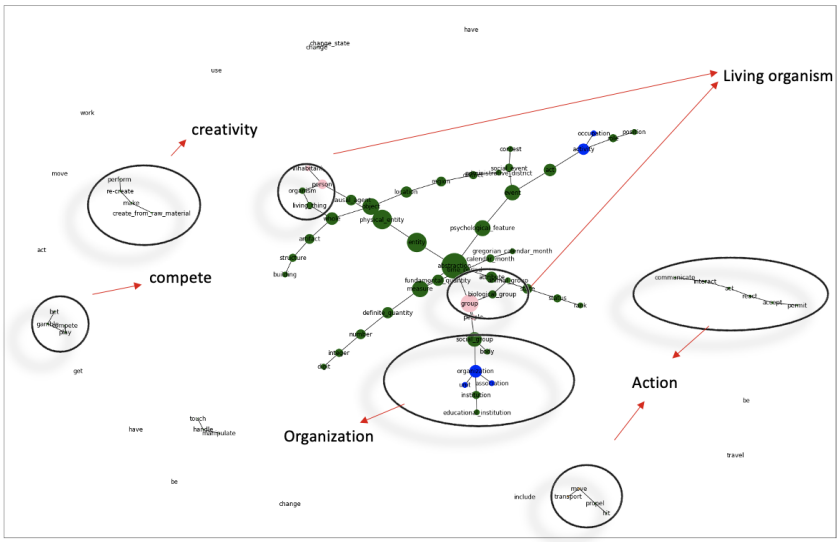


(b) Agent hypernym level 2 semantic graph

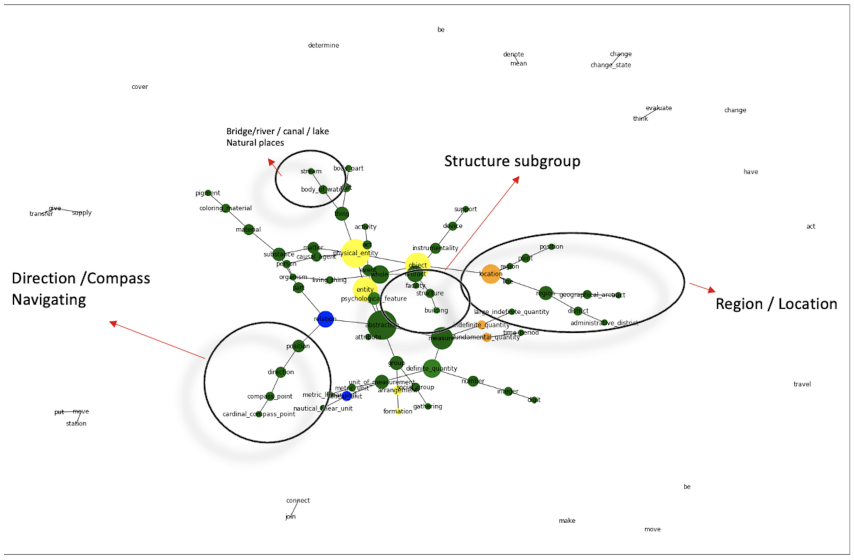


(c) Agent hypernym level 3 semantic graph

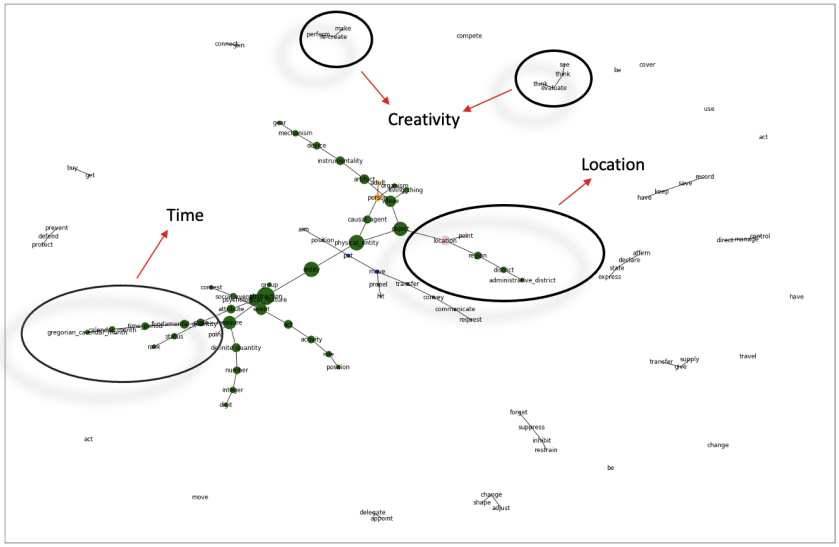
Figure 16: Agent class 3 levels of hypernym graphs



(a) Agent hypernym level 1 Node Groups

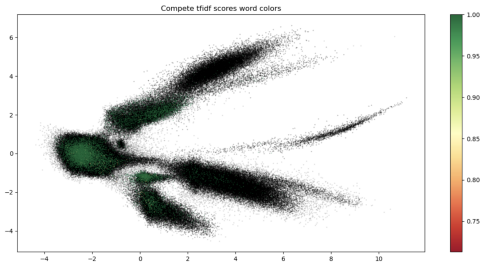


(b) Place hypernym level 1 Node Groups

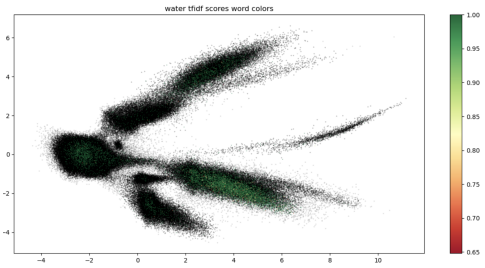


(c) Event hypernym level 1 Node Groups

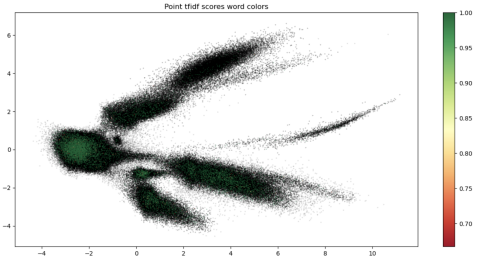
Figure 17: Subgroups for Agent, Place and Event classes



(a) Compete: TF-IDF Scores of 'Compete' Community Group in the Embedding Space - Representing words: 'gamble', 'bet', 'play', and 'compete'



(b) Water: TF-IDF Scores of 'Water' Community Group in the Embedding Space - Representing words: 'stream', 'body of water', 'lake', 'water', and 'waterfall'



(c) Navigation: TF-IDF Scores of 'Navigation' Community Group in the Embedding Space - Representing words: 'cardinal compass point', 'compass point', 'direction', and 'position'

Figure 19: Visualization of TF-IDF Scores for Selected Community Groups within the Embedding Space

5 Improvements on implementation and more granular analysis on topic detection

This section of the thesis presents enhancements to the topic detection approach previously implemented, with the objective of addressing the highlighted limitations and improving the overall effectiveness of the methodology. The advancements include the application of the Louvain algorithm to generate more robust community groups, alongside the incorporation of hierarchical clustering using the Ward linkage method to enhance the granularity of clusters.

These modifications promise to bring about more precise results and deeper insights in semantic topic analysis, providing more practical and autonomous solutions for subtopic detection across a diverse range of datasets. The upcoming subsections delve into the specifics of these enhancements. These sections explain the theoretical foundations behind their implementation and illustrate how they practically influence the topic detection process.

5.1 Advancing Semantic Analysis: The Louvain Method for Community Detection

In Chapter 4, an extensive manual analysis of semantic node graphs is undertaken to extract community groups based on node connections. These groups are derived from the nine known L1 label classes present in the DBPedia dataset. While this approach yields precise community groups for classes with predetermined labels, the need for a versatile topic detection solution, applicable to datasets without prior knowledge of classes, is recognized. This leads to the integration of the Louvain algorithm, an enhancement that enables the identification of more refined and adaptable community groups.

The incorporation of the Louvain algorithm improve the reliability and applicability of community detection within semantic node graphs. Implemented using the Python `community louvain` library, this approach revolves around the automatic identification and adjustment of community groups in the semantic graph, with a focus on optimizing modularity—metric for quantifying network division.

At its core, the Louvain method is founded upon the concept of community groups formed by nodes with strong intra-group connections and weaker inter-group connections. The algorithm embarks on an iterative journey, wherein each node is assigned to an initial

community. Subsequently, the algorithm assesses the potential increase in modularity—a measure denoted as Q—that could result from relocating nodes to different communities. [28]

Mathematically, modularity Q is expressed as:

$$Q = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(c_i, c_j) \tag{1}$$

Where:

- A_{ij} signifies the weight of the edge between nodes i and j.
- k_i and k_j represent the degrees of nodes i and j.
- m corresponds to the sum of all edge weights in the network.
- c_i and c_j denote the communities to which nodes i and j belong.
- $\delta(c_i, c_j)$ is the Kronecker delta function.

The term $\left(A_{ij} - \frac{k_i k_j}{2m}\right)$ encapsulates the difference between the actual edge weight and the expected edge weight between nodes i and j in a randomized network. The summation aggregates this difference over all pairs of nodes. Modularity Q gauges the extent to which observed edge density within communities deviates from randomness [28].

In each iteration, the Louvain algorithm computes the potential change in modularity arising from node repositioning to neighboring communities. Nodes are reassigned to the community that maximizes modularity increase. The iterative process endures until further modularity enhancement becomes infeasible, culminating in network partitioning into communities characterized by stronger intra-community connections compared to inter-community connections [28].

The Louvain algorithm enhances visualization by assigning the same color to nodes within the same community, facilitating comprehension of community structure. The `resolution` parameter, denoted as r, acts as a vital tuning element, influencing community granularity. Higher resolution values yield more detailed, smaller communities, while lower values result in larger, less detailed communities. Mathematically, the resolution parameter r can be adjusted to control community size and density:

Smaller $r \rightarrow$ Larger, less granular communities

Larger $r \rightarrow$ Smaller, more granular communities

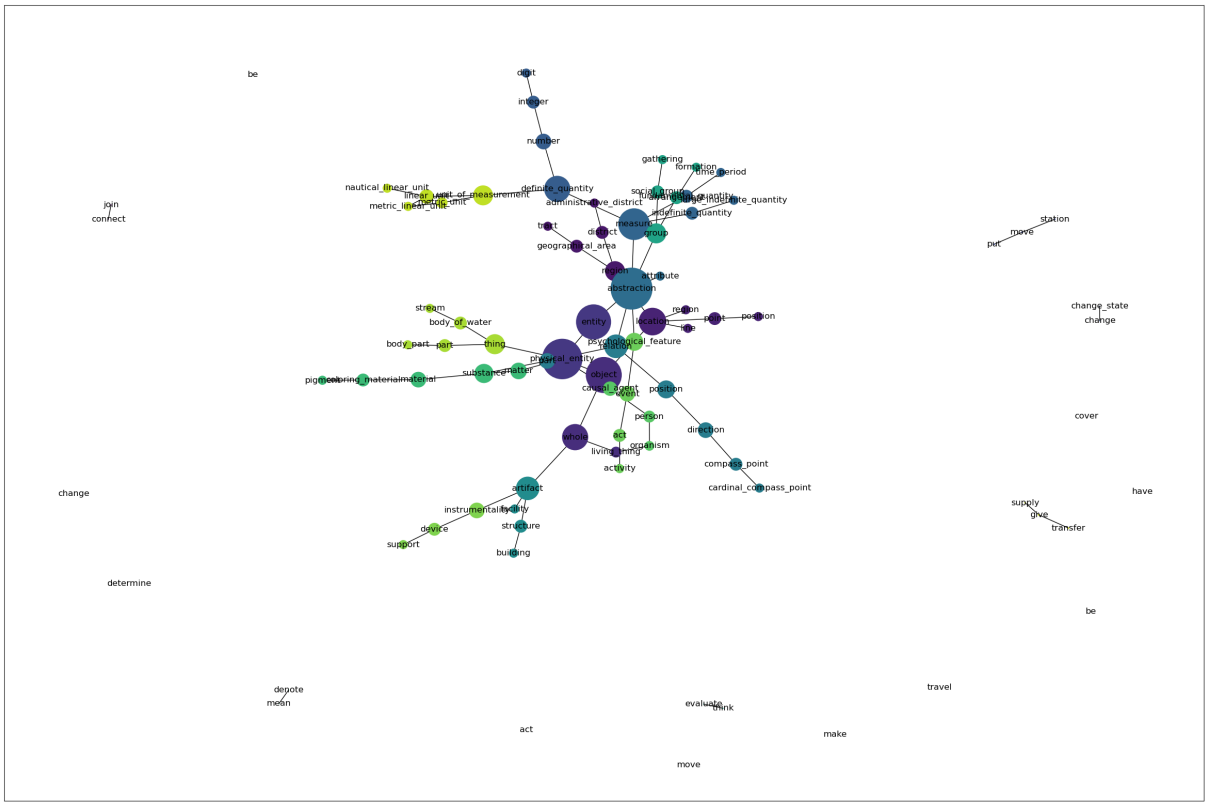


Figure 20: Semantic graph for Place class with Louvain community

The Figure 20 displays the semantic node graph for level 1 hypernyms of the Place class, generated using the Louvain algorithm. Nodes belonging to the same community group are denoted by the same color. As previously discussed, the granularity of these community groups can be adjusted through the resolution parameter within the Louvain method. For this particular scenario, the resolution parameter was set to 5, resulting in community groups that, for the Place class, comprise an average of five nodes each.

5.2 Advancing Topic Detection: Hierarchical Clustering Combined with HDBSCAN

The core idea of this methodology lies in the application of hierarchical clustering atop the foundational clusters derived from HDBSCAN. While HDBSCAN is intrinsically hierarchical, it operates dominantly as a density-based clustering algorithm. It detects

clusters based on varying densities, employing its hierarchical technique mainly to discern regions of distinct densities, which are then reduced to formulate the overarching clusters. The output of HDBSCAN includes the identification of core samples and the assignment of labels to each data point, either designating them to a specific cluster or marking them as noise.The subsequent application of traditional hierarchical clustering is introduced to further parse and clarify these overarching clusters, offering a more nuanced insight into potential subclusters within them.

Starting with HDBSCAN, the data undergoes clustering with parameters set at a minimum cluster size of 15 and the cluster selection method labeled as 'eom' or Excess of Mass. Upon completion, the cluster assignments and core samples produced by HDBSCAN are utilized as input data for the subsequent hierarchical clustering phase. Post this, the data is streamlined using Principal Component Analysis (PCA), serving to lower its dimensionality and thus, making it more amenable for visualization.

On this preprocessed data, traditional hierarchical clustering is administered. This step constructs a linkage matrix, which represents pairwise distances between clusters. This linkage matrix serves as a foundation for merging clusters iteratively, based on their relative proximities, as governed by the Ward's linkage method. From this matrix, flat clusters are derived by setting a specific maximum distance threshold. Data entities within this stipulated distance are then collated into a singular cluster, refining and enhancing the granularity of the previously established labels.

These newly crafted hierarchical labels are allocated to the data points, symbolizing the distinct cluster each belongs to, and thereby, facilitating a deeper and more precise analysis of the initial classes.

A dendrogram, produced subsequently, visually elucidates the hierarchical connections between clusters and individual data entities. This process is illustrated in the final visualization, where outliers are demarcated in gray and the clustered data points are color-coded in accordance with their hierarchical labels.

Through the application of this method, the data revealed distinct subgroups within the original 9 classes, resulting in a total of 21 identifiable clusters. This structured approach offers a more detailed analytical framework, illuminating the inherent complexities within the dataset. Utilizing hierarchical clustering on the preliminary clusters established by HDBSCAN provides a systematic approach to further differentiate and classify topics.

Following the implementation of hierarchical clustering, Figure 21b depicts the 21 subclusters generated from the initial 9 l1 classes. The number of subclusters was determined to be 21, a decision informed by the dendrogram graph shown in Figure 21a, where the threshold value was set at 40

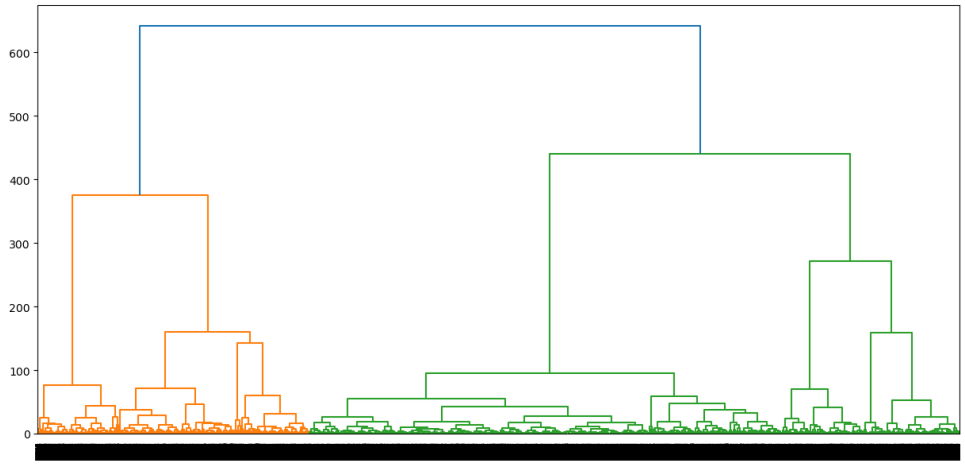
Crucially, after the hierarchical clustering phase, the core information pertaining to the base clusters, from which these subdivisions emerged, is retained. This conservation is pivotal as subclusters undergo both intra-cluster scrutiny and are juxtaposed against all 21 clusters for a comprehensive analysis. To illustrate, the principal cluster dubbed 'Agent' splintered into five distinct subclusters following the hierarchical clustering. These five were compared intrinsically and also juxtaposed against the broader 21 clusters, enriching the overall comparative analysis.

5.3 In-depth Analysis and Refinement of Hierarchical Clusters

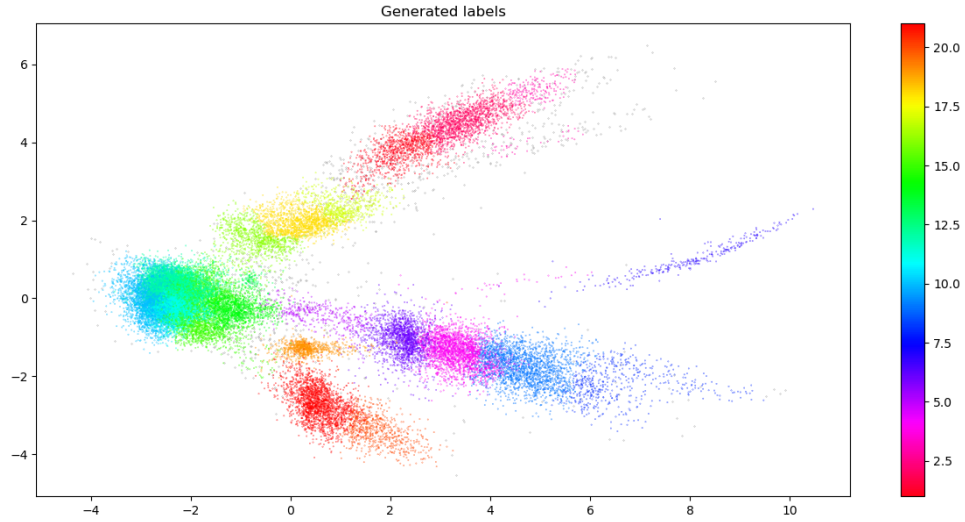
The first step in subcluster analysis involved another round of text preprocessing for these 21 subclusters. As in the earlier steps, the most frequent words, most frequent hypernyms, and hypernyms of the most frequent words are identified. Semantic node graphs for each of the 21 subclusters are then calculated using the Louvain algorithm. With the hypernym graphs and the list of most frequent words for the subclusters in hand, some abstract words are excluded from the top words lists—words such as 'like', 'do', 'be', and 'have', which do not contribute substantial information about the subcluster topics.

Upon a thorough examination of the semantic node graphs of these subclusters, unique topic names is assigned to the 21 subclusters. This method is effective in providing specific and distinct topic names to several subclusters. However, certain other subclusters required more comprehensive clarification.

For instance, the hierarchical clustering combined with HDBSCAN methodology was applied to the 'Place' class (L1), which yielded 6 subclusters. These subclusters are designated names by the author after an analytical review of the semantic graphs and node connections. In Table 4, it can be observed that certain subclusters—such as Celestial body and astronomy, Legal cases and Law, and Railway and public transit system—have been accurately segmented into detailed topic levels. This precision in segmentation has been corroborated by an examination of the text observations pertaining to these subclusters.



(a) Dendogram of Hierarchical clustering



(b) Clusters for Hierarchical Clustering Combined with HDBSCAN

Figure 21: Hierarchical Clustering Combined with HDBSCAN

Nonetheless, it is observed that certain subclusters like Natural places and structures and Railways and Museums encompassed several general topics that required further subdivision. These subtopics contained two distinct themes that could be separated. Furthermore, some subclusters still maintained relatively abstract topic names, which resulted in intertwined subclusters within the cluster, such as the Natural Places subcluster and the Natural Places and Structures subcluster. To resolve these issues, the following section delves deeper into a more detailed analysis of different subclusters.

Table 4: Place Class Subclusters

Subluster No.	Subluster	Top Words	Top Hypernyms
6	Railways and Museums	line, located, station, museum, railway, state, airport, city, new	move,travel,position, location, mark, administrative district
4	Natural places and structures	located, river, state, bridge, lake, mountain, county, airport, km	structure,location, travel, administrative district, position
9	Natural Places	river, lake, located, mountain, county, km, state, south	location, administrative district, travel, structure, direction,region
8	Celestial body and Astronomy	asteroid, river, dam, church, located, orbit, lake, discovered, planet, km	time period, location, structure, metric linear unit, travel, celestial body
7	Legal cases and Law	court, state, united, case, supreme,decision, law	government, administrative district, person, area, container,metallic element
5	Railway and public transit system	line, railway, service, station, system, city, rail, new, operated	mark,activity, formation, location, travel, position

5.3.1 Agent label Subcluster analysis

A detailed examination of subclusters within primary clusters is undertaken to enhance the clustering methodology. This analysis is characterized by two main objectives:

1. **Analysis of Text Observations:** Text observations within certain subclusters, despite sharing a common label, have positions in the embedding space that differ from the majority. This variation prompts an investigation into their thematic consistency.

2. **Segmentation of Abstract Subclusters:** Some subclusters, due to their abstract nature, are further segmented into more defined thematic units. This segmentation aims to ensure more accurate cluster labels, contributing to a refined clustering outcome.

Cosine similarity scores, computed using a PCA-based two-dimensional embedding space, are used to measure the proximity between text observations. While these scores

typically fall between 0 and 1, the precise range and interpretation require clarification, given the noted ambiguities.

The main goal is to understand inherent clustering patterns within the embedding space. The density of text observations is analyzed to verify the accuracy of their clustering. When areas of sparsity are identified, an exploration into the potential causes is initiated. Such areas, especially those with lower cosine similarity scores, provide insights regarding the potential breadth of subclusters and indicate areas for further segmentation or the identification of new subgroups.

For this analysis, a specific, yet-to-be-defined, similarity score threshold is established for each cluster. Subsequently, similarity matrices are filtered to highlight pairs that do not meet this threshold. Each subcluster's number of such pairs is then documented.

These pairs are then visually represented on scatter plots, enabling the identification of potential new subclusters and outlier observations. The visualizations provide clearer insights into the relationships between text observations and help in the further segmentation of potential new subclusters.

In parallel, relationships between different subclusters that seem to have similar themes, but are located at a distance in the embedding space, are also examined. This facet of the analysis aims to enhance the understanding of the underlying structure of the textual data, leading to the identification of densely-clustered areas and the interpretation of sparseness.

After identifying these pairs, they are visually represented on a scatter plot, facilitating the visual identification of potential new subclusters and outlier observations. This visualization aids in understanding the relationships between text observations, and a more detailed investigation is conducted to pinpoint potential new subclusters.

Simultaneously, the connection between different subclusters that share similar topics, despite their distance in the embedding space, is also considered. This analysis helps to analyse the underlying structure of the text data, leading to the detection of dense clusters, the understanding of sparseness, and the creation of more specific and cohesive clusters.

Table 5, shows 5 subclusters for the Agent class. After examining the semantic graphs and text observations for these subclusters, the following subcluster names are assigned:

Table 5: Agent Class Subclusters

Cluster No.	**Cluster**	**Top Words**	**Top Hypernyms**
15	Sport organizations	team, league, party, first, record, new, club, national	structure, location, travel, administrative district, position
12	Person	born, state, school,american, first, year, university, played, one	move, time period, travel, work, gregorian calendar month
10	Sport player	born, team, played, league, season, player, coach, national, football, university	perform, move, time period,travel
14	Educational institutions and companies	school, university, state, service, city, one, new, bank, company	time period, travel, activity, educational institution, body
11	Sports player	born, world, team, championship, played, first, state, american, leagues	travel, make, time period, person, perform

"Sport Organizations (15)", "People (12)", "Sport Player (11)", "Educational Institution (14)", and "Sports Player"(10). In the following parts that explain the analysis, cluster numbers are used instead of the names for the subclusters.

Since subclusters 10 and 11 exhibit very similar semantic graphs and contexts in their text observations, and are located in close proximity within the embedding space, the same topic names have been assigned to these different subclusters. After analyzing the text observation pairs with lower similarity scores, these subclusters can be merged into a single, denser cluster.

Distinct similarity score thresholds are chosen for each class of subclusters based on preliminary testing where various threshold values were experimented with. The aim was to ascertain which threshold resulted in the most cohesive and meaningful clusters for each class. Given that the density of observations varied across different clusters, there was a clear need for class-specific thresholds. Hence, each class's threshold was determined by the level that provided the coherence within its respective subclusters.

A similarity score threshold of 0.7 has been selected for the Agent class subclusters,

and all text pairs within the clusters falling below this threshold in the embedding space have been calculated. For Cluster 11, all cosine similarities for text observation pairs are above the threshold. For the other subclusters, text pairs which have less similarity score than the selected threshold value have been saved for deeper analysis.

Figure 22 presents text observations that are members of dissimilar text pairs for subclusters 10 and 12 at least once. Green observation points represent dissimilar text pairs from subcluster 10, while red observation points symbolize those from subcluster 12.

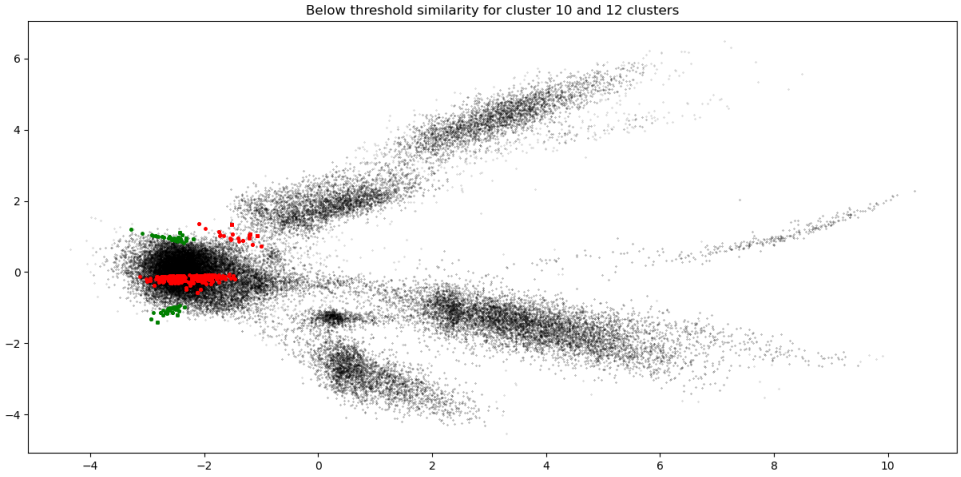
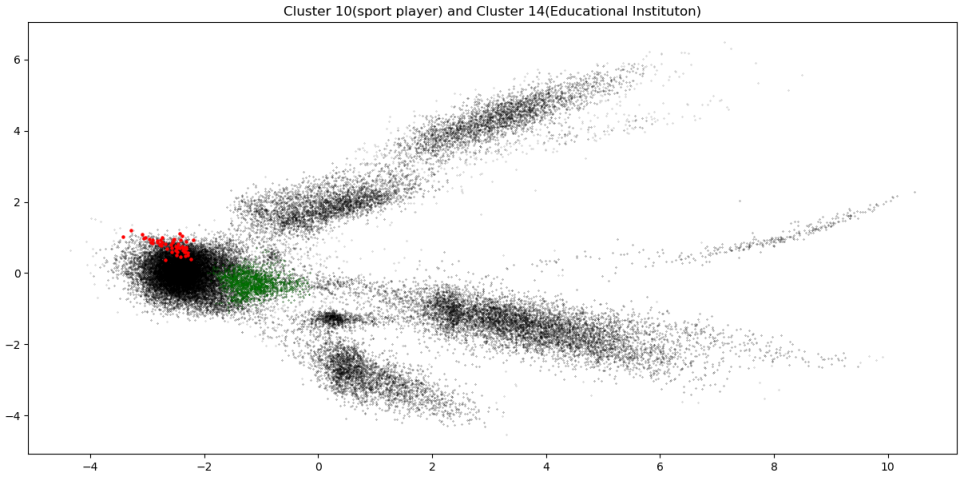


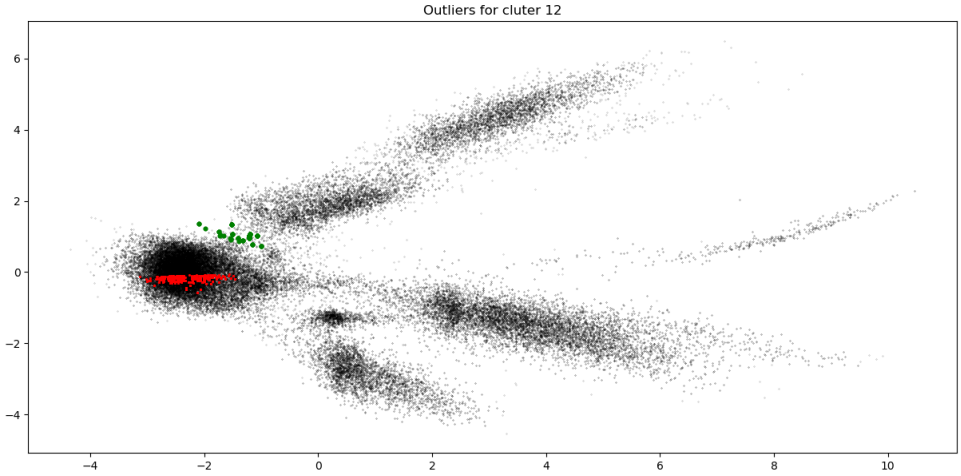
Figure 22: Text Pairs for subclusters 10 and 12

Upon analyzing the dissimilar observation pairs for both subclusters, two scenarios are identified. Firstly, for subcluster 10, two separate observation groups emerge from the calculation of text pairs: one group in the upper part of the subcluster and one in the lower area. When these group observation texts are examined, it is determined that these two parts genuinely correspond to different subgroups. The observations in the upper part are categorized under "Educational Institution," while the group located in the lower part pertains to "Athlete." Interestingly, part of subcluster 10 shares the same topic with another Agent subcluster, cluster 14. This analysis reveals that different subclusters can share the same topic even if they are not closely located in the embedding space. Therefore, text observations related to the educational institution can be combined with cluster 14, and the remaining observations can be associated with the cluster 11 topic, "Sports player."

In Figure 23a , the red text observation points represent the observations related to "Educational Institutions" that belong to Cluster 10. Meanwhile, the green-colored



(a) Subcluster 14 and observations from Subcluster 10 which belong to topic 14



(b) Outlier text pairs which do not provide information for Subcluster 12

Figure 23: Analysis of text pairs for cluster 10 and cluster 12

observations represent Subcluster 14, which is associated with the topic "Educational Institutions and Companies." As illustrated in the figure, two different cluster groups in the embedding space, though located far from each other, share the same topic.

Another consideration in cosine similarity analysis is the presence of outliers, which might lead to text pairs that do not convey useful information. For example, in Cluster 12, observations that are members of dissimilar text pairs are illustrated in Figure 23b. The green observations in this figure represent outliers for the subcluster density. Unfortunately, these two differently colored areas for Subcluster 12 do not reveal distinct topics. The green-colored observations, positioned far from other observations in Cluster 12 within the embedding space, lead to dissimilar text pair cosine similarity scores. Therefore, the text pair similarity method does not assist in extracting more specific topics for Subcluster

12. However, it does aid in identifying outliers within the embedding space.

5.3.2 Species label Subcluster analysis

Another example of similarity analysis within clusters is performed for the Species label.Table 6 displays the top words and hypernyms for the subclusters, and Figure 24 illustrates their locations in the embedding space. The Species label includes three subclusters: "Species (1)," colored green; "Plant (2)," colored red; and "Animal – Horse (3)," also colored red.

Table 6: Species Class Subclusters

Cluster No.	**Cluster**	**Top Words**	**Top Hypernyms**
1	Species	species, family, genus, found, known, one, habitat, name	travel,taxonomic group, person, time period,experience
2	Plant	species, family, genus, found, habitat, known, forest, endemic, tropical	taxonomic group, currency, person, experience, unit, collection
3	Animal (Horse)	species, race, stake, genus, family, thoroughbred, racehorse, desmopachria	travel, taxonomic group, time period, digit

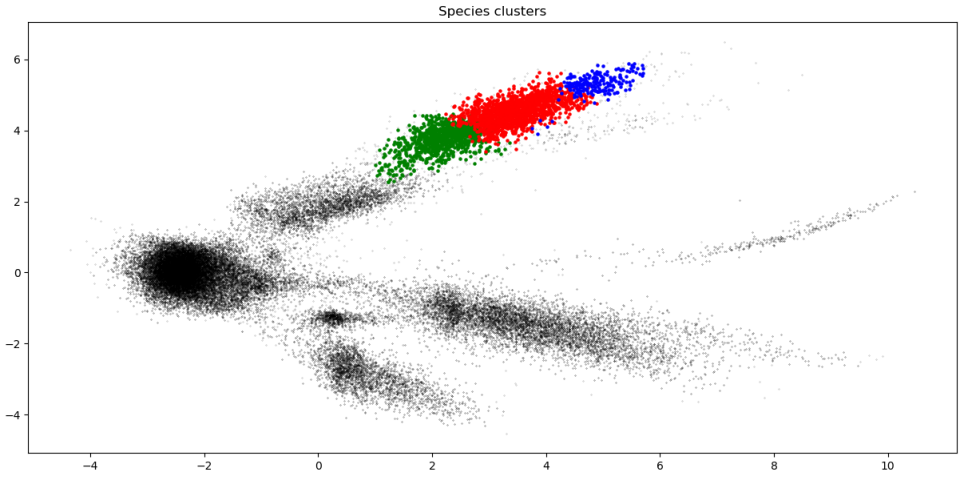
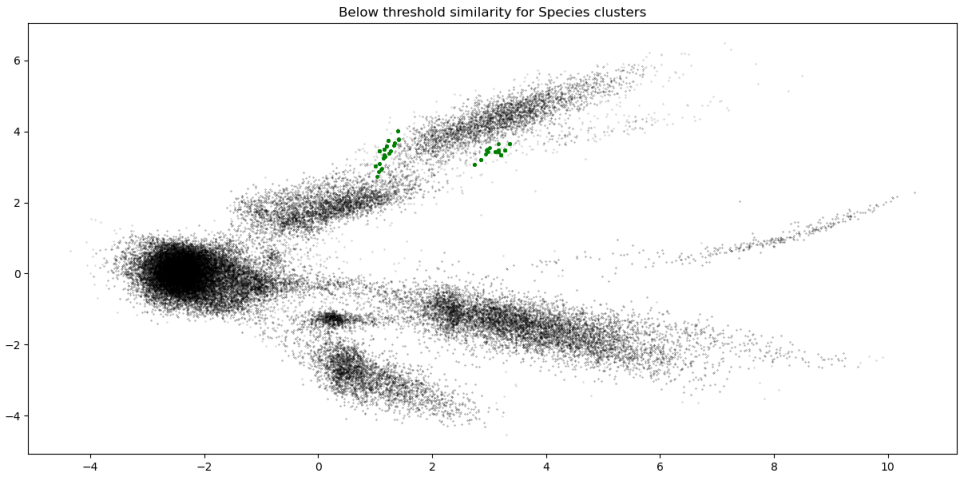


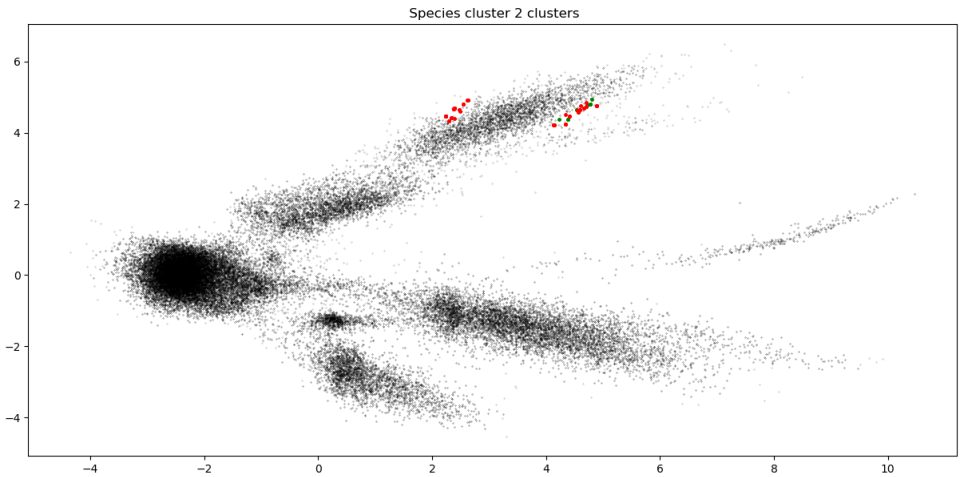
Figure 24: Species Subclusters in the Embedding Space

Since the text observations for each subcluster within the Species class are located closer to one another than those in the Agent class, a higher degree of similarity is expected among them. Therefore, a cosine similarity threshold of 0.95 has been chosen for the Species. This higher threshold reflects the tight clustering of text observations

within each subcluster, necessitating a more stringent similarity criterion to distinguish between them. Upon calculating pairwise similarities for each subcluster, subclusters 1 and 2 reveal text pairs with scores lower than this threshold value. In contrast, subcluster 3 does not present any observation pairs situated far from each other, consistent with the more specific topic that characterizes subcluster 3 in comparison to the other two subclusters.



(a) Subcluster 1 text pairs similarity analysis



(b) Subcluster 2 text pairs similarity analysis

Figure 25: Text pair similarity analysis for subclusters 1 and 2

Analyzing the dissimilar text pairs in the embedding space for subcluster 1 reveals two distinct subgroups. In Figure 25a, the left green part contains observations with the topic "Horse," whereas the right green colored part encompasses more abstract topics that could

still be named "Species," as it includes both "Plant" and "Animal" observations. The left part is therefore assigned the "Horse" topic, the same as cluster 3, further highlighting that different subclusters can contain the same topic even if they are remote from each other in the embedding space.

Subcluster 2 text pairs are then examined, and two distinct observation groups are identified, as shown in Figure 25b. The left part of the observations belongs to the "Plant" topic, while the right-side observations correspond to the "Animal" topic.

From the example analysis for Agent and Species labels, it can be concluded that different subclusters may encompass the same topic even if they are distant in the embedding space. Moreover, the calculation of cosine similarity scores for text pairs within a subcluster can assist in extracting more specific data topics. Generally, these dissimilar text pairs indicate that they pertain to different subtopics within the subclusters and should be treated as distinct topics even though they belong to the same subcluster.

The results of this analysis demonstrate that nested hierarchical clustering can facilitate the discovery of subtopics within subclusters. By comparing these subtopics within the subcluster as well as across all subtopics within the cluster, common themes for various subclusters may be identified. However, this localized text observation analysis is not particularly user-friendly and can be time-consuming for detecting topics across all observations. Consequently, interactive tools have been developed to provide a more user-friendly and automated solution for topic detection tasks. Different tools have been crafted for various use cases, and the subsequent chapter will detail these tools.

6 Enhanced Interactive Tools for Topic Detection Analysis

This chapter presents an in-depth look at interactive tools designed for topic detection analysis. The tools, employing semantic node graphs for interpretability and embedding spaces for quality analysis, aim at facilitating user-friendly interaction and automation in text data analysis.

The tools are designed to recognize the complexities involved in these methods, introducing strategies to enhance usability and efficiency. Two main methodologies are explored: Local analysis, involving hierarchical clustering combined with HDBSCAN, and global analysis, employing nested hierarchical clustering. These can be synergistically used to attain accurate and properly delineated topics.

Additionally, the chapter delves into the examination of embedding spaces within subclusters, leading to the calculation of similarity scores. These scores offer insights into the density of the subclusters, the discovery of more specific subtopics, and the interconnectedness of different subclusters.

The following sections detail the features, functions, and significance of these tools. By demonstrating how they can be employed to extract meaningful insights from complex textual data sets, this chapter contributes to the broader context of text analysis.

6.1 Local Analysis: Cluster Selection with Hierarchical Clustering and HDBSCAN

As the first step of topic detection analysis, local analysis aids in determining the number of topics and allows the user to zoom in on specific areas of the embedding space for a deeper understanding. The user interacts with different sliders, namely ‘Threshold’ and ‘Cluster’, interconnected and designed to provide control over topic selection and analysis as shown in the Figure 26.

These two sliders are interconnected; when the Threshold slider is changed, the Cluster slider is also updated accordingly, and the figures are simultaneously updated when the user makes changes to the sliders.

Initially, the user selects a threshold value to determine the number of topics to be

created using the hierarchical HDBSCAN algorithm. A higher threshold value leads to fewer clusters, while a lower threshold value leads to more clusters.

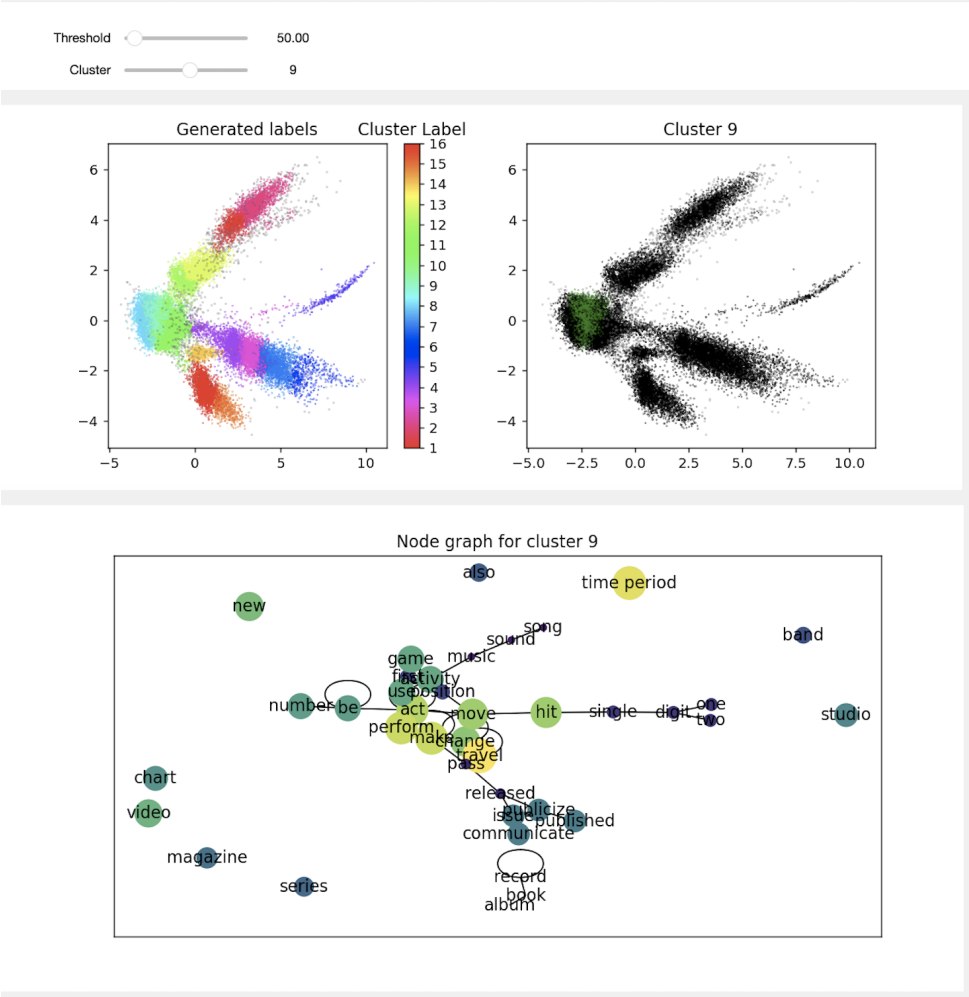


Figure 26: User Interface for Cluster Selection: Interactive sliders for threshold and cluster selection enable users to control the number of topics and the analysis of semantic graphs.

After selecting the number of clusters with the Threshold slider, the next step involves using another slider called Cluster, which enables the user to select a specific cluster from the embedding space. This aids in the analysis of the semantic node graph for the chosen cluster in the model. By picking a particular cluster through this second slider, the user can observe the positions of the text observations within the embedding space and scrutinize the corresponding semantic graph.

The local analysis tool, illustrated in Figure 27, offers a more specific examination within the chosen area of the embedding space. Upon clicking a specific text observation, the text is immediately displayed in the output. This selection allows the user to find

neighboring text observations situated near the selected one. A slider in Figure 27 defines the closest n observations to be analyzed. For example, if set to 10, a group of 11 observations, inclusive of the chosen one, is analyzed, and a semantic graph is generated from them. Furthermore, the output displays all text observations, granting the user the ability to view the entire group and designate a topic name for these texts.

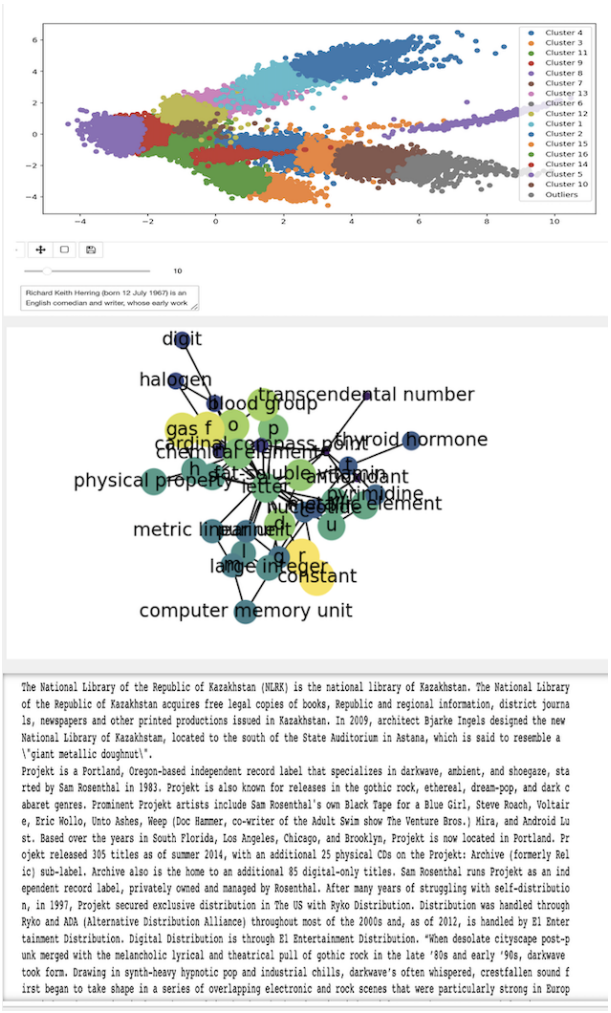


Figure 27: Local Analysis Tool: An illustration of the interactive tool that allows users to perform a specific examination within the embedding space, selecting text observations and analyzing neighboring texts.

This local analysis helps to find subtopics within a topic and analyzing how observations located next to each other are related to each other from a semantic perspective. Therefore, this local analysis helps the user to zoom in on a specific area and gain a deeper understanding of the topic assigned to this group of texts.

6.2 Nested Hierarchical Clustering: Subtopic Analysis with Dual Thresholds

Nested hierarchical clustering provides a method for detailed comparison between smaller groups within larger clusters, enabling a comprehensive examination of both main topics and underlying subtopics. This approach is facilitated by the use of the HDBSCAN algorithm, along with two connected threshold sliders.

Initially, the user determines the primary clusters by utilizing the Threshold 1 slider. Following this selection, the Threshold 2 slider is then employed to define subclusters within the main clusters. It is important to note that these two threshold sliders are intrinsically linked; Threshold 2 cannot exceed Threshold 1, ensuring that it will invariably yield more clusters. This relationship between the sliders ensures a logical and coherent subclustering process.

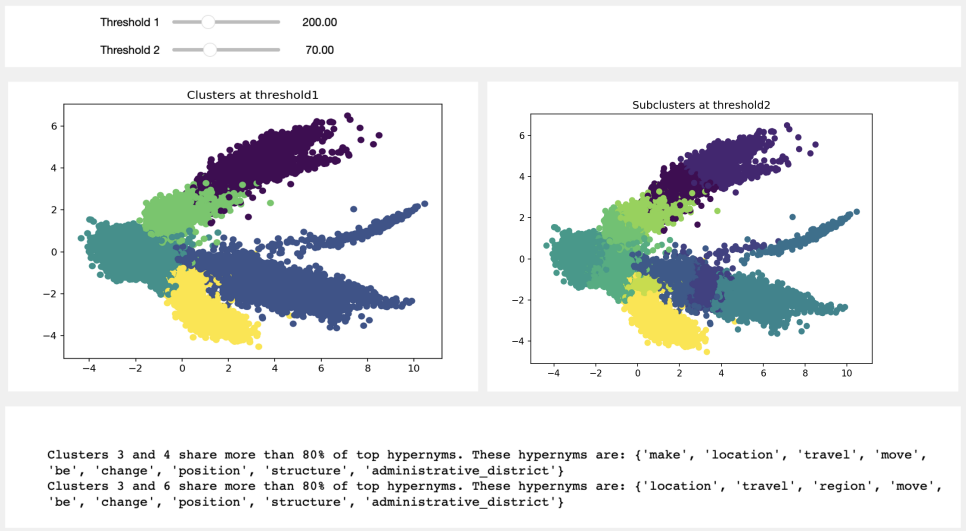


Figure 28: Dual Threshold Hierarchical Clustering: A visual representation of primary clustering (left) and nested subclustering (right) based on selected Threshold 1 and Threshold 2 values

Figure 28 provides a visual representation of this process, where Threshold 1 is set at 200 and Threshold 2 at 70. The algorithm then generates two embedding spaces, as depicted in the figure. On the left, the primary cluster numbers are displayed, while on the right, the corresponding subclusters are illustrated. Each subcluster is then methodically compared to identify similarities within the same primary cluster.

The tool calculates semantic node graphs for each subcluster, allowing for a comprehensive comparison. If the top words and top hypernyms for two subclusters within the same cluster share more than 80 percent similarity, these subclusters are highlighted

in the text output, along with the shared top words and hypernyms.

This analysis offers the user the ability to delve into different areas of the embedding space, fostering an understanding of how many subtopics can be discerned within the main topics. The concept of nested subcluster comparison elucidates both the underlying subtopics and the overarching theme of a group of texts.

Both the local analysis and nested hiearchical clustering approaches can be used together to have comprehensive topic detection and semantic analysis. By utilizing semantic node graphs, common words, and hypernyms, these tools offer a versatile and comprehensive platform for text analysis. Together, they form a coherent and powerful toolkit, enhancing the user's capacity to explore, interpret, and derive meaningful insights from complex textual data sets.

7 Conclusion

This master's thesis explores the challenge of detecting and classifying topics in large, unclassified text datasets. Conventional approaches, such as Term Frequency-Inverse Document Frequency (TF-IDF), have often fallen short in capturing the semantic depth of the textual data, leading to a need for more sophisticated methodologies.

Addressing this challenge, the study merges clustering and semantic analysis techniques to create a comprehensive topic detection algorithm. The foundation of the methodology lies in sentence embeddings, which have significantly advanced the understanding of text semantics. Using the pretrained model architecture, mpnet-base, these embeddings enable the transformation of textual data into a high-dimensional vector space, where each vector encapsulates the contextual meaning of the corresponding sentence.

However, operating in high-dimensional spaces introduces specific challenges. To address this, the study employs a neural network architecture, primarily designed to make the embeddings topic-aware. This not only reduces the dimensionality of the data but also ensures it is contextually relevant and primed for subsequent analysis stages.

The next phase of the methodology introduces an application of the Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) algorithm for text data clustering. When combined with Principal Component Analysis (PCA), the resultant toolset offers a method for topic detection. Not only does it reveal the overarching topics within the text corpus, but it also identifies subtopics, providing a granular understanding of the data landscape.

Another key contribution of the research is the development of interactive tools that allow users to fine-tune the analysis parameters. This feature, applicable for local and nested hierarchical clustering, adds a layer of flexibility and user-interaction to the topic detection process, enhancing its practical utility.

Despite these advancements, the study acknowledges certain limitations. The opacity of the neural network architecture and the methodology's reliance on the quality and diversity of the pretrained model present challenges that may impact the effectiveness of the approach.

Looking forward, several potential improvements have been identified:

- Expanding the Interactive Tools: Further development could include additional interactive tools for anomaly detection or emerging topic detection. Utilizing streaming datasets could enable real-time topic detection, enhancing the practical implications of the methodology.
- Foundational model for topic embeddings: Future research could also focus on developing foundational models for topic-specific embeddings. This avenue holds the potential to enhance the accuracy and specificity of topic detection, building upon the foundation of this thesis methodology.

To address the primary objective of this research, this study has underscored the limitations of conventional techniques and has brought to methodology that holistically addresses topic detection in extensive textual datasets. It has not only emphasized the importance of semantic understanding in topic detection but also showcased the necessity of flexible, user-adaptable solutions in a continually evolving data landscape.

Through the integration of different techniques and the acknowledgement of inherent challenges, this thesis provides a comprehensive approach to topic detection, emphasizing its relevance and applicability in the field.

In conclusion, the study employs a semantically-informed methodology for topic detection and offers user-friendly tools based on its methodologies.

References

[1] Hugging Face. Sentence transformers: all-mpnet-base-v2. Hugging Face Model Hub, 2021.

[2] Leland McInnes, John Healy, and Steve Astels. hdbscan: Hierarchical density based clustering. *The Journal of Open Source Software*, 2(11):205, 2017.

[3] Millman K.J. van der Walt S.J. et al. Harris, C.R. Array programming with numpy. *Nature*, 585(7825):357–362, September 2020.

[4] Wes McKinney et al. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010.

[5] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc., 2009.

[6] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. doi: 10.1109/MCSE.2007.55.

[7] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(85):2825–2830, 2011.

[8] Guido Van Rossum. *The Python Library Reference, release 3.8.2*. Python Software Foundation, 2020.

[9] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: Large-scale machine learning on heterogeneous systems. Machine Learning library, 2015.

[10] Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.

[11] Leipzig University and University of Mannheim. Dbpedia, 2007.

[12] Dbpedia classes. URL `https://www.kaggle.com/datasets/danofer/dbpedia-classes`.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2019.

[14] Zhen Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. XLNet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*, 2019.

[15] Haifeng Wang, Jiwei Li, Hua Wu, Eduard Hovy, and Yu Sun. Pre-trained language models and their applications. *Engineering*, 2022. ISSN 2095-8099. doi: https://doi.org/10.1016/j.eng.2022.04.024. URL `https://www.sciencedirect.com/science/article/pii/S2095809922006324`.

[16] Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. 2020.

[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPS)*, pages 5998–6008, 2017.

[18] Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. A survey of transformers. *AI Open*, 3:111–132, 2022. ISSN 2666-6510. doi: https://doi.org/10.1016/j.aiopen.2022.10.001. URL `https://www.sciencedirect.com/science/article/pii/S2666651022000146`.

[19] D. Rothman. *Transformers for Natural Language Processing: Build innovative deep neural network architectures for NLP with Python, PyTorch, TensorFlow, BERT, RoBERTa, and more.* Packt Publishing, 2021.

[20] S. Ravichandiran. *Getting Started with Google BERT: Build and train state-of-the-art natural language processing models using BERT.* Packt Publishing, 2021.

[21] Jay Alammar. The illustrated transformer, 2018. URL `https://jalammar.github.io/illustrated-transformer/`.

[22] Kirtikumar Girishbhai Pandya. Topic-aware approaches to natural language processing. Master's thesis, Heinrich Heine Universitat Düsseldorf, 2022.

[23] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning with Applications in R*. Springer, New York, NY, 2013.

[24] P. Berba. Understanding hdbscan and density-based clustering, 2020. URL `https://towardsdatascience.com/understanding-hdbscan-and-density-based-clustering-121dbee1320e`.

[25] P. J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.

[26] Wikipedia contributors. Silhouette (clustering), 2020. URL `https://en.wikipedia.org/wiki/Silhouette_(clustering)`.

[27] Patrick Esser, Robin Rombach, and Björn Ommer. A disentangling invertible interpretation network for explaining latent representations, 2020.

[28] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008, Oct 2008.

Appendices

Tables

Table 7: Work, Event, SportSeason and Topical Concept,Device,UnitOfWork Class Subclusters

Cluster No.	Cluster	Top Words	Top Hypernyms
18	Periodical Publications and Musical works	album, released, song, single, game, published, first, series, magazine	make, travel, time period, sound, pass, issue
16	Music	album, song, released, single, first, band, music, one, series	make, sound, time period, travel, pass, activity
17	Games and music	game, released, album, published, song, journal, play, first, single, series	activity, travel, play, diversion, sound, diversion
20	Congests, sport events and Elections	election, event, race, state, place,final, first, battle	travel, time period, contest,vpoint,position
21	Contest and Sport events	event, first, cup, race, final, tournament, match, championship, team	travel, time period, contest,point, group
19	Sport Season	season, team, football, league, game, first, record, th, division, coach	time period,weaken, toughen, end, travel, point
13	Music Genre	music, genre, rock, style, band, also, song, popular, new, pop	sound,activity, auditory communication, travel, punishment

Eidesstattliche Versicherung

(Affidavit)

Alp YALCIN
Name, Vorname
(surname, first name)

230888
Matrikelnummer
(student ID number)

☐ Bachelorarbeit
(Bachelor's thesis)

☒ Masterarbeit
(Master's thesis)

Titel
(Title)

Topic Detection in Natural Language Processing using Clustering and Semantic Analysis Techniques

Ich versichere hiermit an Eides statt, dass ich die vorliegende Abschlussarbeit mit dem oben genannten Titel selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.	I declare in lieu of oath that I have completed the present thesis with the above-mentioned title independently and without any unauthorized assistance. I have not used any other sources or aids than the ones listed and have documented quotations and paraphrases as such. The thesis in its current or similar version has not been submitted to an auditing institution before.

Dortmund / 01.09.2023
Ort, Datum
(place, date)

A. Yalcin
Unterschrift
(signature)

Belehrung:
Wer vorsätzlich gegen eine die Täuschung über Prüfungsleistungen betreffende Regelung einer Hochschulprüfungsordnung verstößt, handelt ordnungswidrig. Die Ordnungswidrigkeit kann mit einer Geldbuße von bis zu 50.000,00 € geahndet werden. Zuständige Verwaltungsbehörde für die Verfolgung und Ahndung von Ordnungswidrigkeiten ist der Kanzler/die Kanzlerin der Technischen Universität Dortmund. Im Falle eines mehrfachen oder sonstigen schwerwiegenden Täuschungsversuches kann der Prüfling zudem exmatrikuliert werden. (§ 63 Abs. 5 Hochschulgesetz - HG -).

Die Abgabe einer falschen Versicherung an Eides statt wird mit Freiheitsstrafe bis zu 3 Jahren oder mit Geldstrafe bestraft.

Die Technische Universität Dortmund wird ggf. elektronische Vergleichswerkzeuge (wie z.B. die Software „turnitin") zur Überprüfung von Ordnungswidrigkeiten in Prüfungsverfahren nutzen.

Die oben stehende Belehrung habe ich zur Kenntnis genommen:

Official notification:
Any person who intentionally breaches any regulation of university examination regulations relating to deception in examination performance is acting improperly. This offense can be punished with a fine of up to EUR 50,000.00. The competent administrative authority for the pursuit and prosecution of offenses of this type is the Chancellor of TU Dortmund University. In the case of multiple or other serious attempts at deception, the examinee can also be unenrolled, Section 63 (5) North Rhine-Westphalia Higher Education Act (*Hochschulgesetz, HG*).

The submission of a false affidavit will be punished with a prison sentence of up to three years or a fine.

As may be necessary, TU Dortmund University will make use of electronic plagiarism-prevention tools (e.g. the "turnitin" service) in order to monitor violations during the examination procedures.

I have taken note of the above official notification:*

Dortmund / 01.09.2023
Ort, Datum
(place, date)

A. Yalcin
Unterschrift
(signature)

***Please be aware that solely the German version of the affidavit ("Eidesstattliche Versicherung") for the Bachelor's/ Master's thesis is the official and legally binding version.**