INSTITUT FÜR INFORMATIK
Datenbanken und Informationssysteme

Universitätsstr. 1 D–40225 Düsseldorf

Heinrich Heine Universität Düsseldorf

Evaluating Semantic Search over an Index of Online Banner Ads

Rihab Al-Yasiri

Bachelorarbeit

Date of issue: 2023
Date of submission: 2024
Reviewers: Prof. Dr. Stefan Conrad
Prof. Dr. Michael Schöttner

Erklärung

Hiermit versichere ich, dass ich diese Bachelorarbeit selbstständig verfasst habe. Ich habe dazu keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Düsseldorf, den 2024

Rihab Al-Yasiri

Acknowledgements

I would like to express my deepest gratitude to everyone who supported me during the preparation of this thesis.

Special thanks to Dr. Matthäus Zloch for his role as supervisor, whose insightful feedback and constructive criticism were crucial to the refinement of this thesis.

Abstract

The goal of this thesis is to evaluate the influence of semantic search technology on search engine, specifically focusing on online banner ads, and comparing its effectiveness to traditional keyword-based searches using Elasticsearch. Semantic search technology aims to understand the semantics, or meaning of queries and advertisement documents, allowing for searches based on the context and meaning of the queries.

To achieve this, we will employ various sentence-transformer models, each trained on specific datasets, to convert sentences into vector representations known as embeddings. These embeddings are d-dimensional vectors comprised of floating-point numbers that capture the semantic essence of the words. We will compare the query embedding with all embeddings of the ads to calculate similarity using three types of similarity metrics: cosine similarity, FAISS similarity, and Euclidean distance. Our experiments will explore various combinations of models, similarity metrics, data preprocessing to ascertain their impact on semantic search performance. Additionally, we will enhance our approach by implementing feature engineering, including keyword extraction using the 'spaCy' library and extracting captions from advertisement images with another transformer model.

This thesis will also discuss how data cleaning influences the process of the model in capturing the semantic of the ads and identify which features most significantly contribute to the model's effectiveness.

For evaluation, we will select three specific queries relevant to our dataset, establish a ground truth, and label each query as true (relevant) or false (not relevant). As this involves a binary classification problem—where the positive class represents relevant ads and the negative class represents irrelevant ads—we will test our semantic search engine against the ground truth to assess its performance relative to a baseline.

In conclusion, we will evaluate the results of the semantic search using popular information retrieval performance metrics, such as precision and recall. Our findings indicate that semantic search significantly increases the number of relevant ads, from 10% to 90%, depending on the query's complexity and the performance compared to the baseline. The code for this thesis is written in Python and is available on Gitlab, along with the ground truth dataset.

Contents

Contents

1 Introduction

As the volume of data increases, search engines face greater challenges in retrieving information relevant to the queries entered. Information retrieval is a challenging subject; users, overwhelm with information through the search engine, seek the most relevant results and presenting irrelevant results can be very frustrating for them.

Search engines based on keyword matching are inefficient because they merely match words word-by-word, neglecting the context of the sentence (query). Additionally, keywords searching can encounter issues with ambiguous words, colloquial language, or misspelled words by the user, all of which can affect the performance of the searching algorithm. Researchers in both academia and industry have developed solutions to enhance the performance of search engines where they found stochastic approaches have been leveraged to improve the effectiveness of Information Retrieval (IR) systems. For example, genetic algorithms have been utilized to enhance IR effectiveness, reformulate queries, select optimal queries, and improve overall search efficiency. Additionally, other techniques, including Fuzzy algorithms, local context analysis, clustering, and ranking improvement strategies, have been applied to further refine and optimize search engine performance. **BPCDNP14**

Elasticsearch excels in full-text search for documents containing many words, like HTML websites, books, etc. However, it performs worse when the dataset does not contain long texts, as is the case with the text of online banner ads. While Elasticsearch performs well with full-text searching, its search capability was limited and did not retrieve relevant data, especially with words that have multiple meanings and whose meanings depend on the context of the sentence.

This limitation becomes evident when processing queries based on the mere presence of words in the documents. For instance, a naive search for "wein kaufen" return records containing either "wein", "kaufen," or both. However, this approach is insufficient for complex queries or terms with multiple meanings, often failing to yield satisfactory results. Additionally, users unfamiliar with specific jargon or those conducting exploratory searches may find the search engine's outcomes unsatisfactory due to its inability to interpret and analyze the intent behind a query, limiting its effectiveness in delivering relevant and comprehensive search results.

As shown in Figure 1, several examples of advertisements are presented, illustrating the limitations of keyword matching.

The examples (a) and (b) showcase some of the ads relevant to the query "Wein kaufen". However, as observed in the first example (a), it consists merely of a call-to-action and lacks detailed information that would aid a search engine in retrieving the document. On the other hand, example (b) is an ad for specific wine brand, demonstrating that without utilizing a model pre-trained to understand the context and significance of such information, it would be difficult to identify these as relevant ads for the query "Wein kaufen". In the third example (c), regarding the query "günstiger Mobilfunktarif", it is

(a) Wein kaufen (b) Wein kaufen (c) günstiger Mobilfunktarif

Figure 1: Examples of advertisements for specific queries, that will be used in this thesis.

evident that there are no words directly referring to the low price of the offer, making it challenging for a keyword-based search engine to match it accurately.

In the following sections, we detail how to address this issue and achieve better, more relevant results by evaluating semantic search against such challenges.

1.1 Objective of this thesis

The objective of this thesis is to address the limitations of search engines based on keywords by applying a semantic approach that fetch results based on the similarity of the context between the query and the ads.

Semantic search enhances search engines by enabling them to understand the context and meaning of queries, rather than relying solely on keyword matching. This approach improves the accuracy and user satisfaction in searching and can be implemented using technologies such as machine learning and artificial intelligence, will be demonstrated through a series of experiments and evaluations.

Semantic search aims to discern the semantic relationships between the query and the documents. It focuses on the user's intent and the meaning of the query to suggest relevant results.

1.2 Methodology

To achieve our goal, Figure 2 illustrates the comprehensive methodology for implementing semantic search on a search engine. The method involves converting advertisements (ads) and user's query into embeddings, where the embeddings encapsulate the semantic meaning of the advertisements as discussed in section 4.5. Then we compare their similarity in the vector space based on similarity metrics as discussed in section 4.6, which return the most relevant ads to the query. We employ several online banner ads indices from Elasticsearch for further research. Given the large volume of ads and the substantial amount of noisy data, which could undermine the effectiveness of our analysis, we have decided to initially, in step (1), prepare the data for further analysis by removing noisy data and focusing exclusively on the German language ads within a specified data

frame, adressing the challenge posed by the volume and complexity of online banner ads as discussed in section 3.

However, even after the data preparation step, our datasets are still not valid for analysis. In step (2), we construct a ground truth by labeling a subset of the data as relevant or not to each query (evaluation queries can be found in section 4.1), establishing a baseline for evaluation, as discussed in section 4.2.

Having established the ground truth, we can, in step (3), extract the feature set (inputs) from it in section 4.4, which be fed into the embedding model to generate vectors.

Subsequently, in step (4), we use a sentence transformer model to transform the sentences (feature sets or query) into d-dimensional embeddings all of which is discussed in section 4.5. Both query and documents will be converted into dimensional vectors in the space, and we apply different similarity metrics, such as cosine similarity and Euclidean distance, to calculate the similarities, as shown in step (5).

After calculating the similarities using the chosen algorithms, in section 5, we conduct several experiments to analyze how data preprocessing, different models, similarity metrics, and features (inputs) affect the relevance of the results. The focus of this thesis is not to identify the optimal experiment to integrate into the search engine but rather to analyze these experiments and infer answers to our research questions from them.

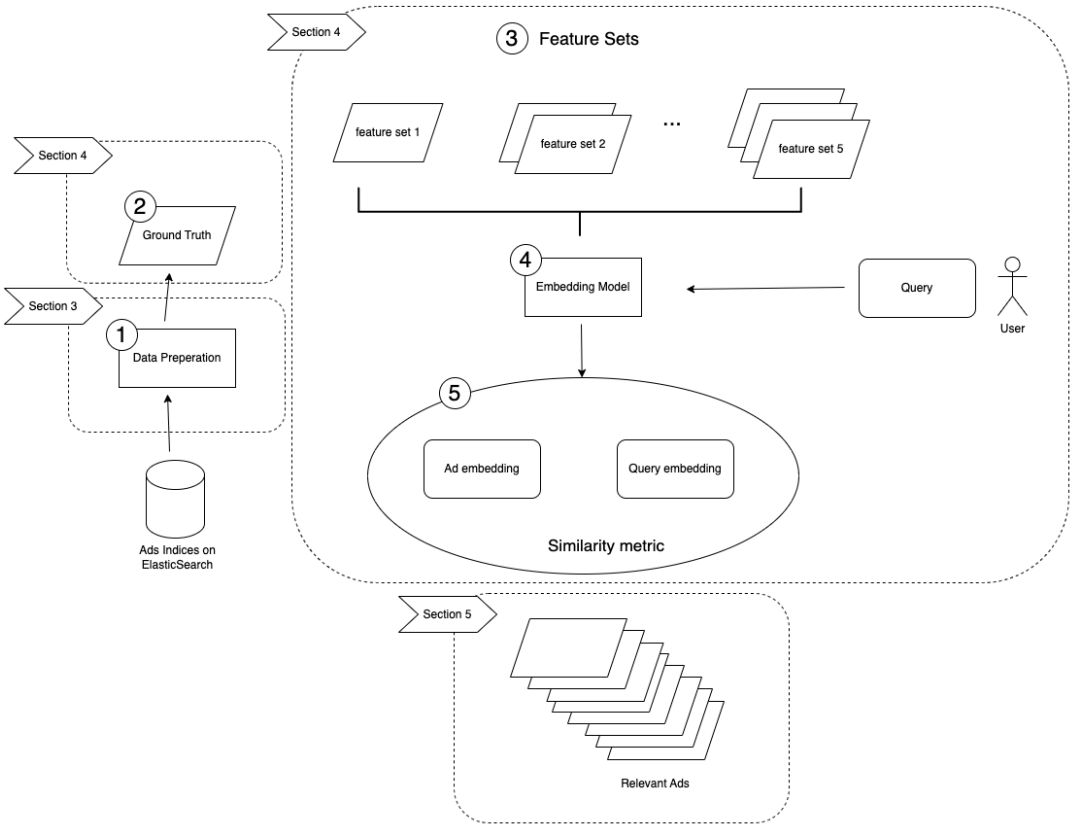


Figure 2: Methodology overview with semantic search approach.

2 Background of the Study

We provide an abstract overview of the background for the study, offering the reader insight into the methodologies used and how they work.

2.1 Natural Language Processing

"Natural Language Processing Definition is a theoretically motivated range of computational techniques for analyzing and representing naturally occurring texts at one or more levels of linguistic analysis for the purpose of achieving human-like language processing for a range of tasks or applications. " **LI01**

NLP (Natural Language Processing) has been used in several application use cases:

- **Translation**: One of the oldest applications, used to translate languages in text and voice formats.
- **Autocomplete**: Or sentence completion, which is used in combination with Machine Learning algorithms to predict the likelihood of the next word or sentence to complete the meaning.
- **Semantic-based search**: Refers to the search based not on keywords but on the meaning of the query, and this use case will be utilized in the thesis.

2.1.1 Tokenization

As we saw in the last section, NLP is used in several use cases, which involves working with natural language (human language). To process sentences in these languages, it is necessary to introduce the tokenization process.

Tokenization is the process of transforming sentence into tokens, where these tokens are not cut up exactly where the words start or end - tokens can include trailing spaces and even sub-words. **openaitokens**:

The number of tokens is the input limitation of the embedding model, which can process them all at once. This limit varies from model to model. The following code snippet demonstrates this with the help of NLTK, the Natural Language Toolkit. NLTK has a built-in function for text tokenization called `word_tokenize`, which takes text as input and returns a list of tokens to illustrate how tokenization works in practice:

```
1  from nltk.tokenize import word_tokenize
2  data = "Ich liebe Machine Learning"
3  tokens = word_tokenize(data)
```

The result will appear as follows: `['Ich', 'liebe', 'Machine', 'Learning']`.

2.2 Sentence Bidirectional Encoder Representations from Transformers (SBERT)

"Sentence-BERT (SBERT), a modification of the BERT network using siamese and triplet networks that is able to derive semantically meaningful sentence embeddings. This enables SBERT to be used for certain new tasks, which up-to-now were not applicable for BERT. These tasks include large-scale semantic similarity comparison, clustering, and information retrieval via semantic search." **NRI19**

SBERT takes a sentence as input and converts it into a fixed-length embedding. Sentences that are similar in context will be placed closer to each other in the vector space. With a similarity algorithm, we can calculate the semantic similarity between vectors.

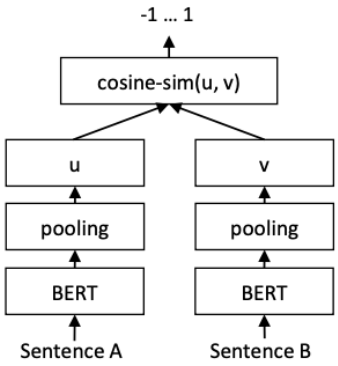


Figure 3: SBERT architecture (siamese network structure). **SBERT**

Figure 3 shows the Siamese network structure of SBERT, where two sentences, Sentence A and Sentence B, are passed as input into separate BERT models. The output (embedding) is then passed as input into a pooling layer to derive a fixed-length embedding for each. Both vectors (u and v) are compared with cosine similarity (or other similarity metric) at the end to obtain results that range from [-1,1].

2.2.1 Sentence Embedding

Embeddings are a specific type of vector representation of content or a query, created by machine learning models that capture the semantic meaning of text or representations of other content such as images. **Microsoft2023VectorSearch**

Sentence embedding is the mapping of a sentence into a d-dimensional vector space as discussed in section 4.5, where it appears as a list of floating-point numbers. These numbers are not random; rather, they are produced from the input multiplied by the model's learned weights. These numbers capture the semantics of the words.

To transform a sentence into an embedding, we pass a sentence such as "es gibt zwei Leute, die Fahrrad fahren auf einem Pfad in den Bergen" into a transformer model from HuggingFace, like the German Semantic Comparator model (which will be discussed

further in section 4.5.2). As output, we obtain a 1024-dimensional dense vector, as shown below:

$$\textbf{Embedding} = \begin{pmatrix} 0.123 & 0.234 & \cdots & 0.678 \\ 0.345 & 0.456 & \cdots & 0.789 \\ \vdots & \vdots & \ddots & \vdots \\ 0.901 & 0.012 & \cdots & 0.345 \end{pmatrix}$$

2.3 Elasticsearch

`"Elasticsearch is the distributed search and analytics engine at the heart of the Elastic Stack."` **EL24**

It is an open-source solution built on Apache Lucene, used to store, search, and analyze large datasets and retrieve data in real time, but it is not considered a replacement for traditional databases. It is primarily used for log or data analysis and full-text search.

Main concepts:

Distributed: Elasticsearch automatically distributes data across the active nodes in a cluster, enabling it to handle large volumes of data.

RESTful API: Interaction with Elasticsearch is conducted through its API, making it very easy to perform requests against it. It stores data as JSON documents in an index.

Full-Text Search: Elasticsearch excels at full-text search by applying data processing steps to the documents added to an index, such as tokenization, lowercasing, stop words removal, and stemming. Subsequently, it builds an inverted index that lists every unique word appearing in any document. When a search query is made, the query undergoes the same data processing steps to match the format of the terms stored in the inverted index. The search engine then uses the inverted index to compare lists of terms rather than scanning entire documents, utilizing various ranking scores (like TF-IDF or BM25) to determine relevance.

2.4 Precision and Recall

Precision and recall are essential metrics for evaluating information retrieval systems, particularly in classification tasks within the context of information retrieval, as discussed in this thesis. The precision and recall metrics can be derived from examining the contingency Table 1, which describes the performance of a classification model on a dataset for which the true classifications are known.

	Relevant	**Non-Relevant**
Retrieved	TP (True Positive)	FP (False Positive)
Not Retrieved	FN (False Negative)	TN (True Negative)

Table 1: Contingency table for evaluating classification model performance. **MAN08**

Precision is the metric that calculates the proportion of retrieved documents that are relevant, reflecting the accuracy of the retrieval.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

where:

- TP (True Positive): is the number of documents that correctly identified as relevant.
- FP (False Positive): is the number of documents that incorrectly identified as relevant.

Recall is the metric that calculates the proportion of relevant documents that have been retrieved over the total amount of relevant documents available, reflecting the completeness of the retrieval.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

where:

- TP (True Positive): is the number of documents that correctly identified as relevant.
- FN (False Negative): is the number of documents that relevant but not retrieved.

3 Data Preparation

The performance of a search engine based on semantic search is influenced by the quality of the data. Therefore, it is crucial to clean the data and filter out noisy data. As shown in Figure 3, we go through two phases. On the left side, from steps 1 to 5, we explain data cleaning, as detailed in Section 3.1, and on the right side, we explain data preprocessing steps, which will be detailed in Section 3.2.

3.1 Data Cleaning

Given that our data is collected via scraper bots, we observed that the quality of the data is lower compared to data gathered through human efforts. To address this, in step (1) we will be removing duplicate ads that contain the same content (body text), as they are merely duplicates that always yield the same vector, encapsulating the same meaning. Consequently, they do not contribute to assessing the performance of the search engine.

```
1   def remove_duplicates(ads):
2       unique_ads = set()
3       return [unique_ads.add(ad['body_text']) or ad for ad in ads if
4               ad.get('body_text', '') not in unique_ads]
```

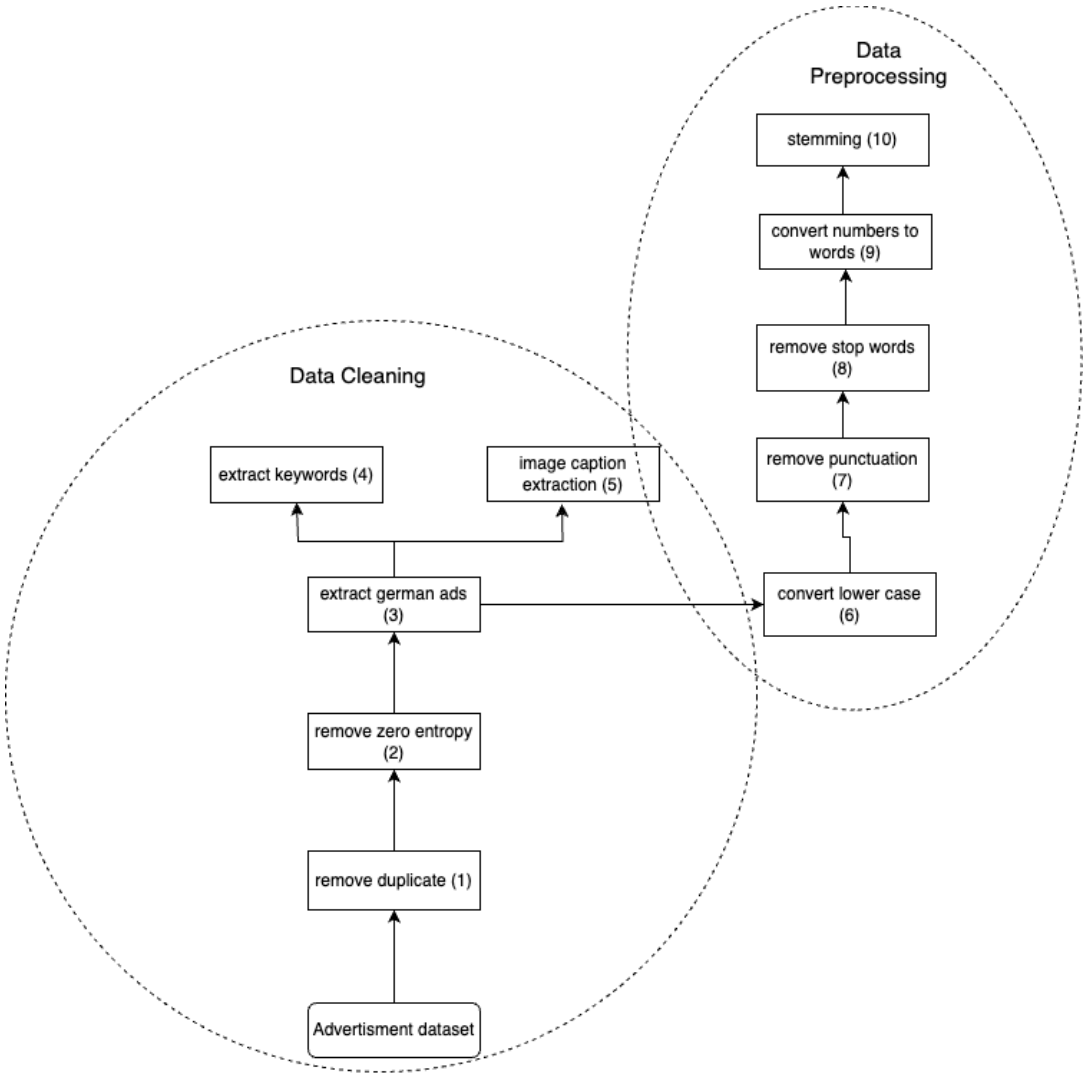


Figure 4: step-by-step illustration of the data preparation approach of this thesis.

In addition to that, we remove any empty text in the $body\_text$ of the ads. In step (2) ads with images that have low $screenshot\_entropy$ will be removed as well using a method called $has\_non\_zero\_entropy(ad)$ as images with lower entropy means images with lower pixels and because caption of the images are an important feature to feed into our HuggingFace model, which will be discussed in a later section, we decided to use data that contain images.

```
1   def has_non_zero_entropy(ad):
2       return ad.get('screenshot_entropy', 0) > 0
```

In this thesis, our focus will be solely on ads in the German language to simplify the process on one hand, and on the other hand, we will be using a HuggingFace model to transform the text into embeddings, that has been trained exclusively on a German dataset. This made deciding on a language a crucial decision in this thesis. In step (3) we detect just ads in german language with help of Python library for language detection ported from Google's language-detection.

In step (4), we extract keywords from the body text of the advertisement document with the help of "spaCy," an NLP library. This process will be part of feature set 4, which is discussed in section 4.4, to evaluate its effect on performance.

At the final step of data cleaning (5), we use the images from the advertisement. With the aid of a pre-trained model from Hugging Face, we extract captions from the images, which will be needed in the feature set 5.

3.2 Data Preprocessing

Data preprocessing involves transforming the data to prepare it for feeding into the model to obtain accurate vectors, and its impact on the results will be observed. This process includes several steps; in step (6), we convert all sentences to lowercase to normalize the sentences to just lower case.

Then in step (7), we remove all punctuation from the sentence, as they do not add meaning to the sentence.

In step (8), we remove all the stop words from a sentence because they do not impact the meaning of the sentence. Examples of stop words in the German language include $['der', 'das', 'die', 'mir', 'nach', 'jedes']$. The process in this step removing all the stop words found in a sentence with the help of a dictionary that contains all the stop words.

In step (9) we apply a method that converts all the numbers found in the sentence into words, with the help of a library designed to convert numbers into words in multiple languages.

In step (10) stemming is a method that returns each word in the sentence to its stem.

Preprocessing step	Result
start	Affiliate Marketing ist eine einfache Möglichkeit, um mit unserer Affiliate-Plattform online Geld zu verdienen.
6	affiliate marketing ist eine einfache möglichkeit, um mit unserer affiliate-plattform online geld zu verdienen.
7	affiliate marketing ist eine einfache möglichkeit um mit unserer affiliate plattform online geld zu verdienen
8	affiliate marketing einfache möglichkeit unserer affiliate plattform online geld verdienen
9	affiliate marketing einfache möglichkeit unserer affiliate plattform online geld verdienen
10	affili market einfach möglichkeit unser affili plattform onlin geld verdienen

Table 2: Evolution of text preprocessing for advertisment example.

After all the preprocessing steps, our data is finally ready for the further steps in the methodology.

4 Experimental Setup

In this section, we will go through all the steps necessary to set up our experiments. This will involve deciding which queries to use for evaluating the semantic search, building the ground truth for the evaluation, determining which features will serve as inputs, selecting the chosen models, and calculating the similarity of embeddings with different similarity metrics.

4.1 Evaluation Queries

Evaluation queries will be chosen based on the context of the advertisement in our dataset. After reading and analyzing a sufficient amount of ads, we have decided to select several queries—up to four—in various topics and complexities. These will be used further for the purpose of labeling the ads as relevant or not relevant based on the query. We have chosen sentences rather than individual words because complex sentences address the limitations of keyword match searching. The queries are listed in order of complexity:

- Schnäppchen Flüge
- Wein kaufen
- Kredit mit niedrigen Zinsen
- günstiger Mobilfunktarif

4.2 Building Ground Truth

In the context of machine learning and data science, ground truth is the dataset used to train and evaluate models, where in supervised learning, the model is fed with features (input) alongside labels and afterall evaluated by comparing results of unseen data with its label (prediction).

In this thesis, we build the ground truth to evaluate our experiments by comparing the prediction with the label. In section 5, we compare the results with baseline results to see if our various experiments have an impact on the efficiency of the search engine.

4.2.1 Online Banner Ads Dataset

Adbubble [1] is a platform designed for the analysis of advertising data, where users can search for a specific query regarding advertisment and receive a list of ads related to their query. The most crucial step in any machine learning project, where most of the time is invested, is understanding the data, which is our focus in this section. Online banner ads are datasets generated by scraper bots, each personalized through cookies for a specific sector and region, and the data stored in an Elasticsearch index with a name equivalent to its personalization.

For this thesis and the scope of the study, we received a database dump from Adbubble as an example below shows one advertisment JSON object:


```
{
    "_index": "ads_pet_food",
    "_type": "_doc",
    "_id": "7c010d16996944da_0_1",
    "_score": 1.0,
    "_source": {
      "frame_hrefs_parsed": [
        {"advertiser.domain_name": "coaching-akademie-berlin.ch"},
        {"advertiser.domain_name": "coaching-akademie-berlin.ch"},
        {"advertiser.domain_name": "coaching-akademie-berlin.ch"},
        {"advertiser.domain_name": "coaching-akademie-berlin.ch"},
        {"advertiser.domain_name": "coaching-akademie-berlin.ch"}
      ],
      "publisher_name": "https://standartnews.com/",
      "body_text": "Coaching Akademie in Zürich\nPraxisorientiert
  und akkreditiert\nCoaching Akademie Berlin\nWeitere Infos",
      "screenshot_entropy": 3.264813435956221,
      "screenshot": "0_1.png",
      "scraper_proxy_name": "Zurich",
      "repository": "/home/adbubble/ad-scraper/operations/adsdata/
2023-04-02/Pet_food/standartnews.com/7c010d16996944da",
      "hash_value": "5b96dcddd3da2aef",
      "day": "2023-04-02",
      "timestamp": "2023-04-02T04:31:03.975796"
  }
```


Table 3 presents the significance of each element within the advertisement document.

[1] Adbubble, dispaly ads monitoring platform https://adbubble.io (last access: 2024-03-18)

Key name	**Description**
_index	Generated by Elasticsearch, it refers to the name of the index where the document is stored.
_type	Generated by Elasticsearch, it refers to the document type, which by default is a document type.
_id	Generated by Elasticsearch, it refers to the unique identifier for the document.
_source	Generated by Elasticsearch, it contains the content of the advertisement.
frame_hrefs_parsed	An array of objects referring to all possible advertisers.
advertiser.domain_name	Refers to the domain name of the advertisement's advertiser.
publisher_name	Refers to the domain name where the advertisement was published.
body_text	Refers to the text of the advertisement.
screenshot_entropy	Refers to the amount of randomness or disorder in an image of the advertisement, where lower values towards 0 indicate images with just one pixel.
screenshot	Refers to the filename of the image belonging to that advertisement.
scraper_proxy_name	Refers to the name of the scraper proxy, where its value always refers to the city where the scraper operates.
repository	Refers to the path where the image is stored on the server.
days	Refers to the date when the data was collected.
timestamp	Refers to the precise date and time when the data was collected.

Table 3: Overview of elasticsearch document fields for advertisement data.

For this thesis, we will be running Elasticsearch as a containerized service on Docker, and our advertisements dataset will be stored in the volumes of the Container. Kibana, part of the Elastic Stack, is used to explore and visualize the data, and we use the Kibana image to provide a UI for interacting with our Elasticsearch service. Although we have a total of 31 indices, we be using only 5 distinct indices due to the large amount of data. The chosen indices are:

- fine art ads
- pet food ads
- pop music ads
- real estate ads
- motherhood ads

4.2.2 Labeling Process

The labeling process involves assigning each advertisement document a label that indicates its relevance to a specific query, which will be relied upon in the evaluation process.

After data cleaning and eliminating noisy data, the focus was narrowed to the timeframe between May and June (for year 2023) to simplify the process of building ground truth, as it consumes a lot of time and effort, we ended up with a total of 13,814 ads. To simplify the labeling of our advertisements dataset, we built a simple web app as shown in Figure 5 with a minimalistic UI. It displays the text of the ad and its image, as the images sometimes provide more information about the ads. The app includes radio buttons to choose between one of the possible queries (“wein kaufen”, “günstiger Mobilfunktarif”, "Kredit mit niedrigen Zinsen", “Schnäppchen Flüge”). An ad that is relevant to one of these queries is labeled its specific key as True; otherwise, it is labeled as False, based on human decision.

Process	**Number of ads**
total	19999+ ads
data preparation (mai & june)	13,814 ads
ground truth	1,425 ads

Table 4: Summary of advertisement processing and quantities

Our labeled dataset is imbalanced, with a predominance of non-relevant ads over relevant ones. This imbalance is due to the time-consuming nature of creating a more balanced dataset. The dataset comprises a total of 1,425 ads, as depicted in Figure 6. We exclude the query "Kredit mit niedrigen Zinsen" from the evaluation, because it is seasonally based and finding relevant ads for it proved to be too difficult. With our ground truth dataset established, we can utilize it in our experiments to evaluate the efficiency of the evaluation method, that will be discussed in section 4.8.

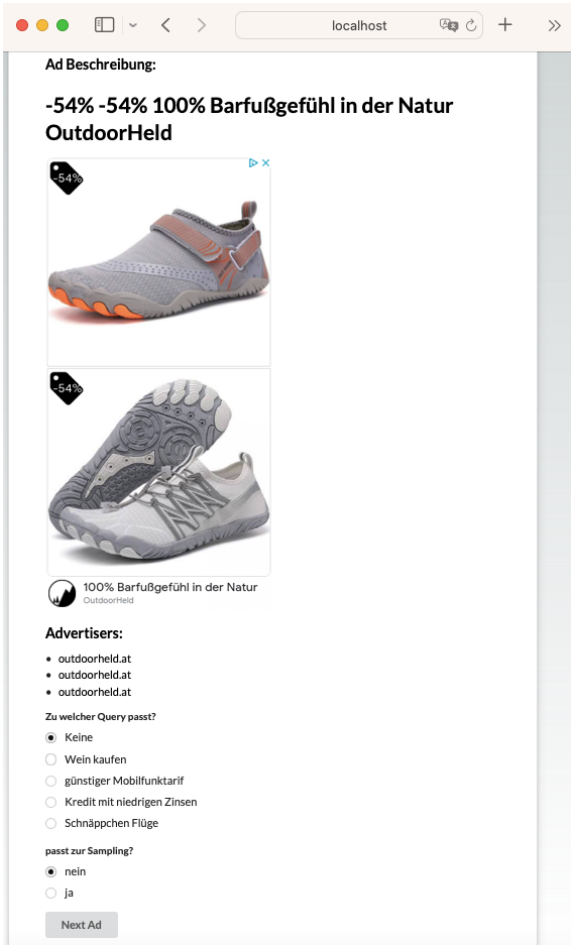


Figure 5: Screenshot of the web app that we developed for building the ground truth.

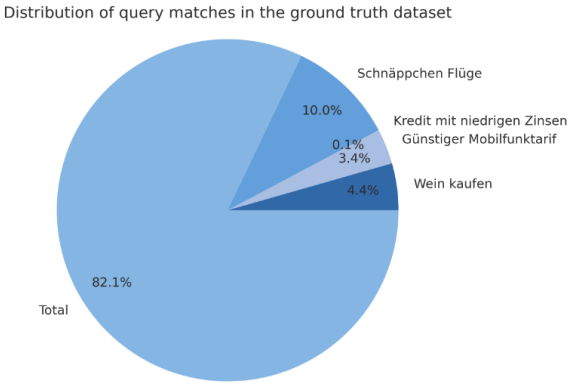


Figure 6: The pie chart displays the distribution of specific query-matching ads in our ground truth, contrasting with other ad types.

4.3 Experiments Overview

In this thesis, our primary focus is to evaluate semantic search within a search engine for the advertisement dataset and assess its efficiency. Our main objective is to examine various experiments to see how they affect the search engine. By doing so, we have created several experiments, as shown in the Figure 7, which detail all the possible experiments that can be applied to our evaluation method. This is how we can infer the experiment from Figure 7 based on parameter:

- Experiment 1: data preprocessing (Yes), German Semantic Comparator Model, cosine similarity and feature set 1.
- Experiment 2: data preprocessing (Yes), German Semantic Comparator Model, cosine similarity and feature set 2.
- ...
- Experiment 40: data preprocessing (No), MultiLang Efficient Embedder Model, euclidean distance and feature set 5.

This approach helps us gain a deeper understanding of the factors influencing search efficiency.

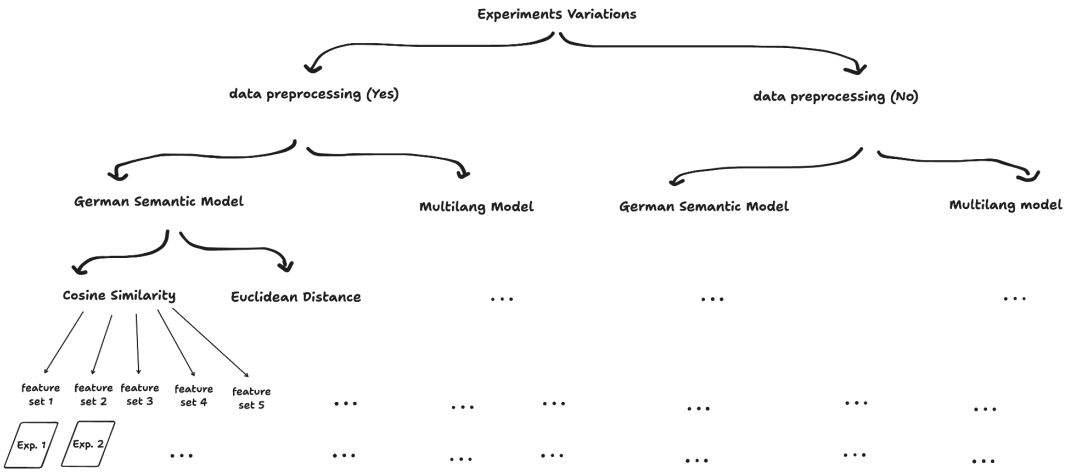


Figure 7: Flowchart of experimental variations within this thesis.

In total, we applied 40 experiments using the evaluation method for each query.

4.4 Feature Sets

In this section, we explore all the features that will serve as inputs for our experiments. Table 5 lists the selected features along with their descriptions and the feature engineering methods applied to them.

Feature No.	Name	Description
feature 1	body text	The main textual content of the advertisement.
feature 2	advertiser domain name	The web domain associated with the advertiser.
feature 3	image caption	The descriptive text that we infer with HuggingFace model from the image.
feature 4	keywords	Extracted keywords from the body text of the advertisment.
feature 5	combined description	Extracted from the description tag HTML of the advertiser Website.

Table 5: Description of features used in advertisement experiments

Table 6 will then detail the generation of a set of features from those selected for further analysis. The text of the ad in *feature set 1* is the minimum input required by the embedding model to generate an embedding that captures the semantic meaning of the ad.

In *feature set 2*, we add the advertiser's domain name, which often indicates the product's nature, for instance the domain name $nike.de$, further aiding the embedding model in understanding the product that it is about shoes.

In *feature set 3*, we incorporate image captions, utilizing the HuggingFace model $Salesforce/blip-image-captioning-large$, which takes an image as input and describes its content. Since the image description results were in English, we translated the text to German with the help of another HuggingFace model, called $Helsinki-NLP/opus-mt-en-de$. The describing images and translation process is time-consuming, so to streamline our work, we expanded our ground truth with new features for $image\_caption\_en$ and $image\_caption\_de$, avoiding repetition each time we run the evaluation method. We believe that the impact of *feature set 3* will be significant, especially for ads with minimal text, where the advertisement's message is conveyed through the image.

feature set 4 includes keywords extracted from the ad's body text using SpaCy, an open-source library for advanced Natural Language Processing (NLP) in Python, and stored as an array of strings.

In *feature set 5*, we add a combined description to the set, which is a description tag in HTML from the advertiser's domain name. Its content describes the advertiser.

Feature set no.	Features
feature set 1	body text
feature set 2	body text, advertiser domain name
feature set 3	body text, advertiser domain name, image caption
feature set 4	body text, advertiser domain name, image caption, keywords
feature set 5	body text, advertiser domain name, image caption, keywords, combined description

Table 6: Feature sets used in experiments

4.5 Language Models

HuggingFace Hub is an open-source data science and machine learning platform for sharing open-source models, datasets, and demo apps. For simplicity, we use a pre-trained model that has already been trained on a specific dataset to perform a specific task. In our case, models should be trained on semantic search tasks, and there is no need to train the model from scratch due to time and cost considerations. After analyzing different pre-trained models and comparing their results on the semantic search task, we ended up choosing just two models that have a significant impact on the results.

4.5.1 MultiLang Efficient Embedder Model

`"This is a sentence-transformers model: It maps sentences and paragraphs to a 768 dimensional dense vector space and can be used for tasks like clustering or semantic search."` **RNGI20**

The model, developed by the Sentence Transformer Team, was not trained on one specific language but on up to 50 languages (including German), and was specifically trained for phrase identification. In other words, it was trained to understand the semantic similarity between sentences, even if they use different words. This model encodes a sentence (text) into a 768 dimensional dense vector that represents its semantic meaning.

It is a multilingual model capable of understanding different languages, making it versatile for applications with datasets in various languages. It is available on HuggingFace Hub and can be easily integrated into our pipeline by using $sentence\_transformers$, a Python framework for state-of-the-art sentence, text, and image embeddings.

In practice, we import the model from the $sentence\_transformers$ framework.

```
1       from sentence_transformers import SentenceTransformer, models
2       word_embedding_model = models.Transformer('sentence-transformers/
3       paraphrase-multilingual-mpnet-base-v2')

4       pooling_model = models.Pooling(word_embedding_model
5                                              .get_word_embedding_dimension())
```

Using pooling, it generates from a variable sized sentence embedding a fixed sized sentence embedding. This layer also allows to use the CLS token if it is returned by the underlying word embedding model. You can concatenate multiple poolings together.

```
1       model_nli = SentenceTransformer(modules=[word_embedding_model, pooling_model])
```

As an example, we call $model\_nli.encode(ad)$ on one of our ads to generate its vector:

```
vector = model_nli.encode("Fachmagazine empfehlen dieses
                                        Weinabo Korkenfreunde Mehr erfahren")
```

The output will be a 768-dimensional dense vector:

$$\mathbf{v} = \begin{bmatrix} v_1 & v_2 & v_3 & \dots & v_{768} \end{bmatrix}$$

To evaluate its efficiency as discussed in section 5, we compare its results with those of the second HuggingFace model, which will be explained in the next section.

4.5.2 German Semantic Comparator Model

This model build on Sentence Transformer, encodes sentences into 1024-dimensional dense vectors. It is primarily used for tasks such as clustering or semantic search.

`"The base model is trained by deepset. The dataset was published / translated by Philip May. The model was fine-tuned by Aaron Chibb."` **german_semantic_sts_v2**

This model is trained exclusively on the German language and is specifically tailored for the Semantic Textual Similarity (STS) benchmark dataset. To utilize this model, we follow the same procedural steps as with the previous model, as demonstrated in the following code snippet:

```
word_embedding_model = models.Transformer("aari1995/German_Semantic_STS_V2")
pooling_model = models.Pooling(word_embedding_model.get_word_embedding_dimension())
model_nli = SentenceTransformer(modules=[word_embedding_model,pooling_model])
```

After setup, $model\_nli$ is ready to generate embeddings for sentences. Using the $encode()$ method, we can create the vector:

```
vector = model_nli.encode("Fachmagazine empfehlen dieses Weinabo Korkenfreunde
                                                        Mehr erfahren")
```

The results differ in dimensionality

$$\mathbf{v} = \begin{bmatrix} v_1 & v_2 & v_3 & \dots & v_{1024} \end{bmatrix}$$

This model features a larger dimensionality of 1024, compared to the previous model's 768. The dimensionality of a vector is crucial as it represents the model's capacity to encode information, which can lead to a better understanding of the advertisement data. However, it also impacts computational requirements, where models with higher dimensionality may require more time and resources for computation. In section 5, we evaluate both models and examine how factors such as the dimensionality of the model, the language it was trained on, and the dataset it was trained on affect the results in information retrieval.

4.6 Similarity metrics

Most algorithms used to calculate the similarity between vectors, or in general, items (such as images, text, etc.), share the concept of representing the vectors as points in a

high-dimensional space and calculating the distance between them. In this thesis, we introduce three distinct metrics:

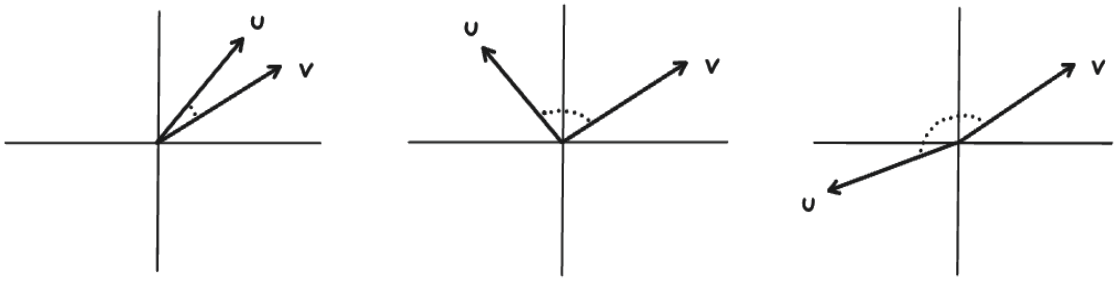
4.6.1 Cosine Similarity

Cosine similarity is a metric used to calculate the similarity between two vectors in space. It is calculated as follows:

$$\text{cosine similarity}(\mathbf{u}, \mathbf{v}) = \cos(\theta) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|}$$

where u and v are vectors, and $||u||$ and $||v||$ represent the magnitude of vectors u and v, respectively. As a result, the values range between -1 and +1. **MI23**.

- A cosine similarity towards +1 indicates that u and v are more similar in meaning.
- A cosine similarity towards 0 indicates that u and v are more dissimilar in meaning.
- A cosine similarity towards -1 indicates that u and v are opposite in meaning.



(a) Similar (b) Unrelated (c) Opposite

Figure 8: Graph illustrating the calculation of cosine similarity in space, where u, v are vectors.

4.6.2 Faiss Similarity

Faiss stands for Facebook AI Similarity Search; it is a library developed by Meta. **FA23**.

Faiss has the ability to index vectors, including both binary index vectors and floating-point index vectors, depending on the requirements of the application, which can choose between them. **FA24**.

We will be focusing on the floating-point index vector because the accuracy of the similarity search in our use case is important, and we will not be focusing on the efficiency of storage and computation.

The Faiss similarity search algorithm, under the hood, uses Euclidean distance, which will be discussed in the next section.

4.6.3 Euclidean distance

It is a distance metric that count as the most common method for calculating the distance between two objects (such as text, images, etc.) on the space.

Euclidean distance is calculated using the Minkowski distance formula, where $p = 2$. **SH19**.

$$\left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{\frac{1}{p}} \qquad \text{(Minkowski distance)}$$

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \qquad \text{(Euclidean distance)}$$

where x_i and y_i are vectors in n-dimensional space.

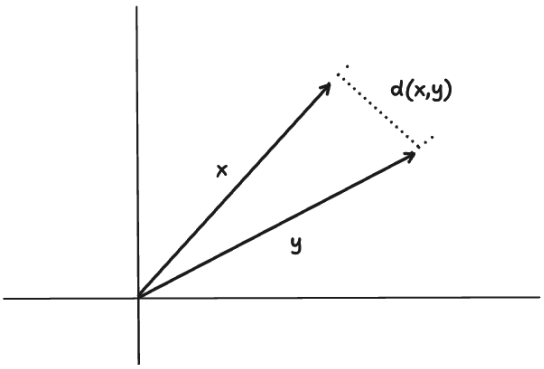


Figure 9: Graph illustrating the calculation of euclidean distance in space. Where x, y are vectors.

4.7 Similarity Calculation

We begin with cosine similarity, for which the sentence_transformers framework provides a built-in method, $cos\_sim(tensor\_1, tensor\_2)$. This method accepts two tensors as arguments; a tensor is a data structure that represents vectors in higher dimensions. It then calculates the angle between the two vectors and returns the similarity value, towards 1 means similar and towards 0 means dissimilar.

Table 7 presents several examples illustrating the similarities between queries and advertisements. For instance, in the first row, there is a similarity score of approximately 0.8 (80%) between the query "Wein kaufen" and the ad "TOP Weine auf VINONIA.com Wein online direkt Abhof kaufen! VINONIA OG by Taboola Sponsored Link". This score suggests a high degree of relevance, as the advertisement indeed pertains to buying wine. Conversely, in the second row, the advertisement "sportausbildung.com Sportausbil-

dung bei flexyfit WEITERE INFOS" bears little relevance to the query, leading to a drop in the similarity score to 0.49. This pattern is consistent across the other examples.

Query	Advertisment	Similarity value
Wein kaufen	TOP Weine auf VINONIA.com Wein online direkt Abhof kaufen! VINONIA OG by Taboola Sponsored Link	0.8067922592163086
Wein kaufen	sportausbildung.com Sportausbildung bei flexyfit WEITERE INFOS	0.4998849034309387
günstiger Mobilfunktarif	Hol dir die BLAU Allnet XL Flat jetzt mit 10 GB LTE für nur 7,99 € im Monat! blau.de	0.805473804473877
günstiger Mobilfunktarif	iPhone 14 für 79,95 € statt 179,95 € Jetzt sichern Bis zu 200 € sparen	0.6311082243919373

Table 7: Calculating the cosine similarity between several advertisements and different queries.

We apply the algorithm to the entire ground truth dataset so we can plot the results. This will help us better understand the relevance of advertisements in relation to the query and vice versa. In the plots 10 and 11, we focus on the query "Wein kaufen." The x-axis will represent the similarity values obtained from the cosine similarity algorithm, ranging from 0 to 1. The y-axis will display the labels of the advertisements (relevant/not-relevant) using random numbers, where positive numbers indicate relevance and negative numbers indicate irrelevance. It's important to note that the position of an advertisement on the y-axis has no intrinsic meaning; it serves only to separate the ads for clearer visualization. However, the x-axis is crucial as it determines the similarity to the query.

On the left side, we utilize Experiment 25, this experiment involves no data preprocessing and employs the German Semantic Comparator model along with cosine similarity, incorporating feature set 5. On the right side, we use Experiment 21 , which also involves no data preprocessing and uses the German Semantic Comparator model and cosine similarity, but focuses solely on feature set 1.

In the plot on the left side, the green circles, which represent relevant ads, are closer to the red circle (representing the query) and are positioned higher on the y-axis because they are labeled as true for "wein kaufen." However, some relevant ads are significantly farther from the query. Upon analysis using a Pandas DataFrame, it was found that these include ambiguous ads, most of the intent of which is captured in images. The text on these images falls outside the scope of this thesis. These ads are not explicitly clear and transparent about buying wine but are related to a platform that hosts an online store for purchasing wines. In contrast, on the plot on the right side, if we consider only similarities greater than 0.7, we find fewer green circle points (relevant ads) appearing on the scatter plot. This indicates that incorporating with more features can help a model discover additional patterns and make better predictions and shows that feature engineering is crucial in helping the model grasp the meaning of the ads and retrieve more relevant ads for a specific query. However, input with more features don't always pro-

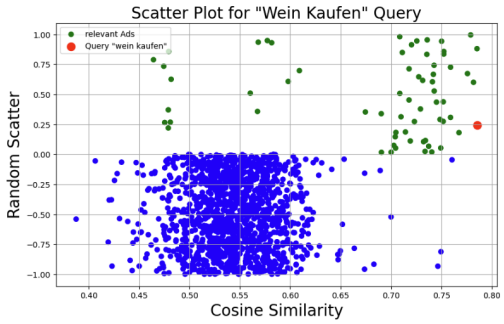


Figure 10: Experiment 25 involves no data preprocessing, German Semantic model, cosine similarity and feature set 5

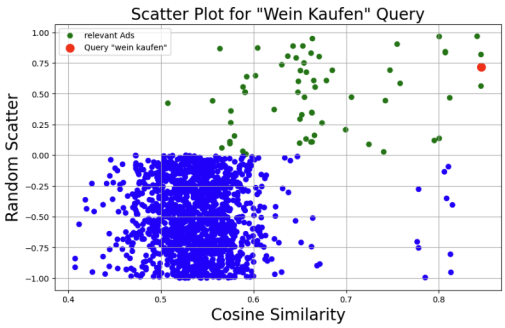


Figure 11: Experiment 21 involves no data preprocessing, German Semantic model, cosine similarity and feature set 1

duce models that make better predictions because some features might have no causal relationship to the label.

We apply the same process using Euclidean Distance for similarity measurement and as shown in plots 12 and 13. Euclidean distance calculates the distance between two vectors in space; therefore, the smaller the value (distance), the more similar the ads are to the query.

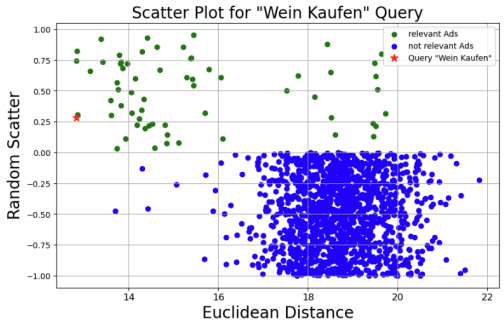


Figure 12: Experiment 30 involves no data preprocessing, German Semantic model, euclidean distance and feature set 5

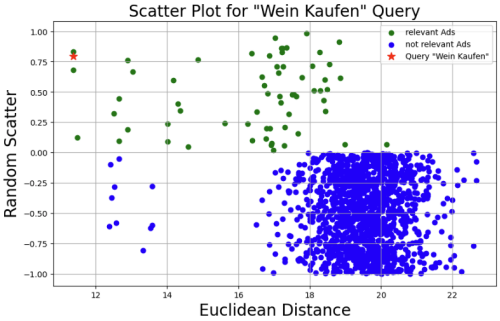


Figure 13: Experiment 26 involves no data preprocessing, German Semantic model, euclidean distance and feature set 1

The plot corresponds to the query "Wein kaufen." In the left plot, we observed that incorporating more features to enrich the advertisement's meaning enables better model understanding. In contrast, the right plot shows the query circle relatively far from many relevant ads. In the next section we will discuss the procedure of evaluation method that will be used to calculate the results of all the experiments.

4.8 Evaluation method

We simulate a search engine by writing an evaluation method. The parameters of the search engine are as follows:

- $query$: a string representing the search sentence.
- $data$: a list of dictionaries containing all the necessary features of an ad (Ground Truth).
- top_k: the number of top results to return (e.g., 10, 25).
- $data\_cleaning$: a boolean where $True$ indicates preprocessing is applied, and $False$ indicates no preprocessing.
- $algorithm$: specifies the similarity measurement to use, which can be:
  - $cosine\_similarity$
  - $faiss\_similarity$
  - $euclidean\_distance$
- $model$: specifies the model to be used for comparison, which can be:
  - $German\_Semantic\_Comparator$
  - $MultiLang\_Efficient\_Embedder$
- $features$: a list of strings representing the keys of the features to be used, such as:
  - "body_text"
  - "frame_hrefs_parsed"
  - "image_caption_de"
  - "keywords"
  - "combined_description"
- $reverse$: a boolean indicating the sorting order.
  - For the Cosine Similarity algorithm, $reverse$ is set to $True$ because the higher values (towards 1) indicate more similarity.
  - For the Faiss and Euclidean Distance algorithms, $reverse$ is set to $False$ because lower distance values indicate more similarity.

General evaluation method procedure:

1. Initialize an empty list to store similarities.
2. Normalize the query if data cleaning is required, then encode it using the model.
3. For each ad in the dataset:

(a) Extract relevant text features from the ad.

(b) If data cleaning is required, normalize the features text, then encode it using the model.

(c) Calculate the similarity between the ad's features and the query.

(d) Add the similarity score and ad to the list of similarities.

4. Sort the list of similarities in descending order if reverse is true, otherwise in ascending order.

5. Return the top k most similar ads with their similarity scores.

The evaluation method will compare the query with the dataset based on the previously mentioned parameters and return the number of relevant ads determined by the top_k value set as an argument.

5 Experimental Results

In this section, we will examine the outcomes of the experiments conducted for this thesis and analyze them based on various criteria such as the impact of data preprocessing, model type, similarity metrics, and feature sets.

Initially, upon comparing the two similarity metrics—cosine similarity and Euclidean distance—we observed that both metrics retrieved nearly identical relevant advertisements when applied to the German Semantic Comparator model. Nevertheless, with the alternative model, cosine similarity proved to be more effective, retrieving a greater number of pertinent ads, as illustrated in Figure 14. Stemming from this observation, we opted to utilize cosine similarity in our subsequent analyses of section 5.1, 5.2 and 5.3.

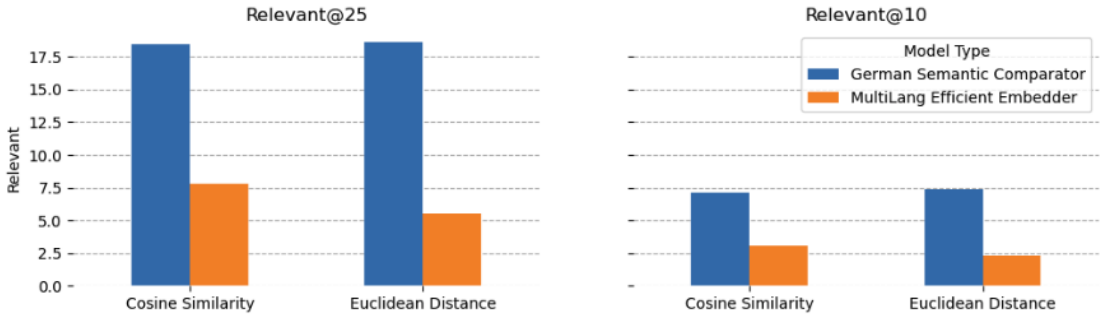


Figure 14: Demonstrating the effect of similarity metric on the accuracy of semantic search results

5.1 Impact of Data Preprocessing on Relevant Ads

Here, we aim to delve into more detail to illustrate the performance impact of data preprocessing on model types used for semantic search. In our experiments, we classify data

preprocessing into two categories: "Yes" for instances where data preprocessing has been applied, and "No" for instances where it has been omitted. Consequently, we aggregated the data according to these categories to calculate the average number of relevant ads accurately predicted by our evaluation method, as compared to the labels in the ground truth. Figure 15 presents a bar plot of this data, where the x-axis denotes the data preprocessing status ("Yes" or "No") and the y-axis represents the average number of accurately predicted relevant ads as it shows with "No" data preprocessing the model better perform in retrieving ads.

Notably, the differences become significantly pronounced with an increase in the value of k. Data preprocessing techniques, such as stemming, removing stop words, and tokenization, may contribute to the loss of semantic meaning. This is particularly critical in the context of an advertisement dataset, where ad texts are generally concise, averaging around 18 words, after analysis with help of pandas.

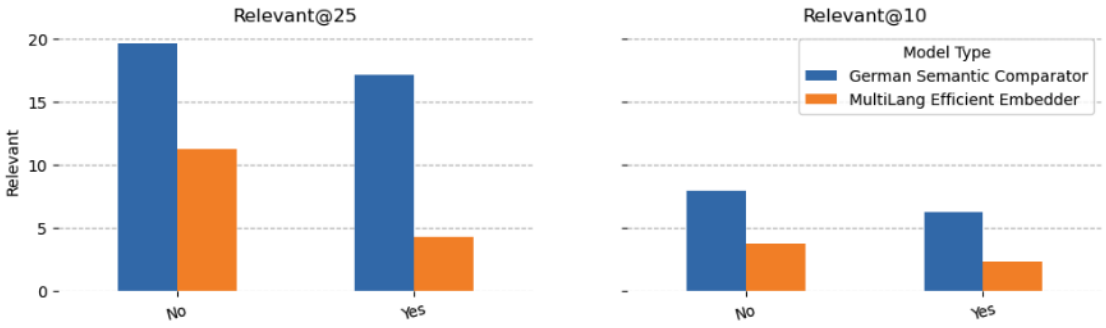


Figure 15: Demonstrating the effect of data preprocessing on the accuracy of semantic search results

5.2 Impact of Model Type on Relevant Ads

In this section, we delve deeper into the impact of model selection on the predictions made by the search engine for our evaluation queries, comparing these results against a baseline. Figure 16 illustrates that the German Semantic Comparator model significantly surpasses the performance of the MultiLang Efficient Embedder model. This superiority can be attributed to the German Semantic Comparator model's specific training on the German language, with a particular focus on Semantic Textual Similarity (STS) tasks.

Furthermore, the model employs a 1024-dimensional dense vector for its representations, unlike the MultiLang Efficient Embedder model, which uses a 768-dimensional dense vector. This increase in dimensionality allows for the retention of more semantic information within the embeddings, thereby enhancing the model's ability to comprehend the context of advertisements, independent of the text length.

Comparing the German Semantic Comparator to the baseline, we notice that for the query "Schnäppchen Flüge," both our model and baseline perform similarly, as it is the easiest query in our list of evaluation queries. However, for other queries such as "Wein kaufen" and "günstiger Mobilfunktarif," our model achieves much better results compared to the baseline. This demonstrates the limitation of our baseline in handling more

complex queries to retrieve relevant ads.

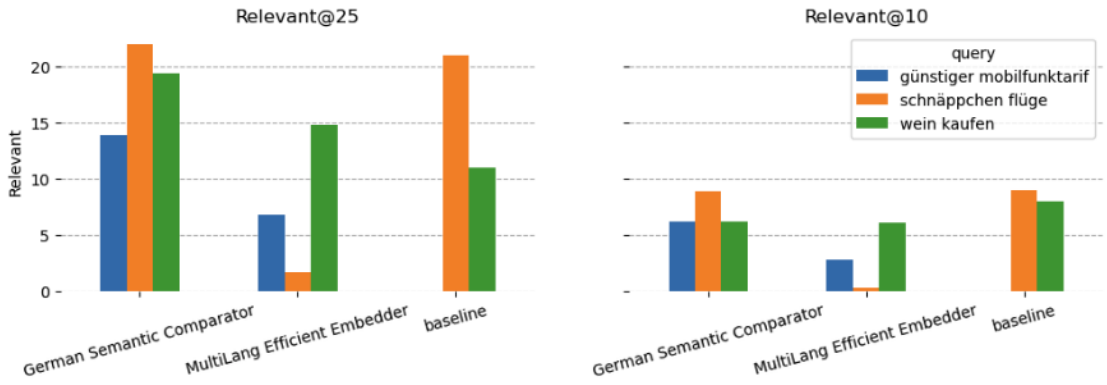


Figure 16: Demonstrating the effect of model type against baseline on the accuracy of semantic search results

5.3 Impact of Features on Relevant Ads

Features (inputs) are pivotal in machine learning projects for enhancing the predictive performance of models. As discussed in Section 4.4, where we introduced the features of the ads, we undertake feature selection to identify the most pertinent features to input into the embedding model, thereby improving its context absorption. Selecting suitable features can boost the embedding model's predictive accuracy, while including irrelevant or redundant features may decrease its efficiency. Such superfluous features can cause confusion in understanding the semantic meaning of sentences, in addition to increasing complexity and inference time.

Figure 17 illustrates that Feature Set 1, being the minimal data fed to our model, shows the least performance, as evident from the plot. As we incrementally introduce Feature Sets 2 and 3, the number of ads predicted as relevant increases, averaging around 20 ads. These feature sets contain significant information that enables the model to better process contextual data.

Conversely, the performance drops significantly with Feature Set 4 after adding keywords from the advertisement text. We hypothesize that this decline is due to redundant information leading to confusion in the model's processing of semantic features.

Feature Set 5 shows an improvement again as it includes description from the advertiser's website, which we anticipated would capture more semantic details about the advertisement. However, it does not surpass the performance of Feature Set 3. This limitation is attributed to the model's capacity to process a finite number of tokens at a time; exceeding this number results in the truncation of input data, a detail that falls outside the scope of this thesis.

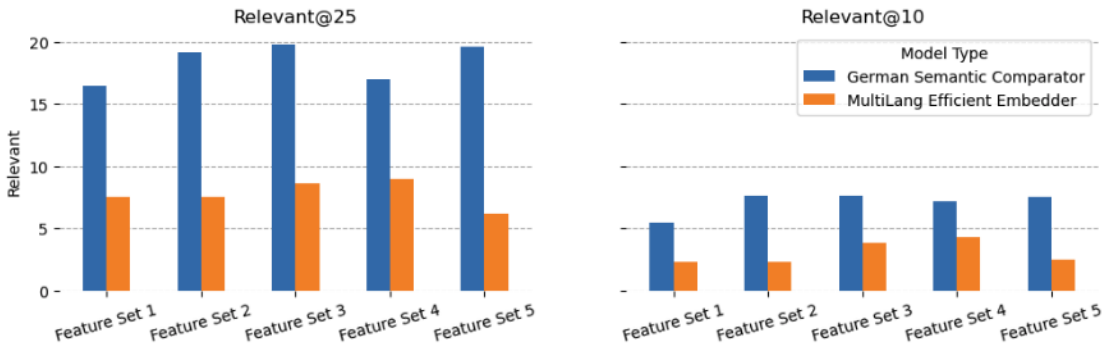


Figure 17: Demonstrating the effect of feature set on the accuracy of semantic search results

5.4 Average Number of Relevant Ads in Experiments

In Section 4, we calculated the number of relevant ads for a specific query for $k = 10$ and $k = 25$ across all experiments on the search engine. In Table 8, we present the average number of relevant results based on specific criteria across all experiments, where Model 1 and Model 2 refer to the German Semantic Comparator and MultiLang Efficient Embedder respectively, and Metric 1 and Metric 2 refer to cosine similarity and Euclidean distance, respectively.

K	With preprocessing	Without preprocessing	Model 1	Model 2	Metric 1	Metric 2
10	5.05	6.6	6.45	5.2	6.15	5.5
25	13.2	18.35	19.1	12.45	17.1	14.45

Table 8: Average of relevant results for query "Wein kaufen"

Calculating the averages, as shown in Table 8, provides more insight into how the experiments impact the efficiency of the search engine. From the data, it is observed that the use of the German Semantic Comparator model and cosine similarity, without applying data preprocessing, significantly affects the search engine's efficiency.

After finishing the examination of the experiments's performance, we aim to evaluate that performance using precision and recall metrics, which will be discussed in detail in the next section. Finally, we compare the values of precision and recall for the semantic search with those of a baseline system (Elasticsearch).

5.5 Precision and Recall Results

The interpretation of precision is that the higher the value, the more accurate the results retrieved. On the other hand, a higher recall means the system retrieves most of the relevant results. However, focusing primarily on recall can lead to the inclusion of more irrelevant results, as the system tries to ensure it doesn't miss any relevant ones. There is a trade-off between precision and recall, and increasing precision can decrease recall,

and vice versa. The decision on which metric to focus on always depends on the use case of the application.

Precision is crucial for us because recall depends on the total amount of relevant ads, which varies across different queries. Therefore, we cannot directly compare recall values for different queries with each other.

To elaborate further, let's calculate precision for one of the experiments. We can use experiment no. 1 for the query "günstiger Mobilfunktarif," where our threshold $k = 10$ (the threshold is the number of ads that we observe). The total number of ads in the ground truth is 1,425, of which only 48 ads are relevant to the query and 1,377 ads are not relevant. In Table 9, we consider the values of the confusion matrix as predicted by the simulated search engine based on the criteria of experiment no. 1 to calculate precision.

	Relevant	**Non-relevant**
Retrived	TP (4)	FP (6)
Non-retrived	FN (44)	TN (1377)

Table 9: Confusion Matrix for experiment no. 1 for query "günstiger Mobilfunktarif"

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{4}{4+6} = 0.4 = 40\%$$

A precision of 40% means that 40% of the ads retrieved by that experiment are relevant. In the next section, we examine the precision metric for further experiments and compare them with the baseline (Elasticsearch) to evaluate accuracy on both sides.

5.6 Compare Semantic Search based against baseline search engine

As discussed earlier, the evaluation method retrieved a higher average of relevant ads when we used no data preprocessing, the German Semantic Comparator model, and cosine similarity. We calculate precision and recall for different thresholds, $k = 10$ and $k = 25$.

Query	**Total no. of Relevant in db**	**Relevant no. of 25**	**Relevant no. of 10**	**Precision at k = 10**	**Recall at k = 10**	**Precision at k = 25**	**Recall at k = 25**
Wein kaufen	**63**	22	9	0.9	0.14	0.88	0.34
günstiger Mobilfunktarif	**48**	19	9	0.9	0.06	0.76	0.13
Schnäppchen Flüge	**143**	25	10	1	0.06	1	0.17

Table 10: Calculation of precision and recall on experiment 37, no data preprocessing, German Semantic Comparator, cosine similarity, feature_set_4

Afterward, we calculate the precision and recall for our baseline search engine with Elasticsearch, importantly using the same ground truth dataset. We index it in Elasticsearch

to ensure a fair comparison, as shown in Table 11.

Query	Total no. of Relevant in db	Relevant no. of 25	Relevant no. of 10	Precision at k = 10	Recall at k = 10	Precision at k = 25	Recall at k = 25
Wein kaufen	**63**	11	8	0.8	0.12	0.44	0.17
günstiger Mobilfunktarif	**48**	0	0	0	0	0	0
Schnäppchen Flüge	**143**	21	9	0.9	0.14	0.84	0.33

Table 11: Calculation of precision and recall on baseline (ElasticSearch).

As observed from Table 11 compared to Table 10, for some queries like "Schnäppchen Flüge" or "Wein kaufen," the performance is quite good. However, it struggles with more complex queries such as "günstiger Mobilfunktarif," which rely more on understanding the context than on comparing keywords, as is the case with the baseline. On the other hand, the semantic search achieves an accuracy > 0.9 when $k = 10$ and an accuracy > 0.76 when $k = 25$.

6 Conclusion

Our initial aim was to apply evaluation method based on semantic search over an index of online banner ads and evaluate it against various experiments to examine the impact of different criteria, such as data preprocessing, various SBERT models, different similarity metrics, and various features. The main idea was to use an SBERT model, trained on datasets for semantic similarity and clustering tasks, to feed it with features (inputs) to generate a vector (embedding). This embedding is then compared with the embedding of the query using a similarity algorithm to return the ads with the highest similarities.

Throughout this thesis, we observed how different experiments affected semantic search. We noted that data preprocessing techniques (such as tokenization, stemming, etc.) reduced the relevance of ads compared to experiments where no data preprocessing was applied. This suggests that data preprocessing can sometimes result in the loss of sentence meaning. Moreover, the choice of model was crucial, as it significantly impacted the ability to capture semantics. In some cases, the MultiLang Efficient Embedder struggled to process the semantics of the ads compared to the German Semantic Comparator, which was trained specifically on the German language and tasks like semantic search and clustering. It generates a more dimensional embedding with 1024 dimensions, whereas the MultiLang Efficient Embedder produces only a 768-dimensional embedding, which may not capture the full semantic meaning of the sentences as effectively.

Semantic search has enhanced the traditional keyword-matching search engine by retrieving more relevant ads, increasing accuracy by up to 10%, 90% depending on the threshold and query complexity.

The evaluation, including building a ground truth, was time-consuming and required significant effort to find related queries for the dataset and label the dataset with

relevant/not-relevant for each query. For this reaseon, we were limited in the queries and the number of ads used in the evaluation section, which prevented us from evaluating our semantic search on a larger dataset of ads and more complex queries to examine if the semantic search had any biases.

One potential improvement could have been to translate the ad dataset into the English language and work on SBERT models trained on English, given the abundance of models available for English compared to German. This approach could have overcome the limitation of being restricted to specific models on HuggingFace. We could also have compared sentence transformers with higher dimensions than the chosen models, as the dimensionality of the embedding is crucial in capturing semantic meaning and language complexity. However, we were constrained by the availability of models trained specifically on the German language and designed for semantic search and clustering tasks.

Nevertheless, we found that semantic search engines have a unique ability to handle more complex queries and retrieve more relevant results than simple keyword matching in particular for our dataset with short texts like online banner ad texts. However, a hybrid solution implementing both approaches could potentially achieve greater accuracy in the search engine.

List of Figures

List of Tables