Czech Technical University in Prague

Faculty of Electrical Engineering
Department of Cybernetics
Czech Institute of Informatics, Robotics and Cybernetics

From FastText to Transformer Models, and their Application in Retrieval-Augmented Generation

Bachelor's Thesis

Yauheni Zviazdou

Prague, May 2024

Study programme: Open Informatics
Branch of study: Artificial Intelligence and Computer Science

Supervisor: Ing. Jan Šedivý, CSc.

Acknowledgments

I would like to express my sincere gratitude to my supervisor, Ing. Jan Šedivý, CSc., for his invaluable advice and unwavering support throughout the past year. His guidance and expertise were instrumental in the completion of this research.

I would like to express my sincere gratitude to my colleague, Bc. Adam Jirkovský, for his invaluable contributions to this research project. Adam's expertise in the field and the code he developed were instrumental in facilitating the completion of this work. His guidance regarding the CIIRC cluster and assistance with the OpenAI API were also immensely helpful.

Furthermore, I extend my heartfelt thanks to my family: Ina Zviazdova, Viachaslau Zviazdou, Maryia Zviazdova, Natallia Pankova, Inessa Davtyan. Their unwavering moral and financial support, along with their constant encouragement, were a source of strength throughout my academic journey.

Finally, I would like to thank my friends, Vsevolod Tiemnohorov, Viachaslau Radzeuski, and Stsiapan Belaushka, for their unwavering support during this challenging yet rewarding experience.

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Zviazdou Yauheni**	Personal ID number:	**507333**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Cybernetics**		
Study program:	**Open Informatics**		
Specialisation:	**Artificial Intelligence and Computer Science**		

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

From FastText to Transformer Models, and their Application in Retrieval-Augmented Generation

Bachelor’s thesis title in Czech:

Od FastText k Transformer modelům a jejich aplikace v Retrieval-Augmented generování

Guidelines:

Review the representation of words, sentences, and paragraphs, progressing from traditional methods like FastText to advanced transformer-based models such as BERT. The primary focus is to evaluate selected representations using analogy tests and confusion matrix. Use the UPV corpus set for evaluation.
In the second part of the study, emphasis will shift towards selecting optimal representations for Retrieval-Augmented Generation (RAG) algorithms. The investigation will determine the most efficient embeddings and optimal text chunk size for question-answering tasks, particularly in the context of natural language answers generation from technical manuals. Conduct a comprehensive evaluation with a particular focus on suggesting an optimal representation model that balances factuality and CPU requirements.

Bibliography / sources:

[1] FastText documentation and tutorials, https://fasttext.cc/
[2] LangChain documentation, https://js.langchain.com/docs
[3] Word2Vec tutorials, https://www.tensorflow.org/text/tutorials/word2vec

Name and workplace of bachelor’s thesis supervisor:

Ing. Jan Šedivý, CSc. Big Data and Cloud Computing CIIRC

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

Date of bachelor’s thesis assignment: **02.01.2024** Deadline for bachelor thesis submission: **24.05.2024**

Assignment valid until: **21.09.2025**

Ing. Jan Šedivý, CSc. Supervisor’s signature	prof. Dr. Ing. Jan Kybic Head of department’s signature	prof. Mgr. Petr Páta, Ph.D. Dean’s signature

III. Assignment receipt

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

.

Date of assignment receipt	Student’s signature

Declaration

I declare that presented work was developed independently, and that I have listed all sources of information used within, in accordance with the Methodical instructions for observing ethical principles in preparation of university theses. I acknowledge the use of AI tools for rephrasing and grammatical correction.

Date ....

Abstract

This thesis examines the evolution of text representation methods, starting from traditional techniques like FastText and advancing to sophisticated transformer-based models such as Bidirectional Encoder Representations from Transformers (BERT). The study evaluates these representations through analogy tests and confusion matrix analysis, utilizing the UPV corpus set for comprehensive assessment.

In the latter part of the research, the focus shifts to optimizing text representations for Retrieval-Augmented Generation (RAG) algorithms. The investigation aims to identify the most effective embeddings and determine the optimal text chunk size for Question Answering (QA) tasks, particularly within the realm of generating natural language answers from technical manuals. A thorough evaluation is conducted to recommend an optimal representation model that strikes a balance between factual accuracy and computational efficiency.

Keywords Natural Language Processing (NLP), Word Embedding, Transformers, FastText, RAG, QA, Semantic Textual Similarity (STS)

Abstrakt

Tato práce zkoumá vývoj metod reprezentace textu, od tradičních technik jako FastText až po sofistikované modely založené na transformátorech, jako je Bidirectional Encoder Representations from Transformers (BERT). Studie hodnotí tyto reprezentace prostřednictvím testů analogie a analýzy matic záměn, přičemž využívá korpus UPV pro komplexní posouzení.

V pozdější části výzkumu se pozornost přesouvá k optimalizaci reprezentací textu pro algoritmy Retrieval-Augmented Generation (RAG). Výzkum si klade za cíl identifikovat nejúčinnější vektory a určit optimální velikost textových bloků pro úkoly Question Answering (QA), zejména v oblasti generování odpovědí v přirozeném jazyce z technických manuálů. Provádí se důkladné hodnocení s cílem doporučit optimální model reprezentace, který vyvažuje faktickou přesnost a výpočetní efektivitu.

Klíčová slova Zpracování přirozeného jazyka, Word Embedding, Transformátory, FastText, RAG, QA, Sémantická Podobnost Textu

Abbreviations

NLP Natural Language Processing

RAG Retrieval-Augmented Generation

STS Semantic Textual Similarity

QA Question Answering

ML Machine Learning

NN Neural Network

LLM Large Language Model

TF-IDF Term Frequency-Inverse Document Frequency

BoW Bag-of-Words

CBOW Continuous Bag-of-Words

GloVe Global vectors

BERT Bidirectional Encoder Representations from Transformers

mBERT Multilingual Bidirectional Encoder Representations from Transformers

mE5 Multilingual E5

LaBSE Language-Agnostic BERT Sentence Embedding

XLM-R XLM-Roberta

BGE BAAI General Embeddings

GTE General Text Embedding

MLM Masked Language Modeling

NSP Next Sentence Prediction

SimCSE Simple Contrastive Learning of Sentence Embeddings

RetroMAE Retrieval-oriented Language Models Via Masked Auto-Encoder

MTEB Massive Text Embedding Benchmark

Contents

1 Introduction

1.1 Text representation

The human language, with its nuances and complexities, presents a significant challenge for machines to understand. Natural Language Processing (NLP) bridges this gap, and at its core lies the critical concept of text representation. This process acts as a translator, bridging the gap between the richness of text and the numerical language that machines understand. By effectively capturing the meaning within words and their relationships, text representation empowers NLP models to leverage capabilities of the Machine Learning (ML). From sentiment analysis to machine translation, this ability to represent meaning fuels the advancements in NLP, enabling machines to interact with and decipher human language with ever-increasing accuracy.

1.2 Evolution of text representation methods

NLP has undergone a significant transformation in its approach to text representation. Early methods, such as one-hot encoding (e.g. Bag-of-Words (BoW) [38], Term Frequency-Inverse Document Frequency (TF-IDF) [39]), while simple to implement, suffered from limitations in efficiency due to dimensionality and sparsity issues.

Word embedding techniques (e.g., Word2Vec, Global vectors (GloVe), FastText) offered a significant improvement by capturing semantic relationships between words through high-dimensional word vectors. However, these techniques primarily focused on local context within a limited window, hindering their ability to capture complex relationships within sentences or documents.

The emergence of deep learning architectures, particularly transformer-based models like Bidirectional Encoder Representations from Transformers (BERT), revolutionized the field of text representation. These models allows to not only understand the meaning of individual words but also consider their interaction and context within a sentence or document.

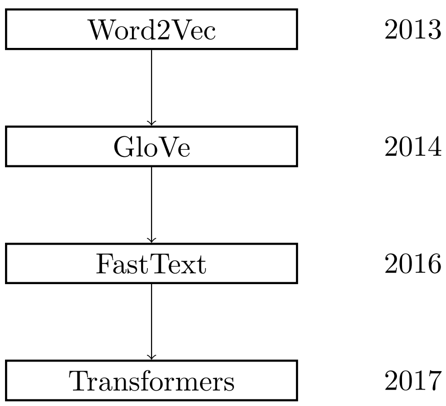


Figure 1: Evolution of the text representation methods.

1.3 Research objective

This research aims to evaluate the effectiveness of various word, sentence, and paragraph representations for their subsequent application in Retrieval-Augmented Generation (RAG) algorithms, with a specific focus on the domain of technical Question Answering (QA).

2 Literature Review

2.1 Traditional word embedding methods

Word2Vec

Word2Vec [36] is algorithm that generates word embedding using information about target word (context). Word2Vec uses Neural Network (NN) and ML techniques to generate word embedding for every word in vocabulary during training. As NN architecture are used Continuous Bag-of-Words (CBOW) and Skip-gram, Fig. 2.

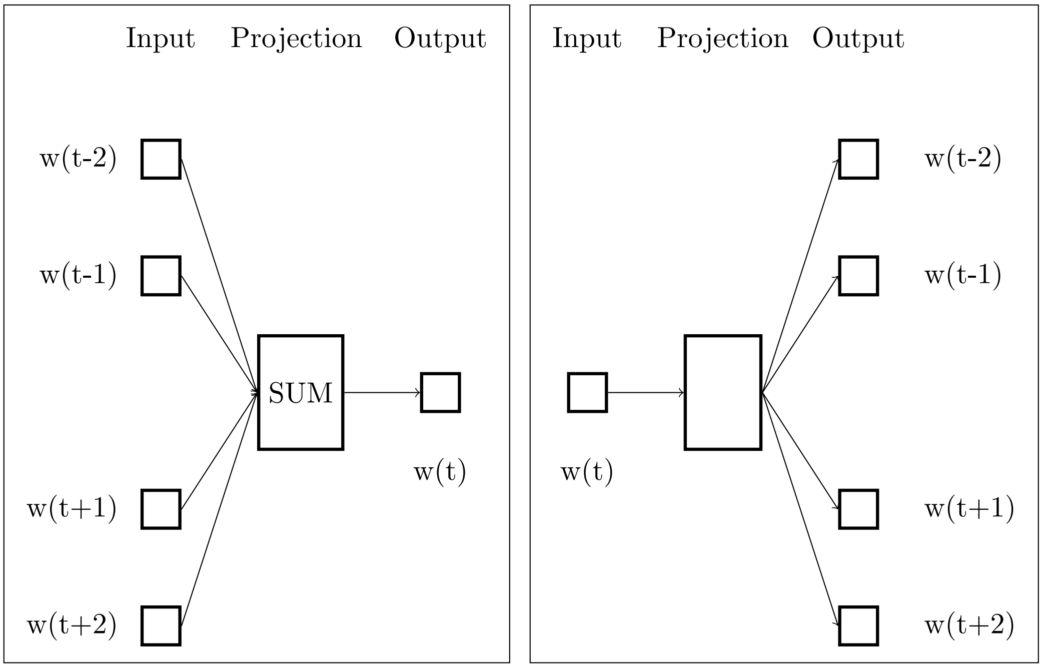


Figure 2: CBOW and Skip-gram schemes respectively

Due to its algorithmic simplicity and efficiency, Word2Vec has established itself as a strong baseline for numerous NLP tasks. Compared to more recent and complex models, Word2Vec requires minimal hyperparameter tuning, making it a relatively straightforward approach.

However, it is important to acknowledge that Word2Vec has limitations. These include its inability to capture **global information** within a document, its challenges in effectively handling **morphologically rich languages** (languages with many word variations), and its lack of awareness of the **broader context** beyond a limited window of surrounding words.

Global vectors (GloVe)

GloVe [35] leverages the co-occurrence statistics of words within a corpus to learn vector representations. This approach involves constructing a co-occurrence matrix, where each entry reflects the frequency of two words appearing together within a predefined window size. This matrix essentially captures the relative importance of various word pairings.

A core principle of GloVe lies in the notion that word vectors should effectively encode the ratios between co-occurrence probabilities of words. By analyzing these ratios, GloVe can identify semantic relationships between words. This is achieved by factorizing the co-occurrence matrix into a lower-dimensional space, allowing for efficient representation and manipulation of word meanings.

To optimize the learned word embeddings, GloVe employs a weighted least squares objective function. This function aims to minimize the discrepancy between the dot product of two word vectors and the logarithm of their co-occurrence probability. Through iterative adjustments of the word vectors, GloVe converges on a solution that yields the desired word embeddings.

FastText

FastText [33] utilizes similar NN architectures as Word2Vec, namely CBOW and Skip-gram, but applies them to character n-grams (subwords) instead of entire words. This decomposition allows FastText to represent a word's meaning by considering its constituent subword components. Consequently, FastText offers advantages in two key areas:

- **Rare Word Embeddings**: Unlike Word2Vec, which struggles with words appearing infrequently in the training data, FastText can construct meaningful representations for rare words. By leveraging known subwords, FastText can represent unseen words, making it particularly valuable for working with large and diverse datasets.
- **Handling Morphologically Rich Languages**: Languages with complex morphology, where words are formed through prefixes and suffixes, often pose challenges for Word2Vec. FastText overcomes this limitation by capturing the shared subwords between derived words and their root forms. This allows FastText to represent the inherent relationships between words in these languages, leading to more accurate NLP tasks.

However, it's important to acknowledge that FastText also has limitations:

- **Context Insensitivity**: Similar to Word2Vec, FastText embeddings do not inherently capture the order or context in which words appear within a sentence. This can be a drawback for tasks like sentiment analysis or machine translation, where word order and context are crucial for accurate interpretation.
- **Limited Long-Range Dependency Capture**: While subwords allow FastText to capture local context, it might not effectively capture long-range dependencies within sentences. This can be a disadvantage for tasks requiring analysis of complex sentence structures, where understanding the relationships between words across larger distances is important.

2.2 Transformer-based models

Transformer models [13] underpin powerful NLP models like BERT. A key advantage is their self-attention mechanism, which assigns importance to words based on context, not just position. This enables efficient parallel processing of entire sentences. Architecture of transformers visualized in Fig. 3

BERT [26] builds on transformers with pre-training on a massive text corpus. Masked Language Modeling (MLM) and Next Sentence Prediction (NSP) further enhance BERT's capabilities, fostering deep contextual understanding and grasp of sentence relationships. MLM injects a deeper understanding of context into BERT by requiring it to predict masked words

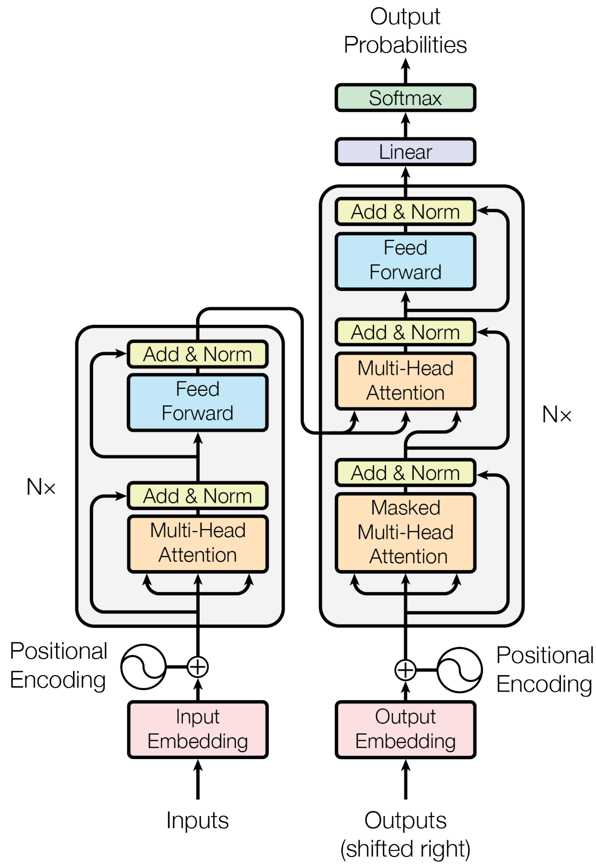


Figure 3: The Transformer - model architecture.

within a sentence. Through this process, BERT learns the relationships between words and their meaning based on the surrounding context. This goes beyond simple memorization - it allows BERT to grasp the nuances of language and handle even unseen words. NSP, on the other hand, strengthens BERT's ability to understand the flow and connection between sentences. During training, BERT is presented with sentence pairs and tasked with determining if the second sentence logically follows the first. By tackling this objective, BERT develops a grasp of sentence relationships, enabling it to analyze and process text that unfolds across multiple sentences, like news articles or conversations.

These strengths make transformers, particularly BERT, well-suited for NLP tasks. Their advantage lies in capturing contextual understanding, leading to richer text representations and superior comprehension of semantic relationships.

Furthermore, BERT excels in transfer learning, readily adapting to various tasks (sentiment analysis, QA) with minimal modifications. Additionally, efficiency and speed are benefits due to parallel processing and pre-training.

Transformer models effectiveness is validated by state-of-the-art performance across NLP benchmarks. Finally, BERT's robustness allows it to handle nuances in text without significant performance degradation.

2.3 Methods of text representations evaluation

Analogy Tests

Analogy tests, as demonstrated in the seminal Word2Vec paper [36], are a widely used method for assessing the quality of text representations, particularly word embeddings. These tests evaluate whether the semantic relationships between words are effectively captured and preserved within the vector space employed by the model.

A typical analogy test question follows the format "A is to B as C is to D," where A, B, C, and D represent words. For instance, the question "man is to king as woman is to queen" probes the model's understanding of gender relations. If the word embeddings are of high quality, performing the vector operation vector(king) - vector(man) + vector(woman) should result in a vector that closely resembles vector(queen). This outcome indicates that the model has successfully learned the analogous relationship between "man" and "king" and "woman" and "queen."

Confusion matrix

Confusion matrices are a widely used tool for evaluating classification algorithms, and they can be adapted to assess text representations in tasks such as word sense disambiguation, part-of-speech tagging, or sentiment analysis. A confusion matrix is a table that describes the performance of a classification model by comparing predicted and actual labels.

2.4 Retrieval-Augmented Generation (RAG)

RAG [20] is an advanced NLP framework that combines the strengths of retrieval-based and generation-based models to produce high-quality, contextually relevant text based on provided document (web-page etc.), instruction and query (question).

Architecture of RAG

The RAG algorithm leverages a two-stage approach for answer generation: retrieval and generation. Both stages rely heavily on the chosen text representation technique.

- **Retrieval**: In the initial phase, the algorithm extracts relevant information from the document and splits it into manageable chunks. These chunks are then fed into text representation models, which convert them into a format suitable for efficient retrieval. This process results in encoded representations of the information, which are then stored within a vector database. During the retrieval phase, RAG utilizes the same text representation model to encode the user's query (question). Subsequently, it searches the vector database and identifies the top-K most relevant passages based on their encoded representations.
- **Generation**: The K retrieved passages identified in the information retrieval phase serve as crucial contextual information for the Large Language Model (LLM) within the RAG system. By providing this context alongside the user's question and any additional instructions, the LLM is empowered to generate a comprehensive and informative answer.

Architecture of the RAG algorithm is shown on the Fig. 4

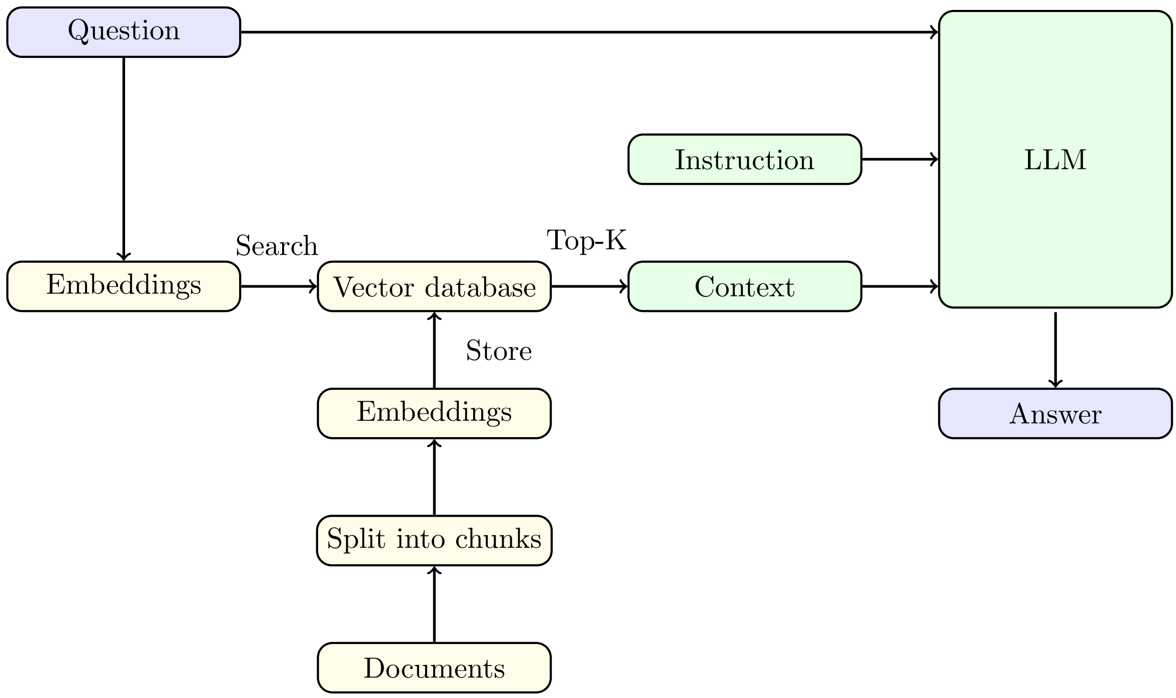


Figure 4: Retrieval-Augmented Generation architecture.

Factors influencing RAG performance

The performance of RAG systems can be influenced by several key factors. Two important aspects are:

- **Embedding Model Selection**: Previous work [1] suggests that the choice of embedding model significantly impacts RAG performance. Different embedding models offer varying strengths in capturing semantic relationships within text data. Selecting the most suitable model depends on the specific task and dataset.
- **Document Chunking Size**: Another factor influencing RAG performance is the size of the document chunks used for retrieval [41]. Splitting documents into smaller chunks can potentially improve retrieval efficiency. However, excessively small chunks may lead to a loss of context and hinder the RAG system's ability to generate coherent and relevant text. Finding the optimal chunking size requires careful consideration of the task and available computational resources.

3 Methodology

3.1 Embedding methods evaluation process

This work specifically targets the evaluation of word, sentence, and paragraph representation methods on datasets in the Czech language. This focus on Czech allows for a deeper understanding of how these methods perform in a language with specific characteristics, such as a rich inflectional morphology and the presence of diacritics.

Text Data Preparation

In certain foreign languages, a common issue arises when individuals incorrectly write words by omitting diacritics or altering letters, Fig. 5.

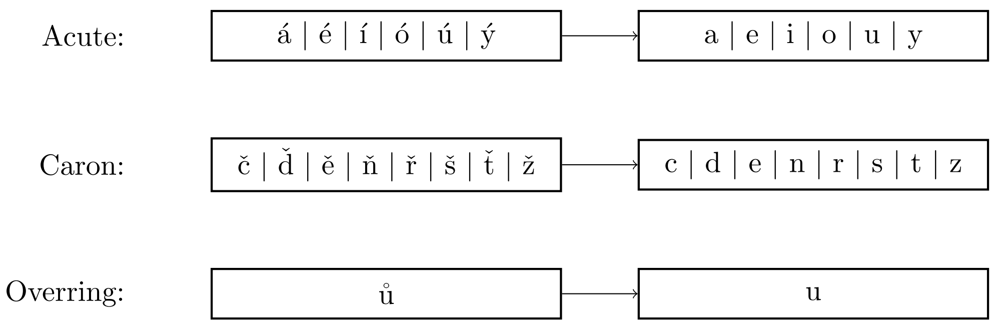


Figure 5: Usual changes in informal czech texts.

This problem is prevalent in social media, chatbots, and other informal written communications. As a result, embedding models face challenges in comprehending text without diacritics (hereinafter diacriticless), because the meanings of words may be compromised:

- byt (apartment) - být (to be)
- rád (to be glad) - řad (row or a line) - řád (order, religious order)
- krize (crisis) - kříže (crosses)

A potential solution involves adapting data representation to accommodate both formal and informal styles of writing.

This study will employ two distinct text representations: text with diacritics and text without diacritics. To ensure optimal evaluation, the diacritic text will be assessed using datasets that preserve these diacritics, while the diacriticless text will be evaluated using datasets that lack diacritics.

As detailed in Lst. 1, this script is used for creating diacriticless versions of the datasets.

```
sed 's/.*/\L&/' "$1" | iconv -f utf-8 -t ascii//TRANSLIT > diacriticless/"$1"
```

Listing 1: Script for removing diacritics using Unix utilities

Evaluation benchmark

UPV FAQ

Dataset is comprised of frequently asked questions (FAQs) and their answers from the Industrial Property Office of the Czech Republic (UPV) website[1]. Each test within the UPV FAQ dataset follows a consistent structure, consisting of three key elements:

- **Question**: This element specifies the question from the UPV website[1].
- **Answer**: This element provides the expected or correct response to the corresponding question.
- **Semantic Class**: This element assigns the test item to a specific category based on its semantic meaning. The details regarding the number of distinct semantic classes are presented in Table 1.

Test set	N_{tests}	$N_{classes}$
FAQv5	2054	211
FAQ50	561	49
FAQ76	2025	75
FAQ76v2	1965	75
All	6605	-

Table 1: Information about UPV FAQ tests, where N_{tests} is a number of tests in test set, $N_{classes}$ is a number of semantic classes.

This work will evaluate two key metrics using UPV FAQ dataset

- **Question matching accuracy**: This metric involves calculating the cosine similarity (1) between all possible question pairs within a dataset. A question is considered successfully matched if its second-highest cosine similarity score corresponds to another question belonging to the same class (i.e., the question with the highest similarity is likely the same question itself). The overall question matching accuracy is then computed as the ratio of successfully matched questions to the total number of question pairs evaluated.
- **Answer matching accuracy**: This metric assesses the system's ability to identify the correct answer for a given question. The system accomplishes this by directly comparing the question with pre-generated answer embeddings. By evaluating the similarity (1) between the question and each answer embedding, the system classifies the question as corresponding to the answer with the highest similarity score. The overall answer matching accuracy is then calculated as the proportion of questions for which the system correctly identifies the corresponding answer.

$$S_c(A, B) = \frac{A \cdot B}{|A|\,|B|}, \quad \text{where } A, B \text{ are vectors.} \tag{1}$$

Traditional models will be assessed using *jirkoada/upv_faq*[2] evaluator based on *fastText*[3] library, while transformer models will be evaluated using *ezvezdov/upv_faq_transformers*[4] eval-

[1] UPV website: https://upv.gov.cz/
[2] *jirkoada/upv_faq evaluator* on github: https://github.com/jirkoada/upv_faq
[3] fastText.cc library website: https://fasttext.cc/
[4] *ezvezdov/upv_faq_transformers* evaluator on github: https://github.com/ezvezdov/upv_faq_transformers

uator based on *sentence-transformers* library [5] [29].

Analogies

This study opted to exclude analogy-based evaluation methods for assessing the performance of transformer models. This decision stems from the fundamental differences between traditional pre-trained models and transformer architectures in how they generate word embeddings.

Traditional models typically employ pre-training techniques that result in static word embeddings. These embeddings capture comprehensive information about a word based on its occurrences throughout the training corpus. This characteristic allows for the use of analogy tests where the model is presented with a triplet of words (e.g., "King" - "Man" + "Woman") and expected to predict the fourth word that completes the analogy ("Queen").

However, transformer models operate differently. They generate word embeddings dynamically, considering the specific context in which a word appears within a sentence. This context-dependent nature of transformer embeddings renders traditional analogy tests inapplicable. Providing a single, isolated word to a transformer model would not be sufficient for it to generate a high-quality embedding that effectively captures all potential word contexts.

In light of these limitations, analogy-based evaluations are not well-suited for assessing the performance of transformer models. Therefore, we opted to prioritize alternative evaluation methods that are more compatible with the context-aware embedding generation process employed by transformers. These alternative methods, such as masked language modeling tasks, can provide a more accurate assessment of a transformer model's ability to capture semantic relationships within text.

Baseline

Due to the inherent morphological richness of the Czech language, this study adopts **FastText** as the baseline word embedding method. This decision is motivated by FastText's ability to effectively capture morphological variations within words, a characteristic that has been shown to be advantageous for languages like Czech. While other techniques like Word2Vec and GloVe have been explored for word embedding generation, they have demonstrated lower performance in this context. The FastText word embedding model will be trained using the fastText.cc library[3].

Training parameters

- Architecture: CBOW
- Vector dimensionality: 300
- Loss function: Negative sampling loss
- Dictionary threshold (the frequency of the word to be included in the dictionary): 130

Training data

FastText model is trained on a segment of a preprocessed Common Crawl repository [37], which encompasses raw data from web pages, as well as metadata and text extractions. Given

[5] *sentence-transformer* library: https://sbert.net/

the nature of this dataset, it is expected to include misspellings and text lacking diacritics.

(i) Eliminating duplicate entries
(ii) Filtering out lines containing fewer than 9 characters
(iii) Excluding URLs
(iv) Breaking lines into individual sentences
(v) Converting all text to lowercase
(vi) Removing lines containing words exceeding 30 characters
(vii) Excluding Hypertext Markup Language (HTML) tags with less than 100 characters
(viii) Removing lines surpassing 500 characters
(ix) Omitting words longer than 21 characters

Processed corpus contains 3.87 billion words.

Model summary

The FastText models trained in this study exhibit differences in vocabulary size. The diacritic model possesses a dictionary of approximately 800,000 words, while the diacriticless model contains roughly 762,000 words. This discrepancy reflects the inherent reduction in word count due to the removal of diacritics in the diacriticless dataset.

These vocabulary sizes directly influence the number of trainable parameters within each model. The number of parameters (N) can be calculated using the formula (2).

$$N = V_{\text{size}} d + d \tag{2}$$

where V_{size} is model vocabulary size
d is chosen dimensionality of word embeddings

Applying this formula to the vocabulary sizes of our models:

- The diacritics model possesses approximately 240 million parameters.
- The diacriticless model has approximately 229 million parameters.

Chosen transformer models

To ensure a comprehensive evaluation, a curated selection of embedding models will be utilized. This selection encompasses three distinct categories:

(i) **Existing Czech Embedding Models**: This category incorporates established Czech embedding models developed within the Czech NLP community. Their inclusion allows for a focused analysis of how these models perform specifically for the Czech language.
(ii) **Multilingual Models from the Massive Text Embedding Benchmark (MTEB)**: The evaluation will leverage highly regarded multilingual models readily available through the MTEB. This inclusion enables an assessment of how these models generalize to the Czech language, providing insights into their adaptability across languages.
(iii) **Popular Monolingual Models from the MTEB (rank is lower than 50)**: In addition to multilingual models, this selection will also include well-regarded monolingual models (models trained on a single language) from the MTEB. This allows for a comparative analysis of how these models, potentially trained on English or other high-resource languages.

To ensure transparency, reproducibility, and foster community development, this study will exclusively evaluate open-source text embedding models. Furthermore, to prioritize computational efficiency and applicability to our research setting, we will restrict our evaluation to models with a parameter size of less than 1 billion. In cases where a model suite offers both monolingual and multilingual versions, we will prioritize the multilingual version for evaluation. This choice aligns with our focus on tasks that may involve processing text data in multiple languages, including Czech.

For a comprehensive overview of the chosen models' architectural details, please refer to Appendix A. This appendix provides a more in-depth examination of the specific configurations employed by each model.

Czert-B

The Czert model [21] is a set of Czech BERT-like language representation models developed specifically to enhance performance in processing the Czech language. These models leverage the BERT and A Lite BERT (ALBERT) [25] architectures and are designed to outperform multilingual models by training exclusively on Czech data. The training set includes a comprehensive corpus of Czech texts, such as Wikipedia articles, news, and other texts, accumulating to around 36GB of data.

There are 2 variants of the Czert model, Czert-A and Czert-B. Unfortunately Czert-A model is not available, so we will test only Czert-B model. Czert-B model is based on the traditional BERT architecture (110M parameters). Model are pre-trained from scratch using MLM and NSP tasks. However, a slight modification is made to the NSP task to adapt it better to the Czech language corpus structure.

Seznam's models

This study leverages a group of compact word embedding models specifically designed for the Czech language. These models were developed by the Seznam research group with a focus on efficient word representation generation [11].

- **RetroMAE-Small**: This model leverages a BERT-small architecture pre-trained with the Retrieval-oriented Language Models Via Masked Auto-Encoder (RetroMAE) objective [18] on a custom Czech corpus. The RetroMAE objective focuses on enhancing the model's ability to learn from MLM tasks.
- **Dist-MPNet-ParaCrawl** & **Dist-MPNet-CzEng**: These models are distilled versions of the *sentence-transformers/all-mpnet-base-v2*[6] using a knowledge distillation approach. Distillation involves training a smaller model (BERT-small in this case) to mimic the performance of a larger, pre-trained model (*sentence-transformers/all-mpnet-base-v2* [6]). The two distilled models differ in their training data: Dist-MPNet-ParaCrawl utilizes the parallel cs-en dataset from ParaCrawl [27], while Dist-MPNet-CzEng leverages the parallel cs-en dataset CzEng [24].
- **SimCSE-RetroMAE-Small** & **SimCSE-Dist-MPNet-ParaCrawl** & **SimCSE-Dist-MPNet-CzEng**: These models are based on the previously described RetroMAE-Small, Dist-MPNet-ParaCrawl, and Dist-MPNet-CzEng models respectively. Each is further fine-tuned with the Simple Contrastive Learning of Sentence Embeddings (SimCSE)

[6] *sentence-transformers/all-mpnet-base-v2* model on the huggingface website: https://huggingface.co/sentence-transformers/all-mpnet-base-v2

objective [16]. SimCSE focuses on improving sentence embedding quality by encouraging models to generate similar representations for semantically equivalent sentences.

- **SimCSE-Small-E-Czech**: This model builds upon the Czech ELECTRA model [19]. It is fine-tuned with the SimCSE [16] objective to enhance the quality of its sentence embeddings.

The developed models are about eight times smaller and five times faster than conventional base-sized models, making them suitable for real-time applications where computational efficiency is critical. The models are trained using techniques like pre-training, knowledge distillation, and unsupervised contrastive fine-tuning to adapt to the limited availability of labeled Czech data.

Multilingual Bidirectional Encoder Representations from Transformers (mBERT)

mBERT [26] leverages the same transformer-based architecture as the original BERT model but boasts an increased number of parameters (178M) to enhance its capabilities. A key distinction of mBERT lies in its ability to understand and process text data across multiple languages. To achieve this multilingual proficiency, mBERT is trained on a massive dataset sourced from Wikipedia entries in 104 different languages. This corpus is meticulously constructed to ensure balanced representation, meaning each language is included regardless of the size or depth of its corresponding Wikipedia. This approach ensures that even languages with limited resources are adequately represented within the training data, fostering better performance for these languages.

Multilingual E5 (mE5)

mE5 models by Microsoft [9] are advanced text embedding models designed to operate across multiple languages based on English-only E5 models [10]. These models are available in three variants — small, base, and large — catering to different computational efficiency and performance needs. The mE5 models are trained using a two-phase approach. The first phase involves weakly-supervised contrastive pre-training on about 1 billion text pairs sourced from diverse multilingual corpora (Wikipedia, mC4, Multilingual CC News, Reddit, etc.). The second phase is supervised fine-tuning on approximately 1.6 million data points from high-quality labeled datasets (MS MARCO [31], Natural Questions (NQ) [28], TriviaQA [32], SQuAD [34], etc.).

Language-Agnostic BERT Sentence Embedding (LaBSE)

LaBSE model [15], developed by Google, is a state-of-the-art model for generating sentence embeddings that are effective across 109 languages. It leverages the transformer architecture and is trained on both monolingual (from sources like CommonCrawl [37] and Wikipedia) and bilingual data (mined from web pages). LaBSE utilizes a dual-encoder structure with BERT-based encoding modules. This setup enables the efficient processing of text pairs in multiple languages.

XLM-Roberta (XLM-R)

The XLM-R model [23] is a significant advancement in unsupervised cross-lingual representation learning, introduced by Facebook AI. It is specifically designed to improve performance across a wide range of cross-lingual tasks. XLM-R is pre-trained on a dataset dubbed 'CC-100', derived from Common Crawl [37], covering about 2.5 terabytes of text across 100 languages. This dataset is significantly larger than the ones used by its predecessors, offering a broader and more diverse linguistic foundation.

SentenceTransformers models

This work leverages SentenceTransformers [29], a Python framework offering a comprehensive collection of pre-trained models designed for state-of-the-art sentence, text, and image embeddings. These models are specifically tuned for various tasks, providing researchers with a powerful starting point for their investigations.

The following pre-trained models from SentenceTransformers will be evaluated in this study:

- **Distiluse-Base-Multilingual-Cased-v2**: This model leverages knowledge distillation, a technique for creating a smaller and faster model by capturing the knowledge from a larger, pre-trained model. In this case, it is a distilled version of the multilingual Universal Sentence Encoder [30]. Notably, this version supports sentence encoding for over 50 languages, including Czech.
- **Paraphrase-Multilingual-MiniLM-L12-v2**: This pre-trained model focuses on paraphrase identification. It is a multilingual version of the *sentence-transformers/paraphrase-MiniLM-L12-v2*[7] model, trained on parallel datasets encompassing over 50 languages, including Czech. By learning paraphrase relationships, this model can potentially capture semantic similarities between sentences.
- **Paraphrase-Multilingual-MPNet-Base-v2**: Similar to the previous model, this is a multilingual version of the *sentence-transformers/paraphrase-mpnet-base-v2*[8] model, trained on parallel data for over 50 languages, including Czech. This model is also designed for paraphrase identification, potentially aiding in tasks that require an understanding of semantic equivalence across sentences.

UAE-Large-V1

The UAE-Large-V1 model [5] focuses on enhancing short and long Semantic Textual Similarity (STS) tasks through a novel angle-optimized text embedding approach (AnglE) that works by dividing text embeddings into real and imaginary components in a complex space. This model is designed to address the challenges posed by the saturation zones of the cosine function, which can impede learning by causing vanishing gradients.

The model is trained using a hybrid objective that combines cosine similarity, in-batch negatives, and angle differences in complex space. This approach helps overcome the limitations of traditional cosine similarity measures by ensuring better gradient flow during training.

[7] *sentence-transformers/paraphrase-MiniLM-L12-v2* model on the huggingface website: https://huggingface.co/sentence-transformers/paraphrase-MiniLM-L12-v2

[8] *sentence-transformers/paraphrase-mpnet-base-v2* model on the huggingface website: https://huggingface.co/sentence-transformers/paraphrase-mpnet-base-v2

Training dataset includes around 21K samples and is specifically designed to evaluate STS performance on long texts, which are common in real-world applications.

Mxbai embed

This is study incorporates two models developed by MixedBread AI, mxbai-embed-large-v1 and mxbai-embed-2d-large-v1.

- **Mxbai-Embed-Large-v1** [3]: This model stands out as a high-performance English embedding model specifically designed for RAG systems. As of March 2024, it holds the leading position among publicly available models of its class within the MTEB. Notably, Mxbai-Embed-Large-v1 surpasses other models in tasks such as classification, clustering, and retrieval. The success of this model can be attributed to its robust training methodology. Mxbai-Embed-Large-v1 is trained on a massive dataset exceeding 700 million text pairs using a contrastive learning approach. This approach focuses on maximizing the similarity between semantically similar texts while contrasting dissimilar ones. Furthermore, the model undergoes fine-tuning with 30 million high-quality triplets leveraging the AnglE loss function. This fine-tuning step further refines the model's ability to distinguish semantic relationships within text data
- **Mxbai-Embed-2D-Large-v1** [4]: This model introduces a novel 2D-Matryoshka architecture [6], marking a significant advancement in the field of text embedding. The 2D-Matryoshka architecture offers a key advantage over traditional approaches: it allows for both dimensionality reduction of embeddings and chunking of model layers. This flexibility enables users to tailor the model size and complexity based on their specific computational needs. This allows for a crucial trade-off between computational efficiency and accuracy in resource-constrained environments. The model was designed to address the limitations of traditional dense embedding models, which produce fixed-size embeddings. These fixed-size embeddings can be inefficient for tasks requiring rapid processing or limited memory footprints. Mxbai-Embed-2D-Large-v1 tackles this challenge by employing a novel training strategy. This strategy incorporates contrastive training on a diverse dataset and fine-tuning on high-quality triplets. This approach allows the model to achieve competitive performance while offering significant reductions in resource consumption compared to traditional dense models.

Nomic-Embed-v1 and Nomic-Embed-v1.5

The Nomic model [7], focuses on generating high-quality embeddings for long-context text in a reproducible manner. It leverages a modified BERT architecture specifically optimized for handling sequences of up to 8192 tokens. This optimization includes innovative techniques like rotary positional embeddings and SwiGLU activations, which enhance the model's capacity to process longer texts effectively. The training process for Nomic Embed employs a two-stage approach. The first stage involves unsupervised contrastive pre-training on large-scale datasets. This pre-training equips the model with a strong foundation for capturing semantic relationships within text data. Subsequently, the model undergoes supervised fine-tuning using human-annotated data. This stage further refines the model's ability to generate accurate text embeddings, particularly beneficial for tasks involving both short and long contexts. Contrastive learning plays a crucial role in both stages, ultimately improving the overall effectiveness of the model in generating robust text embeddings.

Nomic Embed v1.5 builds upon the success of v1 by incorporating Matryoshka Representation Learning [2]. This approach offers developers greater flexibility in terms of embedding size, allowing for a trade-off between embedding size and performance. While there may be a slight reduction in performance, developers can choose a smaller embedding size if computational resources are limited.

General Text Embedding (GTE) and GTE-v1.5

The GTE model is a pivotal development in NLP, utilizing a deep Transformer encoder based on a BERT-like architecture for generating dense text embeddings. Initially, the GTE model is unsupervisedly pre-trained on approximately 800 million text pairs from diverse web sources, enabling broad semantic coverage. It then undergoes supervised fine-tuning with 3 million annotated text triples from varied datasets, including MS MARCO and Natural Questions, applying contrastive learning to enhance text relevance detection and similarity assessments. This dual-stage training strategy equips the GTE model to excel in complex NLP tasks, demonstrating significant versatility and robust performance across multiple applications.

There is also updated version GTE-v1.5.

BAAI General Embeddings (BGE)-v1.5

BGE Models [14] are built on a BERT-like architecture, available in three sizes: small, base, and large. They are trained using a sophisticated multi-stage process involving pre-training on large unlabeled data, contrastive learning for fine-tuning on text pairs, and multi-task learning with high-quality labeled datasets. This training regimen equips BGE models to handle a wide range of text embedding tasks with high efficiency and accuracy.

GIST-Embedding-v0

This study incorporates the GIST-Embedding-v0 suite of models, developed using the "Guided In-sample Selection of Training Negatives for Text Embedding Fine-tuning" (GIST) technique [8]. These models leverage pre-trained models as a foundation and are then fine-tuned on specific datasets.

The fine-tuning process for GIST-Embedding-v0 utilizes the Multitask Embeddings Data with Instructions (MEDI) datasets [12], which are further enhanced by the inclusion of mined triplets derived from the MTEB Classification training dataset. This targeted augmentation strategy aims to improve the model's performance on specific tasks.

Based on their performance in the MTEB, our evaluation will focus solely on the GIST-Embedding-v0 models built upon the BGE-v1.5 architecture (small, base, and large sizes). This selection ensures we investigate the most promising fine-tuning approaches within the GIST-Embedding suite.

TaylorAI tiny models

This study incorporates two distilled transformer models from TaylorAI for evaluation:

- **TaylorAI/BGE-micro-v2**[9]: This model is a 2-step distilled version of the small BGE-v1.5 model
- **TaylorAI/GTE-tiny**[10]: This model is a distilled version of the small version of GTE model

Ember-v1

This study leverages the Ember-v1 model, developed by LLMRails. Ember-v1 is a text embedding model trained on a comprehensive dataset of text pairs encompassing a wide range of domains such as finance, science, medicine, law, and beyond. Notably, the training process incorporates techniques inspired by both RetroMAE [18] and SetFit [17].

3.2 Optimizing Text Representations for RAG in Technical QA

Key factors

To identify the most suitable text representation model for RAG in technical QA, we propose a comprehensive evaluation approach that considers the following key factors:

- **Embedding Efficiency**: Computational efficiency is crucial for real-world applications. We will evaluate the processing time required for different representation models, considering factors like embedding dimensionality and model complexity. This ensures the chosen model can handle real-time QA tasks within reasonable processing time constraints.
- **Text Chunk Size**: The size of text chunks used by the RAG algorithm (e.g., words, sentences, or paragraphs) can impact performance. We will investigate the optimal chunk size for technical QA tasks. Here, we will balance the granularity of information retrieved by the model with computational efficiency. Smaller chunks (words) might capture finer details but require more processing, while larger chunks (paragraphs) might be faster to process but might miss relevant details.
- **Factuality of Generated Answers**: The primary objective is to generate answers that are factually accurate and consistent with the technical document. We will evaluate the models based on metrics that assess the factual correctness and coherence of the generated answers in response to technical questions. This ensures the generated answers are reliable and trustworthy for the user.

Chunk Size and K Parameter Selection

This section explores the selection of chunk sizes and the corresponding K parameter for evaluation using the RAG model. The chunk size refers to the length of text passages (number of characters) that will be processed by the RAG model during evaluation. The K parameter, on the other hand, determines the number of retrieved passages that the RAG model will consider when generating a response. Chosen parameters are shown at Table 2

[9]TaylorAI/bge-micro-v2 model on the huggingface website: https://huggingface.co/TaylorAI/bge-micro-v2

[10]TaylorAI/gte-tiny model on the huggingface website: https://huggingface.co/TaylorAI/gte-tiny

Chunk size	K
256	12
512	6
1024	3
2048	2
4096	1

Table 2: Chunk sizes and corresponding K parameter.

RAG pipeline

This section describes the evaluation methodology employed for the RAG model. The evaluation will be conducted using the *jirkoada/qa_evaluator*[11] tool.

Due to limitations in the availability of Czech language testing data, we opted to utilize a private technical English data corpus for this evaluation. The test set comprises two components: a technical manual and a set of 200 questions designed to assess document information retrieval capabilities.

For the embedding generation stage within RAG, we will leverage the $\text{GTE}_{\text{Small}}$ model as popular small model with good MTEB rating.

For the LLM component of RAG, we will utilize GPT-3.5-turbo [22]. The quality of the answers will be controlled and verified using GPT-4o[12].

Experiment hardware

The computational workload was handled by an Intel Core i7-8550U CPU. This is a 14nm mobile processor featuring 4 cores and 8 threads with a base clock of 1.8 GHz and a turbo boost frequency of 4.0 GHz.

[11] *jirkoada/qa_evaluator* on github: https://github.com/jirkoada/qa_evaluator
[12] Information about GPT-4o on OpenAI website: https://openai.com/index/hello-gpt-4o/

4 Experiments and Results

4.1 Text representation models

Table 3 presents the evaluation results for the chosen text embedding models.

Analysis of results

Czech models

Among the evaluated Czech models, SimCSE-RetroMAE-Small demonstrated the most promising performance. This model effectively handled both text with diacritics and diacriticless variants, outperforming other models within this category. Notably, the remaining Czech models achieved performance below the established baseline.

The performance of Czert-B can potentially be attributed to the fact that it was not specifically fine-tuned for the task of generating text embeddings. This highlights the importance of fine-tuning models for the specific task at hand to optimize their performance.

Regarding the Seznam models, the observed differences in results likely stem from variations in their training processes. The SimCSE-RetroMAE-Small model emerged as the clear leader within this group, suggesting that the fine-tuning strategy employed with SimCSE and RetroMAE may be particularly effective for these models.

Multilingual models

The evaluation revealed particularly strong performance from all versions of mE5 and LaBSE among the multilingual models. Notably, mE5 models achieved the highest overall results within the tested set. This finding suggests that the training objectives and architectures employed for these models are well-suited for capturing semantic similarity across languages, including Czech. Confusion matrices for the $\text{mE5}_{\text{Large}}$ evaluation are presented in Fig. 6.

Paraphrase-Multilingual-MiniLM-L12-v2 and Paraphrase-Multilingual-MPNet-Base-v2 demonstrated strong performance with text that included diacritics. However, their effectiveness notably decreased when processing diacriticless text. This suggests a limitation in these models' capacity to generalize across different forms of Czech text. To improve their performance, additional training or adaptation tailored to handle both diacritic and diacriticless variations in Czech text would be beneficial.

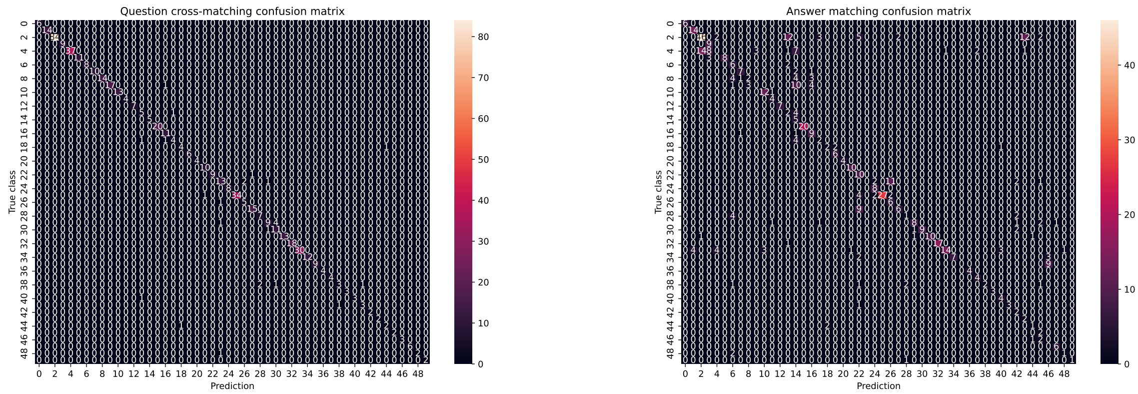
While mBERT and XLM-R did not achieve the same level of success as other models, it is important to consider the limitations of their unsupervised training methods (e.g., MLM and NSP) which may not be specifically optimized for the task of assessing semantic similarity. Utilizing alternative training objectives or supervised learning approaches tailored for this task could potentially lead to improved performance from these models in the context of Czech text analysis.

Monolingual models

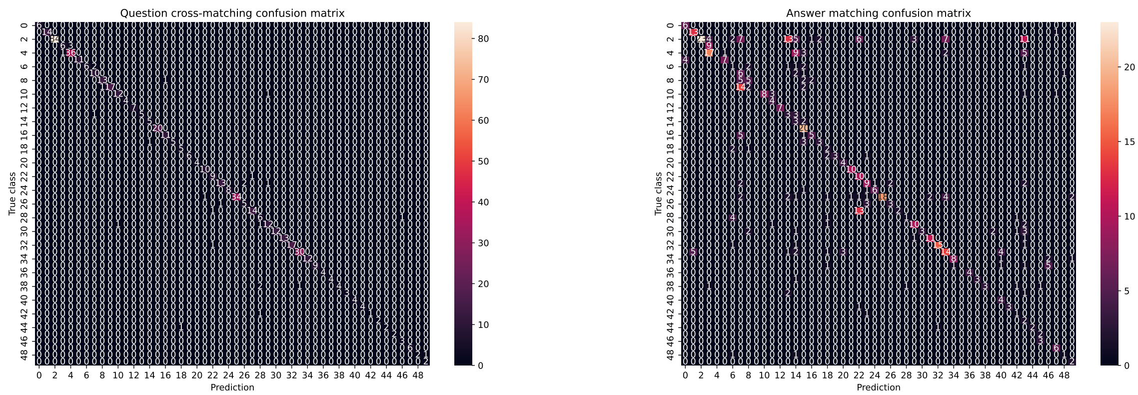
While all monolingual models achieved performance near the established baseline, this is a noteworthy finding considering they were not specifically trained on the Czech language.

This suggests that some level of semantic similarity can be captured between languages with inherent structural similarities, even without targeted training on the target language. However, it is important to acknowledge that languages possess distinct vocabularies, grammatical structures, and cultural nuances. Models trained solely on a language like English may struggle to fully grasp the intricacies of Czech text, potentially hindering their ability to achieve optimal performance in tasks involving semantic similarity assessment.

In contrast, several monolingual models, particularly the Nomic Embed models, the small version of GTE, and the large version of GTE-v1.5, exhibited performance exceeding the baseline. This highlights the potential effectiveness of certain model architectures, even when not specifically trained on the target language. Further investigation into the specific characteristics of these models that contribute to their success in this context might provide valuable insights for future research.



(a) Evaluatinon using diacritics tests.



(b) Evaluatinon using diacriticless tests.

Figure 6: Confusion matrices of $\mathrm{mE5}_{\mathrm{Large}}$ evaluated using diacritics (a) and diacriticless (b) FAQ50 subsets of UPV FAQ dataset.

Model	**QMA**$_\text{d}$	**AMA**$_\text{d}$	**QMA**$_\text{dl}$	**AMA**$_\text{dl}$	**#params**
		BASELINE			
FastText$_\text{diacritics}$	0.8304	0.2899	0.8110	0.2923	240M
FastText$_\text{diacriticless}$	0.8331	0.2864	0.8320	0.3020	229M
		CZECH MODELS			
Czert-B	0.8759	0.2469	0.8388	0.0977	110M
RetroMAE-Small	0.8651	0.2893	0.8634	0.2437	24M
Dist-MPNet-ParaCrawl	0.8540	0.1089	0.8344	0.0808	24M
Dist-MPNet-CzEng	0.8705	0.0487	0.8322	0.0426	24M
SimCSE-RetroMAE-Small	0.8682	0.3647	0.8649	0.3316	24M
SimCSE-Dist-MPNet-ParaCrawl	0.8817	0.2552	0.8602	0.2322	24M
SimCSE-Dist-MPNet-CzEng	0.8833	0.2278	0.8556	0.1450	24M
SimCSE-Small-E-Czech	0.8094	0.1074	0.8160	0.0878	13M
		MULTILINGUAL MODELS			
mBERT	0.8584	0.2012	0.8361	0.1306	178M
mE5$_\text{Small}$	0.8952	0.6078	0.8564	0.4446	118M
mE5$_\text{Base}$	0.8961	0.6019	0.8726	0.5134	278M
mE5$_\text{Large}$	**0.9084**	**0.6559**	**0.8944**	**0.5593**	560M
LaBSE	0.8875	0.3525	0.8594	0.3264	471M
XLM-R$_\text{Base}$	0.8011	0.0098	0.7701	0.0198	279M
XLM-R$_\text{Large}$	0.7884	0.0460	0.7411	0.0298	560M
Distiluse-Base-Multilingual-Cased-v2	0.8335	0.2978	0.7784	0.2369	135M
Paraphrase-Multilingual-MiniLM-L12-v2	0.8502	0.4062	0.8029	0.2576	118M
Paraphrase-Multilingual-MPNet-Base-v2	0.8752	0.4538	0.8354	0.3174	278M
		MONOLINGUAL MODELS			
UAE-Large-V1	0.8241	0.2913	0.8237	0.2931	335M
Mxbai-Embed-Large-v1	0.8308	0.2998	0.8302	0.2994	335M
Mxbai-Embed-2D-Large-v1	0.8260	0.2511	0.8260	0.2516	335M
Nomic-Embed-v1	0.8523	0.3553	0.8541	0.3751	137M
Nomic-Embed-v1.5	0.8513	0.3537	0.8520	0.3533	137M
Ember-v1	0.8259	0.2971	0.8253	0.2966	335
GTE$_\text{Small}$	0.8549	0.3632	0.8543	0.3634	33M
GTE$_\text{Base}$	0.8443	0.3645	0.8437	0.3643	109M
GTE$_\text{Large}$	0.8376	0.3345	0.8370	0.3352	335M
GTE-v1.5$_\text{Base}$	0.8501	0.3336	0.8499	0.3305	137M
GTE-v1.5$_\text{Large}$	0.8592	0.3294	0.8586	0.3289	434M
BGE-v1.5$_\text{Small}$	0.8479	0.3816	0.8474	0.3798	33M
BGE-v1.5$_\text{Base}$	0.8368	0.3246	0.8362	0.3240	109M
BGE-v1.5$_\text{Large}$	0.8244	0.2938	0.8238	0.2955	335M
GIST-Embedding-v0$_\text{Small}$	0.8498	0.2664	0.8493	0.2653	33M
GIST-Embedding-v0$_\text{Base}$	0.8307	0.3023	0.8307	0.3023	109M
GIST-Embedding-v0$_\text{Large}$	0.8219	0.2579	0.8213	0.2588	335M
TaylorAI/BGE-micro-v2	0.8476	0.3616	0.8475	0.3616	17M
TaylorAI/GTE-tiny	0.8492	0.3343	0.8488	0.3342	23M

Table 3: **Evaliation of models.** We show evaluation results where: **QMA**$_\text{d}$ (**QMA**$_\text{dl}$) are Question Match Accuracy for diacritics (diacriticless) model. **AMA**$_\text{d}$ (**AMA**$_\text{d}$) are Question Match Accuracy for diacritics (diacriticless) model. **#params** is total number of parameters.

Balanced models

To ensure the effectiveness of the evaluation process, a selection criterion was applied to the initial set of candidate models. This criterion focused on Question Matching Accuracy and Answer Matching Accuracy for both diacritic and diacriticless models. Models that exhibited performance below the established baseline for their respective category (diacritic or diacriticless) were excluded from further evaluation.

Additionally, models with lower performance metrics were removed if a smaller, more efficient model demonstrated comparable or superior accuracy. This approach ensures that the final selection of models for evaluation represents a balance between effectiveness and efficiency.

Model	**QMA**$_{\text{d}}$	**AMA**$_{\text{d}}$	**QMA**$_{\text{dl}}$	**AMA**$_{\text{dl}}$	**#params**
SimCSE-RetroMAE-Small	0.8682	0.3647	0.8649	0.3316	24M
GTE$_{\text{Small}}$	0.8549	0.3632	0.8543	0.3634	33M
mE5$_{\text{Small}}$	0.8952	0.6078	0.8564	0.4446	118M
mE5$_{\text{Base}}$	0.8961	0.6019	0.8726	0.5134	278M
mE5$_{\text{Large}}$	**0.9084**	**0.6559**	**0.8944**	**0.5593**	560M

Table 4: **Balanced models.** We show most factual models according to their efficiency, where: **QMA**$_{\text{d}}$ (**QMA**$_{\text{dl}}$) are Question Match Accuracy for diacritics (diacriticless) model. **AMA**$_{\text{d}}$ (**AMA**$_{\text{d}}$) are Question Match Accuracy for diacritics (diacriticless) model. **#params** is total number of parameters.

4.2 RAG Optimization

Table 5 presents the retrieval time associated with the GTE$_{\text{Small}}$ model for varying chunk sizes.

S_{chunk}	t_1	t_2	t_3	t_4	t_5
256	131s	132s	130s	131s	135s
512	170s	164s	156s	160s	160s
1024	159s	161s	168s	162s	159s
2048	112s	111s	110s	111s	111s
4096	77s	81s	81s	92s	86s

Table 5: Time required for RAG to generate document part embeddings at various chunk sizes. Where S_{chunk} represents size of the chunk in characters and t_n is time of n_{th} experiment.

An unexpected observation is the increase in processing time when moving from a chunk size of 256 to 512 characters. However, the processing time remains constant for chunk sizes of 512 and 1024 characters. Interestingly, the processing time then decreases for chunk sizes of 2048 and 4096 characters.

While the reason for the initial increase in processing time is unclear, the subsequent decrease can potentially be attributed to text truncation. With larger chunk sizes (2048 and 4096 characters), the text may be divided into fewer parts, each exceeding the maximum sequence length. This could lead to improved efficiency due to the Transformer model's ability to leverage parallelism when processing longer sequences. Conversely, smaller chunk sizes might result in a higher number of text segments, potentially hindering the model's ability to exploit parallelism effectively.

Results of the RAG evaluation are presented in Table 6.

S_{chunk}	K	ACC	t_{mean}
256	12	0.435	132s
512	6	0.575	162s
1024	3	0.645	162s
2048	2	0.640	111s
4096	1	**0.670**	**83s**

Table 6: RAG evaluation with different parameters.

Our analysis of the results in Table 6 (or similar table title) reveals a positive correlation between chunk size and model accuracy. The model achieves its highest accuracy when processing chunks of 4096 characters. This suggests that providing the model with larger text segments during the embedding process might contribute to improved performance in capturing semantic relationships within the text.

5 Discussion

5.1 Findings

This study explores the potential benefits of transformer-based models for text representation compared to traditional methods like FastText. Interestingly, the SimCSE-RetroMAE-Small transformer model achieves superior performance despite having significantly fewer parameters than FastText. This finding suggests that the inherent architecture of transformer models may be particularly adept at capturing semantic meaning within textual data.

Furthermore, the study reveals that certain monolingual models, even those not specifically trained on the Czech language used for evaluation, achieve surprisingly positive results. This suggests that some level of semantic similarity can be identified between languages with structural similarities, even without targeted training. However, it is important to acknowledge that these models might not reach optimal performance for tasks involving Czech text analysis.

This study observes that unsupervised training methods employed by some models (e.g., MLM and NSP) lead to lower performance compared to supervised training approaches. This suggests that supervised training on high-quality, task-specific datasets might be necessary to achieve optimal performance in tasks involving semantic similarity assessment.

Results indicate that embedding generation for less segmented text is faster compared to highly segmented text. However, the study finds that QA accuracy is maximized when utilizing larger chunk sizes. This suggests a potential trade-off between processing efficiency and model performance, requiring further investigation to determine the optimal balance for specific applications.

5.2 Improvements for future research

This study relies exclusively on pre-trained models for evaluation. While these models achieve promising results, a potential avenue for future research lies in fine-tuning these models specifically for the task of assessing semantic similarity in Czech text. Fine-tuning pre-trained models on a Czech-specific dataset tailored for semantic similarity tasks could potentially lead to further performance improvements.

The current study utilizes an English dataset for RAG evaluation. Optimal chunk size can vary between languages due to differences in average word length. Therefore, a valuable future research direction involves creating a new dataset focused on technical QA in Czech. Testing different chunk sizes within this new dataset would be crucial for identifying the optimal configuration for Czech technical QA tasks.

The study employs the K parameter within the RAG evaluation process. The initial K values were chosen proportionally to the chunk size, mirroring the default settings within the evaluation tool. However, to optimize performance for the QA task, further investigation into the impact of varying K values is recommended. Evaluating a broader range of K values alongside the varying chunk sizes could lead to the identification of the optimal configuration for maximizing RAG's performance in the context of this specific English technical document retrieval task. This optimal configuration could then be compared to performance on the newly created Czech QA dataset.

6 Conclusion

This thesis investigated the effectiveness of transformer-based models for text representation compared to traditional methods in the context of semantic similarity assessment for Czech text.

The analysis began with a comprehensive review of traditional text representation methods. Their architectures, underlying principles, strengths, weaknesses, and specific applications were examined. This review established a foundation for understanding the evolution of text representation techniques.

Following this, the study shifted its focus to transformer architectures. Here, the investigation delved into the inner workings of these models and explored their advantages over traditional methods. The BERT model served as a specific example, with an explanation of its training process and its strengths in capturing semantic meaning from textual data.

To ensure objective assessment of model quality for the chosen task, two relevant evaluation methods were reviewed. These established methods provided a framework for comparing the performance of different text representation models used for semantic similarity assessment in Czech text.

Next, the investigation explored the RAG model. Core concepts, operational principles, and critical parameters influencing RAG's accuracy were examined. This in-depth analysis proved crucial for effectively configuring and evaluating RAG within the context of the chosen task.

Recognizing the importance of language-specific analysis, Czech was chosen as the target language. The study incorporated both diacritic and diacritic-less text versions to account for potential variations within Czech text data. The established UPV FAQ benchmark served as the standard for consistent and reliable evaluation.

A baseline performance metric was established using the FastText model. This baseline provided a benchmark for comparing the performance of the transformer-based models. Following the establishment of this baseline, a diverse selection of 15 transformer-based model groups (encompassing a total of 37 models) were chosen for further evaluation.

The RAG evaluation process involved testing five different chunk sizes. This exploration aimed to understand the impact of chunk size on both the factuality (accuracy) of retrieved information and computational efficiency (processing time). The initial stage focused on selecting optimal text representation models. We employed $\mathrm{GTE}_{\mathrm{Small}}$ for embedding generation and GPT-3.5-turbo for answer generation. GPT-4o was used in a separate process to assess the quality of answers generated by GPT-3.5-turbo.

After we made evaluation of the chosen models to detect best text representation models. The evaluation results revealed that a significant portion of the transformer-based models outperformed the baseline, suggesting their promise for semantic similarity assessment in Czech text. A detailed analysis of model performance and influencing factors identified mE5 $_{\mathrm{Large}}$ as the top performer. A confusion matrix visualized its evaluation on a specific benchmark subset. Additionally, "balanced models" exhibiting the best performance relative to their model size were highlighted. These included SimCSE-RetroMAE-Small, the small version of GTE, and all sizes of mE5, demonstrating the potential of both large and efficient models for this task.

The evaluation then proceeded to analyze the impact of chunk size on the RAG model itself. The first stage involved calculating the average time required for embedding generation with different chunk sizes using $\mathrm{GTE}_{\mathrm{Small}}$. This analysis aimed to identify potential variations in processing time based on chunk size. The results yielded unexpected findings, which were subsequently interpreted as a potential consequence of improved transformer model parallelism when handling larger sequences of tokens.

Following the analysis of processing time, the study investigated the impact of chunk size on model accuracy. Based on the evaluation results, a chunk size of 4096 characters was identified as optimal for the RAG model.

7 References

[1] A. Joshi, *Rag series part 1: How to choose the right embedding model for your application*, 2024. [Online]. Available: https://www.mongodb.com/developer/products/atlas/choose-embedding-model-rag/.

[2] A. Kusupati, G. Bhatt, A. Rege, *et al.*, *Matryoshka representation learning*, 2024. arXiv: 2205.13147 [cs.LG].

[3] S. Lee, A. Shakir, D. Koenig, and J. Lipp. "Open source strikes bread - new fluffy embeddings model." (2024), [Online]. Available: https://www.mixedbread.ai/blog/mxbai-embed-large-v1.

[4] S. Lee, A. Shakir, J. Lipp, and D. Koenig. "Fresh 2d-matryoshka embedding model." (2024), [Online]. Available: https://www.mixedbread.ai/blog/mxbai-embed-2d-large-v1.

[5] X. Li and J. Li, *Angle-optimized text embeddings*, 2024. arXiv: 2309.12871 [cs.CL].

[6] X. Li, Z. Li, J. Li, H. Xie, and Q. Li, *2d matryoshka sentence embeddings*, 2024. arXiv: 2402.14776 [cs.CL].

[7] Z. Nussbaum, J. X. Morris, B. Duderstadt, and A. Mulyar, *Nomic embed: Training a reproducible long context text embedder*, 2024. arXiv: 2402.01613 [cs.CL].

[8] A. V. Solatorio, *Gistembed: Guided in-sample selection of training negatives for text embedding fine-tuning*, 2024. arXiv: 2402.16829 [cs.LG].

[9] L. Wang, N. Yang, X. Huang, L. Yang, R. Majumder, and F. Wei, *Multilingual e5 text embeddings: A technical report*, 2024. arXiv: 2402.05672 [cs.CL].

[10] L. Wang, N. Yang, X. Huang, *et al.*, *Text embeddings by weakly-supervised contrastive pre-training*, 2024. arXiv: 2212.03533 [cs.CL].

[11] J. Bednář, J. Náplava, P. Barančíková, and O. Lisický, *Some like it small: Czech semantic embedding models for industry applications*, 2023. arXiv: 2311.13921 [cs.CL].

[12] H. Su, W. Shi, J. Kasai, *et al.*, *One embedder, any task: Instruction-finetuned text embeddings*, 2023. arXiv: 2212.09741 [cs.CL].

[13] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, *Attention is all you need*, 2023. arXiv: 1706.03762 [cs.CL].

[14] S. Xiao, Z. Liu, P. Zhang, and N. Muennighoff, *C-pack: Packaged resources to advance general chinese embedding*, 2023. arXiv: 2309.07597 [cs.CL].

[15] F. Feng, Y. Yang, D. Cer, N. Arivazhagan, and W. Wang, *Language-agnostic bert sentence embedding*, 2022. arXiv: 2007.01852 [cs.CL].

[16] T. Gao, X. Yao, and D. Chen, *Simcse: Simple contrastive learning of sentence embeddings*, 2022. arXiv: 2104.08821 [cs.CL].

[17] L. Tunstall, N. Reimers, U. E. S. Jo, *et al.*, *Efficient few-shot learning without prompts*, 2022. arXiv: 2209.11055 [cs.CL].

[18] S. Xiao, Z. Liu, Y. Shao, and Z. Cao, *Retromae: Pre-training retrieval-oriented language models via masked auto-encoder*, 2022. arXiv: 2205.12035 [cs.CL].

[19] M. Kocián, J. Náplava, D. Štancl, and V. Kadlec, *Siamese bert-based model for web search relevance ranking evaluated on a new czech dataset*, 2021. arXiv: 2112.01810 [cs.IR].

[20] P. Lewis, E. Perez, A. Piktus, *et al.*, *Retrieval-augmented generation for knowledge-intensive nlp tasks*, 2021. arXiv: 2005.11401 [cs.CL].

[21] J. Sido, O. Pražák, P. Přibáň, J. Pašek, M. Seják, and M. Konopík, *Czert – czech bert-like model for language representation*, 2021. arXiv: 2103.13031 [cs.CL].

[22] T. B. Brown, B. Mann, N. Ryder, *et al.*, *Language models are few-shot learners*, 2020. arXiv: 2005.14165 [cs.CL].

[23] A. Conneau, K. Khandelwal, N. Goyal, *et al.*, *Unsupervised cross-lingual representation learning at scale*, 2020. arXiv: 1911.02116 [cs.CL].

[24] T. Kocmi, M. Popel, and O. Bojar, *Announcing czeng 2.0 parallel corpus with over 2 gigawords*, 2020. arXiv: 2007.03006 [cs.CL].

[25] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, *Albert: A lite bert for self-supervised learning of language representations*, 2020. arXiv: 1909.11942 [cs.CL].

[26] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, 2019. arXiv: 1810.04805 [cs.CL].

[27] M. Esplà, M. Forcada, G. Ramírez-Sánchez, and H. Hoang, "Paracrawl: Web-scale parallel corpora for the languages of the eu," in *Proceedings of Machine Translation Summit XVII: Translator, Project and User Tracks*, European Association for Machine Translation, 2019, pp. 118–119.

[28] T. Kwiatkowski, J. Palomaki, O. Redfield, *et al.*, "Natural questions: A benchmark for question answering research," *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3877–3886, 2019.

[29] N. Reimers and I. Gurevych, *Sentence-bert: Sentence embeddings using siamese bert-networks*, 2019. arXiv: 1908.10084 [cs.CL].

[30] Y. Yang, D. Cer, A. Ahmad, *et al.*, *Multilingual universal sentence encoder for semantic retrieval*, 2019. arXiv: 1907.04307 [cs.CL].

[31] P. Bajaj, D. Campos, N. Craswell, *et al.*, *Ms marco: A human generated machine reading comprehension dataset*, 2018. arXiv: 1611.09268 [cs.CL].

[32] M. Joshi, E. Choi, D. S. Weld, and L. Zettlemoyer, *Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension*, 2017. arXiv: 1705.03551 [cs.CL].

[33] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *arXiv preprint arXiv:1607.04606*, 2016.

[34] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, *Squad: 100,000+ questions for machine comprehension of text*, 2016. arXiv: 1606.05250 [cs.CL].

[35] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162.

[36] T. Mikolov, K. Chen, G. Corrado, and J. Dean, *Efficient estimation of word representations in vector space*, 2013. arXiv: 1301.3781 [cs.CL].

[37] *Common crawl*. [Online]. Available: https://commoncrawl.org/.

[38] W. Contributors, *Bag-of-words model - wikipedia*. [Online]. Available: https://en.wikipedia.org/wiki/Bag-of-words_model.

[39] W. Contributors, *Tf-idf - wikipedia*. [Online]. Available: https://en.wikipedia.org/wiki/Tf%E2%80%93idf.

[40] U. Jamil, *Umar jamil [youtube channel]*. [Online]. Available: https://www.youtube.com/@umarjamilai.

[41] R. Theja, "Evaluating the ideal chunk size for a rag system using llamaindex," *LLAMAi*, [Online]. Available: https://www.llamaindex.ai/blog/evaluating-the-ideal-chunk-size-for-a-rag-system-using-llamaindex-6207e5d3fec5.

A Appendix A

Table 7 presents a comparison of the architectural characteristics of the evaluated text embedding models. This table includes the following parameters for each model

- **Number of Layers** (L): This refers to the number of encoder or decoder layers stacked within the model architecture. A higher number of layers typically indicates a more complex model with greater capacity to learn complex relationships within the data.
- **Number of Hidden States** (H_m): This represents the dimensionality of the internal representations processed by each layer within the model. A larger number of hidden states allows the model to capture a richer set of features from the input data.
- **Dimension of Feed-Forward Layer** (H_{ff}): This parameter specifies the dimensionality of the hidden layer within the feed-forward sub-layer of each transformer encoder block. The feed-forward sub-layer allows the model to learn non-linear relationships between input features.
- **Number of Attention Heads** (A): This refers to the number of parallel attention mechanisms employed within each encoder or decoder layer. A higher number of attention heads allows the model to focus on different aspects of the input data simultaneously.
- **Dimension of Output Embedding** (D): This specifies the dimensionality of the final vector representation generated by the model for each input text sequence.
- **Maximum Sequence Length** (T_{max}): This parameter indicates the maximum number of tokens a model can process within a single input sequence. Models with a larger T_{max} can handle longer text inputs without requiring truncation.
- **Vocabulary Size** (V): This represents the total number of unique words (tokens) the model's vocabulary encompasses.
- **Total Number of Parameters** (N_p): This denotes the total number of trainable parameters within the model. A larger number of parameters typically indicates a more complex model with greater capacity, but also higher computational demands.

For Transformer encoders, the number of parameters can be approximated by Eq. (3).

$$N_p \approx 4LH_m^2 + 2LH_mH_{ff} + VH_m \tag{3}$$

Model	L	H_m	H_{ff}	A	D	T_{max}	V	N_p
$\text{FastText}_{\text{diacritics}}$	-	-	-	-	300	-	800K	240M
$\text{FastText}_{\text{diacriticless}}$	-	-	-	-	300	-	762K	229M
Czert-B	12	768	3072	12	768	512	31K	110M
RetroMAE-Small	12	256	1024	4	256	512	58K	24M
Dist-MPNet-ParaCrawl	12	256	1024	4	256	512	58K	24M
Dist-MPNet-CzEng	12	256	1024	4	256	512	58K	24M
SimCSE-RetroMAE-Small	12	256	1024	4	256	512	58K	24M
SimCSE-Dist-MPNet-ParaCrawl	12	256	1024	4	256	512	58K	24M
SimCSE-Dist-MPNet-CzEng	12	256	1024	4	256	512	58K	24M
SimCSE-Small-E-Czech	12	256	1024	4	256	512	31K	13M
mBERT	12	768	3072	12	768	512	120K	178M
$\text{mE5}_{\text{Small}}$	12	384	1536	12	384	512	250K	118M
mE5_{Base}	12	768	3072	12	768	514	250K	278M
$\text{mE5}_{\text{Large}}$	24	1024	4096	16	1024	514	250K	560M
LaBSE	12	768	3072	12	768	512	502K	471M
$\text{XLM-R}_{\text{Base}}$	12	768	3072	12	768	514	250K	279M
$\text{XLM-R}_{\text{Large}}$	24	1024	4096	16	1024	514	250K	560M
Distiluse-Base-Multilingual-Cased-v2	6	768	3072	12	512	512	120K	135M
Paraphrase-Multilingual-MiniLM-L12-v2	12	384	1536	12	384	512	250K	118M
Paraphrase-Multilingual-MPNet-Base-v2	12	768	3072	12	768	514	250K	278M
UAE-Large-V1	24	1024	4096	16	1024	512	31K	335M
Mxbai-Embed-Large-v1	24	1024	4096	16	1024	512	31K	335M
Mxbai-Embed-2D-Large-v1	24	1024	4096	16	1024	512	31K	335M
Nomic-Embed-v1	12	768	3072	12	768	8192	31K	137M
Nomic-Embed-v1.5	12	768	3072	12	768	8192	31K	137M
Ember-v1	24	1024	4096	16	1024	512	31K	335M
$\text{GTE}_{\text{Small}}$	12	384	1536	12	384	512	31K	33M
GTE_{Base}	12	768	3072	12	768	512	31K	109M
$\text{GTE}_{\text{Large}}$	24	1024	4096	16	1024	512	31K	335M
$\text{GTE-v1.5}_{\text{Base}}$	12	768	3072	12	768	8192	31K	137M
$\text{GTE-v1.5}_{\text{Large}}$	24	1024	4096	16	1024	8192	31K	434M
$\text{BGE-v1.5}_{\text{Small}}$	12	384	1536	12	384	512	31K	33M
$\text{BGE-v1.5}_{\text{Base}}$	12	768	3072	12	768	512	31K	109M
$\text{BGE-v1.5}_{\text{Large}}$	24	1024	4096	16	1024	512	31K	335M
$\text{GIST-Embedding-v0}_{\text{Small}}$	12	384	1536	12	512	512	31K	33M
$\text{GIST-Embedding-v0}_{\text{Base}}$	12	768	3072	12	768	512	31K	109M
$\text{GIST-Embedding-v0}_{\text{Large}}$	24	1024	4096	16	1024	512	31K	335M
TaylorAI/BGE-micro-v2	3	384	1536	12	512	512	31K	17M
TaylorAI/GTE-tiny	6	384	1536	12	384	512	31K	23M

Table 7: Details on model sizes.