Final report Summer Internship

Tilian bourachot

July 2024

Contents

1 Quick introduction about language models and Llms **2**

2 Goals of the report **3**

3 Feature engineering and Words Embedding **4**
3.1 Dynamic Representation 4
3.1.1 Transformer model 4
3.1.2 Precision about how BERT and GPT are working 6
3.1.3 Conclusion 7
3.2 Static Reprensation 7
3.2.1 Introduction about Word2Vec 7
3.2.2 Applications of Word2Vec 8
3.2.3 Word2Vec and Text Generation 9
3.2.4 Conclusion 10

4 Size compression for Llms **11**
4.1 Model's parameters reduction 11
4.1.1 Compression Techniques 11
4.1.2 Evaluation and Benchmarking 13
4.1.3 Challenges and Future Directions 13
4.2 Embeddings' dimension reduction 14

5 Annex **16**
5.1 Annex : FlashAttention 16
5.2 Annex : High quality data 17
5.3 Annex 2 : how do they determine the number of dimensions ? . . 18

Chapter 1

Quick introduction about language models and Llms

In recent years, the field of natural language processing (NLP) has witnessed significant advancements, primarily driven by the development and application of language models (LMs). Language models, such as GPT (Generative Pre-trained Transformer) and BERT (Bidirectional Encoder Representations from Transformers), are designed to predict the next word in a sequence based on the preceding words. These probabilistic models are capable of generating text that adheres to specific linguistic styles or patterns, making them invaluable for various applications, including text completion, translation, and summarization.

Building upon the foundation laid by traditional language models, Large Language Models (LLMs) have emerged, leveraging advanced deep learning techniques to enhance their capabilities in understanding and generating natural language. LLMs aim to produce coherent and contextually appropriate text by analyzing extensive datasets during their training. The core of LLMs' functionality lies in transformer architectures, which allow these models to handle complex language tasks with remarkable accuracy and fluency.

The versatility and power of LLMs have opened up numerous possibilities across different domains. They are utilized in conversational agents, content creation, sentiment analysis, and many other areas where the generation of human-like text is essential.

Chapter 2

Goals of the report

Our goal in this study is, first and foremost, to discover how language models function, what their architecture is, and what the state-of-the-art models are. Next, we want to conduct experiments on new models such as Phi3 and Llama3, as well as LLMs like GPT and BERT, to try to understand how they represent words in embeddings and how those models interpret this. Finally, our main goal is to use PyChest, introduced by A. Khaleghi as a Python package that provides tools for the simultaneous estimation of multiple changepoints in the distribution of piecewise stationary time series. We would like to adapt this algorithm to feature engineering and word embeddings to detect when the main subject of a text is changing. To achieve this goal, we first need to find a way for word embedding, choose between dynamic and static word embedding according to our issue. Subsequently, the algorithm of PyChest is implemented for low-dimension vectors, so we would have to find a way to reduce the number of dimensions of our word embeddings.

Chapter 3

Feature engineering and Words Embedding

Feauture engineering : [1] Feature extraction is vital in ANLP tasks, allowing for the effective representation and analysis of textual data. It captures linguistic properties, semantic information, and structural patterns, enhancing performance in sentiment analysis, machine translation, text summarization, and more. Extracting meaningful features enables deeper linguistic analysis, improving accuracy and efficiency in ANLP applications.

Feature extraction involves selecting and transforming raw data into a set of relevant features that effectively represent and describe the data. Word embedding, as a semantic approach to feature extraction, captures the underlying meaning and semantic relationships between words, representing textual data in a dense vector space. This representation enhances the understanding of semantic similarities and contextual associations, facilitating more effective text analysis and interpretation.

There is a wide range of state-of-the-art approaches for feature extraction, specifically word embedding. This subsection provides an overview of the most commonly used methods in the Arabic language, categorized as either static or contextualized.

The article [1] indicates that the embedding step significantly impacts model performance in ANLP. Additionally, the Arabic pre-trained BERT achieves the best performance when used to generate embeddings.

3.1 Dynamic Representation

3.1.1 Transformer model

The Transformer model represents a significant advancement in natural language processing and understanding. Introduced in [6] as a neural network architecture, it revolutionizes how machines comprehend the context of words

within sentences. Unlike traditional Recurrent Neural Networks (RNNs), which process words sequentially, Transformers can handle words in parallel, thanks to their attention mechanisms. This ability allows Transformers to understand word context more effectively and efficiently.

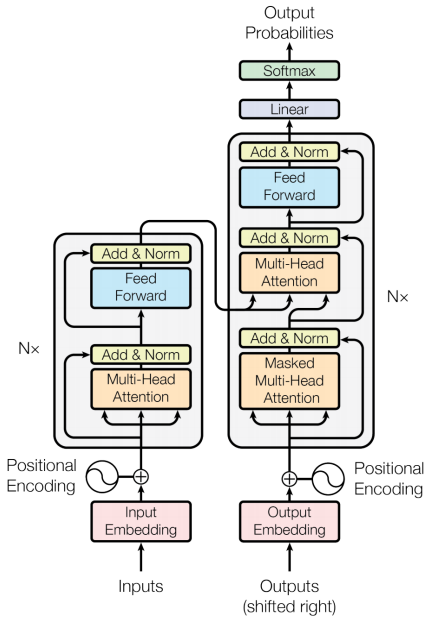

Figure 1: The Transformer - model architecture.

- Attention Mechanism

  A key component of the Transformer model is the self-attention mechanism. This mechanism weighs the relevance of different words in the input when making predictions. For each token in the sequence, the self-attention process calculates a weighted sum of all tokens, with the weights determined by the attention mechanism. This approach allows the model to capture the dependencies between words irrespective of their position in the sequence, enhancing its contextual understanding. It allows the vectors to talk to each other and to pass information to uptade their weights according to the sentence. For example : a machin Learning model and a fashion model, model has not the same meaning in both sentences. A current state-of-the-art mechanism is FlashAttention, which is used in small language models like Phi3 to achieve almost the same performance as LLMs with fewer resources. See the appendix to explore FlashAttention [2].

- Advantages Over Previous Architectures

Transformers offer several advantages over previous neural network architectures like RNNs and their variants (LSTM, GRU). Firstly, they are highly suitable for parallel computing because they do not require sequential processing of tokens. This parallelism significantly speeds up the computation, making Transformers more efficient in handling large datasets and complex tasks.

- Pre-trained Models

  Pre-trained Transformer models like BERT and GPT are trained on vast amounts of text data. These models can be fine-tuned for specific tasks, such as sentiment analysis, language translation, and text summarization. The pre-training involves exposing the models to large corpora of text, enabling them to learn grammar, facts about the world, reasoning abilities, and even biases present in the data.

- ChatGPT and GPT Models

  The core of ChatGPT is a Transformer. Specifically, ChatGPT is based on the GPT (Generative Pre-trained Transformer) architecture. GPT models are designed to predict the next word in a sequence, processing text from left to right. This unidirectional approach makes them particularly well-suited for text generation tasks. GPT-3, for example, consists of 175 billion parameters, making it one of the most powerful language models available.

- Self-Supervised Learning

  Transformer models like GPT and BERT are trained using self-supervised learning. For GPT, this means predicting the next word in a sentence, while BERT predicts a word based on the surrounding context. This training methodology allows the models to learn rich, contextual embeddings of the input text, known as dynamic embeddings, which can be fine-tuned for a variety of downstream applications.

3.1.2 Precision about how BERT and GPT are working

The transformer architecture is divided in 3 main part : Encoder-Decoder, Encoder only and Decoder only.

Encoder-Decoder can understand and generate sentences, they are used for Translation and Text-summurization.

The encoder is responsible for understanding and extracting relevant information from the input text. It then produces a continuous representation (embedding) of the input text, which is passed to the decoder. Encoders are useful for tasks such as classification, sequence tagging, and sentiment analysis. For example, BERT (Bidirectional Encoder Representations from Transformers) is a well-known encoder model.

BERT uses masked language modeling and next sentence prediction as pre-training objectives. In masked language modeling, random word tokens in the

input sequence are masked, and the model is trained to predict these masked tokens based on the surrounding context. This approach allows BERT to learn rich contextual representations of the input texts. Additionally, BERT's next sentence prediction objective helps the model understand the relationship between consecutive sentences.

Typically, models like BERT need to be fine-tuned for various downstream tasks, such as sentiment analysis or other specific tasks. However, it's important to note that while BERT can understand text, it cannot generate text. Its primary function is to comprehend and analyze textual information rather than create new text.

Finally, the Decoder model generates output word by word, and the predictions for position i can depend only on the known outputs at positions less than i. LLms like GPT2,3,4, ChatGPT are decoder and also state of art model such as Phi3 or Llama3 are too. Also, Decoder are controlled by prompting (instruction).

3.1.3 Conclusion

We have explored the state-of-the-art mechanism of transformer models, which can generate coherent text by utilizing the context and meaning of other words in a sentence. Each token can be represented by different vectors based on its meaning, context, and position within the sentence. Conversely, these models require extensive training data. Some advanced models, such as Phi3, attempt to reduce the required amount of data by using high-quality datasets, as mentioned in the appendix. Additionally, these models contain a large number of parameters (approximately 3 billion for Phi3 and 175 billion for GPT-3), which necessitate significant storage. The details on how models determine the number of parameters they retain are described in the appendix. In a next section, we will discuss ways to reduce this number. Lastly, it is important to consider static embeddings, as they use fewer parameters and will be more useful in our future experiments on change-point detection.

3.2 Static Reprensation

3.2.1 Introduction about Word2Vec

According to the article "word2vec Parameter Learning Explained" by Xin Rong [5], published in 2016, which was before the discovery of the Transformer architecture, Word2Vec is only an algorithm based on neural network learning. It enables the generation of a matrix that contains, for each word in the vocabulary, its vector representation based on its definition, extracted from context in the studied corpus / training dataset. It then captures similarities with other words etc.

Then, Word2Vec and GloVe produce **static embeddings** (the same vector for a word regardless of its position or context), in contrast to Transformer

models, which generate embeddings that vary depending on the context in which the words appear.

3.2.2 Applications of Word2Vec

- **Modeling Word Similarity**

  Word2Vec captures the semantic similarity between words by representing them as vectors in a continuous vector space where similar words are positioned close to each other. This enables various applications that require understanding and leveraging word relationships.

  Applications:

  - *Synonym Search:* Identifying synonyms by finding close vectors in the space. For example, the words "happy" and "joyful" will have similar vector representations.
  - *Search and Recommendation:* Enhancing the relevance of results by considering semantic similarity. Search engines and recommendation systems can suggest items that are contextually similar to what the user is interested in.

- **Text Analysis and NLP** Word2Vec provides dense word representations that can be used in various text analysis and NLP tasks.

  Applications:

  - *Text Classification:* Using word vectors as input features for text classification models, which categorize texts into predefined categories, such as spam detection or topic classification.
  - *Sentiment Analysis:* Representing words and phrases as vectors to determine the sentiment expressed in texts, such as identifying positive, negative, or neutral sentiments in reviews.

- **Machine Translation and Multilingual Processing** Word2Vec assists in aligning words from different languages based on their semantic meanings.

  Applications:

  - *Semantic Alignment:* Aligning words from different languages to similar concepts, aiding in the creation of multilingual word embeddings.
  - *Cross-lingual Tasks:* Enhancing the performance of tasks involving multiple languages, such as translating documents or extracting information from multilingual sources.

- **Anomaly and Error Detection** Word2Vec can detect anomalies and linguistic errors by analyzing words in their context.

  Applications:

- *Spelling and Grammar Correction:* Detecting and correcting linguistic errors by understanding the context in which words appear. For example, suggesting corrections for words that do not fit well in their context.
- *Outlier Detection:* Identifying words or phrases that are unusual or out of place in a given context, useful for spotting errors or anomalies in text data.

- **Clustering and Data Visualization** Word vectors can be clustered to uncover themes or topics within a text corpus, and visualized in lower-dimensional spaces.

  Applications:

  - *Word Clustering:* Grouping words to discover themes or topics within a corpus, revealing underlying patterns and relationships in the data.
  - *Dimensionality Reduction:* Visualizing word vectors using techniques like t-SNE or PCA, making it easier to interpret the relationships between words.

3.2.3 Word2Vec and Text Generation

Word2Vec is not directly used to generate text in the same way that models like GPT (Generative Pre-trained Transformer) do. However, it plays a foundational role in natural language processing (NLP) by providing meaningful word embeddings that can be utilized in various NLP tasks, including text generation.

- **Differences between Word2Vec and GPT**
  - **Purpose:**
    - *Word2Vec:* Designed to learn vector representations of words, capturing semantic relationships between them.
    - *GPT:* Designed to generate coherent and contextually relevant text by predicting the next word in a sequence.
  - **Architecture:**
    - *Word2Vec:* Utilizes shallow neural networks (such as CBOW or Skip-gram models) to produce word embeddings.
    - *GPT:* Utilizes transformer architecture, consisting of multiple layers of self-attention mechanisms to model long-range dependencies in text.
  - **Output:**
    - *Word2Vec:* Outputs dense vectors representing words.
    - *GPT:* Outputs sequences of words to generate text.

- **How Word2Vec is Utilized in Text Generation**

  While Word2Vec itself does not generate text, it can be a component in more complex systems that do.

  - **Preprocessing and Feature Extraction:**
    - *Embedding Initialization:* Word2Vec embeddings can initialize the embedding layers of more complex models like transformers.
    - *Feature Representation:* Word2Vec can convert text into vector representations that are fed into other NLP models.

3.2.4 Conclusion

We just see that static embeddings, like Word2vec, allow representing words as vectors. However, unlike dynamic embeddings and transformer architectures, static embeddings do not consider the context of the sentence and the position of the word. This means that a word has the same representation in two different sentences, even if the context is not the same. But this method also has advantages: for instance, words are represented with vectors of 300 dimensions with some Word2Vec models, instead of 3000-dimensional vectors for Phi3 or 12000 for GPT-3. Thus, it is easier to process them compared to high-dimensional vectors. Finally, our main goal, which is to detect change points (topics) in a text, will also be a way to consider words out of their context and correctly spot the change in context.

Chapter 4

Size compression for Llms

Large Language Models (LLMs) such as GPT-175B have revolutionized natural language processing by demonstrating exceptional capabilities. However, these advancements come with substantial resource requirements for storage and computation, presenting significant challenges. For instance, GPT-175B demands at least 320GB of storage and multiple A100 GPUs for inference. Additionally, the environmental impact of LLMs is considerable due to their significant energy consumption, which contributes to carbon emissions. Therefore, model compression is crucial for making these technologies more sustainable and accessible.

Model compression involves transforming large, resource-intensive models into more compact versions that retain much of the original performance while being more efficient in terms of storage and computation. This not only alleviates the resource requirements but also helps in reducing the environmental impact of deploying LLMs.

We can distinguish 2 compression modes : Model's parameters reduction and Embeddings' dimension reduction.

4.1 Model's parameters reduction

This paper [7] provides an overview of recent advancements in LLM compression, highlighting the importance of ongoing research to enhance the efficiency and applicability of these models in real-world contexts. Model compression is a promising solution to address the current limitations of LLMs in terms of resource requirements and environmental impact. Through continued innovation and evaluation, we can achieve more sustainable and accessible AI technologies.

4.1.1 Compression Techniques

- Quantization reduces the precision of model weights and activations, thereby decreasing the model size and speeding up computations. This can be done

during training (Quantization-Aware Training) or after training (Post-Training Quantization).

  - **Quantization-Aware Training (QAT)**: Techniques like LLM-QAT, PEQA, and QLORA train the model with quantized weights, allowing the model to adapt to lower precision during the training process.
  - **Post-Training Quantization**: This involves quantizing the model weights and activations after training. Methods like GPTQ, AWQ, and SqueezeLLM are prominent examples, often used due to their simplicity and effectiveness.

- Pruning reduces the number of parameters in a model by removing less important neurons or connections.
  - **Unstructured Pruning**: This method removes individual weights or neurons, leading to a sparse model structure. Techniques like SparseGPT and Wanda focus on optimizing this process to achieve high sparsity with minimal performance degradation.
  - **Structured Pruning**: This approach removes entire structural components such as layers or channels, maintaining the overall architecture but simplifying the model. Methods like GUM and LLM-Pruner have shown promise in reducing model complexity while preserving performance.

- Knowledge distillation involves transferring knowledge from a large "teacher" model to a smaller "student" model. This allows the student model to approximate the performance of the teacher model while being significantly smaller.
  - **White-box Distillation**: Access to the teacher model's parameters allows for deeper insights and better performance improvements, as seen in methods like MINILLM and GKD.
  - **Black-box Distillation**: Only the teacher model's predictions are available, which is often sufficient for many applications. Techniques here leverage emergent abilities like In-Context Learning, Chain-of-Thought, and Instruction Following to distill knowledge effectively.

- Other advanced methods include:
  - **Neural Architecture Search (NAS)**: This automates the design of compressed models, optimizing for both performance and efficiency.
  - **Low-Rank Factorization**: Techniques like TensorGPT decompose the weight matrices into lower-rank approximations, significantly reducing the model size while retaining performance.

4.1.2 Evaluation and Benchmarking

Evaluating the effectiveness of compressed models involves several key metrics and benchmarks:

- Metrics
  - **Number of Parameters**: Indicates the total count of learnable weights or variables in the model, impacting both memory usage and computational requirements.
  - **Model Size**: Refers to the disk space or memory footprint needed to store the model, including all weights and biases.
  - **Compression Ratio**: The ratio between the original size of the uncompressed model and the size of the compressed model, indicating the efficiency of the compression.
  - **Inference Time**: Measures the time taken by the model to process and generate responses for input data, crucial for real-time applications.
  - **Floating Point Operations (FLOPs)**: Counts the number of arithmetic operations the model performs, providing an estimate of its computational requirements.
- Benchmarks and datasets
  - **GLUE and SuperGLUE**: Evaluate performance across a range of natural language understanding tasks.
  - **LAMBADA**: Tests context-dependent understanding.
  - **LAMA and StrategyQA**: Assess reasoning abilities.
  - **SQuAD**: Focuses on machine reading comprehension tasks.
  - **BIG-Bench (BBH)**: Covers over 200 NLP tasks, offering a comprehensive evaluation of model performance across diverse challenges.

4.1.3 Challenges and Future Directions

- Performance-Size Trade-off One of the primary challenges is balancing the reduction in model size with the preservation of performance. This requires more theoretical and empirical analyses to understand the underlying trade-offs.
- Dynamic Compression Current methods often rely on manual design and trial-and-error. Integrating automated techniques like NAS could streamline and enhance the compression process.
- Explainability As compressed models are deployed in critical applications, it is crucial to develop explainable compression methods to ensure transparency and reliability.

- Sustainability Reducing energy consumption and carbon emissions remains a priority. Continued research in model compression can make AI technologies more sustainable and accessible.

4.2 Embeddings' dimension reduction

Bibliography

[1] Ghizlane Bourahouat, Manar Abourezq, and Najima Daoudi. Word embedding as a semantic feature extraction technique in arabic natural language processing: an overview. *Int. Arab J. Inf. Technol.*, 21(2):313–325, 2024.

[2] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

[3] Ronen Eldan and Yuanzhi Li. Tinystories: How small can language models be and still speak coherent english? *arXiv preprint arXiv:2305.07759*, 2023.

[4] Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, et al. Textbooks are all you need. *arXiv preprint arXiv:2306.11644*, 2023.

[5] Xin Rong. word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*, 2014.

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[7] Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. A survey on model compression for large language models. *arXiv preprint arXiv:2308.07633*, 2023.

Chapter 5

Annex

5.1 Annex : FlashAttention

FlashAttention [2]

- Motivation :
  - modelling longer distance sequences
  - NLP : large context required to understand books, plays, instruction manuals time series, audio, video, data naturally modeled as sequences of millions steps
- Challenge : How to scale Transformers to longer sequences ?
- Several solutions to memory-efficient attention :
  - One row at a time
  - One block at a time
- Here come FlashAttention : fast and memory-efficient algorithm for exact attention
- Key algorithmic ideas :
  - Tiling : Restructure algorthm to load block by block from HBM to SRAM to compute attention. Compute softmax reduction without access to full input
  - Recomputation : dont store attention matrix from forward, recompute it in the backward.
- The upshot : faster model training, better models with longer sequences

5.2 Annex : High quality data

When They deal with high quality data in the Report of phi3, they refers to another paper from Microsoft Research [4]. BUT this article deals with phi1 who was specialised in Python Coding. **Central Focus on Textbook-Quality Training Data:** The model's success relies on high-quality training data derived from textbooks. This dataset includes under 1 billion tokens of GPT-3.5 generated Python textbooks. These textbooks are synthesized to provide a high-quality source of natural language text interspersed with relevant code snippets. The content is specifically targeted to cover topics that promote reasoning and basic algorithmic skills. Diversity is achieved by setting constraints on the topics and the target audience of the generated textbooks. Pre-training phase : they use CodeTextbook. To fine tune the model, they use CodeExercices. This dataset is a small synthetic collection of Python exercises and solutions, containing less than 180 million tokens. Each exercise consists of a docstring for a function that needs to be completed, aiming to train the model for function completion tasks based on natural language instructions. Generated by GPT-3.5, diversity is achieved by constraining the function names. Explicit decontamination and alternative evaluations are conducted to ensure that problems similar to those from the HumanEval benchmark are not encountered during finetuning. This contrasts with previous approaches using standard sources like The Stack and web-based datasets. **Critique of Standard Code Datasets:**Existing datasets are not ideal for teaching algorithmic reasoning and planning. Issues identified in these datasets include: Lack of Self-Containment: Many code snippets depend on external modules or files, making them hard to understand in isolation. Trivial Content: Many examples consist of non-instructive, trivial code such as defining constants or setting parameters. Buried Algorithmic Logic: Useful algorithmic content is often hidden within complex or poorly documented code. Skewed Distribution: There's an imbalance in the representation of coding concepts, favoring certain topics over others.

Challenge of Creating High-Quality Datasets:

- Ensuring examples are diverse and non-repetitive is a major challenge.
- Diversity in examples is crucial for exposing the model to various coding concepts, skills, and scenarios.

Importance of Diversity:

- Helps the language model learn different ways of expressing and solving problems in code.

– Reduces the risk of overfitting or memorizing specific patterns or solutions.
– Enhances the model's generalization and robustness to unseen or novel tasks.

Inducing Creativity and Diversity:

-Finding the right "trick" to induce creativity and diversity while maintaining quality and coherence is essential. -Inspired by [3], where diversity was achieved by including a random subset of words from a fixed vocabulary in the prompt.

TinyStories: How Small Can Language Models Be and Still Speak Coherent English? Ronen Eldan and Yuanzhi Li

In this work, they presented TinyStories, a synthetic dataset of short stories generated by GPT-3.5 and GPT-4, using words typically understood by 3 to 4-year-olds. TinyStories is used to train and evaluate small language models (SLMs) that are significantly smaller than state-of-the-art models but still produce fluent, consistent, and grammatically correct stories. These models demonstrate reasoning capabilities and diversity in their outputs.

While large models trained on vast internet datasets show impressive capabilities, such datasets are too large for SLMs to effectively capture language complexity. TinyStories enables the study of capabilities like coherent text generation, reasoning, and instruction following on a smaller scale. Training SLMs on TinyStories has revealed behaviors similar to large language models (LLMs), such as scaling laws and trade-offs between model width and depth.

Moreover, SLMs trained on TinyStories offer higher interpretability, allowing visualization and analysis of their attention and activation patterns to understand story generation and comprehension. The study provides evidence that models trained on TinyStories can produce genuinely new stories, not just replicate text from the dataset. They have also introduced a new paradigm for the evaluation of language models, which uses GPT-4 to grade the content generated by these models as if those were stories written by students and graded by a (human) teacher.

Conclusion : We cannot determine if data is of good quality or not; it is up to us to assess whether we think the data is of good quality. There are no predefined criteria.

5.3 Annex 2 : how do they determine the number of dimensions ?

https://www.baeldung.com/cs/dimensionality-word-embeddings

Parameters do not hold any inherent meaning. Instead, they function collectively, working in unison to map intricate relationships between words and phrases in the training data. The **dimensionality** of word embedding refers to the number of dimensions in which the vector representation of a word is defined. This is typically a fixed value determined while creating the word embedding. The dimensionality of the word embedding represents the total number of features that are encoded in the vector representation.

Deciding on the appropriate dimensionality for a word embedding depends on several factors, such as the size and nature of the dataset we are using, the specific NLP task we are working on, and the computational resources available to us.//

Size of the Dataset

- Larger datasets can support higher-dimensional embeddings due to more training data.
- Datasets with less than 100,000 sentences: Benefit from lower-dimensional embeddings (50-100 dimensions).
- Larger datasets: Benefit from higher-dimensional embeddings (200-300 dimensions).

NLP Task

- Task requirements impact embedding dimensionality: High semantic accuracy tasks (e.g., sentiment analysis, machine translation): Benefit from higher-dimensional embeddings. Easier tasks (e.g., named entity recognition, part-of-speech tagging): May not require high-dimensional embeddings.

Computational Resources

- -Higher-dimensional embeddings require more memory and processing power

In conclusion, to find the Right Number of Dimensions, we need to Experiment with different dimensionalities and evaluate model performance on a validation set and adjust dimensionality to find the optimal balance between semantic accuracy and computational efficiency for the specific use case. In phi3-mini, each vectors has 3072 dimension..