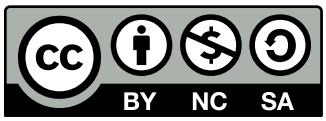
Aalto University
School of Science

Master's programme in Computer, Communication and Information Sciences

Internet-scale Topic Modeling using Large Language Models

Roope Kajoluoto

Master's Thesis
2024



Aalto University
School of Science

Author Roope Kajoluoto

Title Internet-scale Topic Modeling using Large Language Models

Degree programme Computer, Communication and Information Sciences

Major Machine Learning, Data Science and Artificial Intelligence

Supervisor Prof. Pekka Marttinen

Advisor D.Sc. (Tech.) Janne Toivola, L.Sc. (Tech.) Kenneth Oksanen

Date 22.04.2024 **Number of pages** 80+5 **Language** English

Abstract

Topic models attempt to uncover latent themes underlying a corpus. Traditionally, topics have consisted of n-grams selected from a set of training documents. However, these topic representations are often overly restrictive in the domain of internet data. Contemporary topic models also face significant challenges with short documents, multilinguality, noisiness and topic specificity, especially when the breadth of topics increases. These challenges, among others, make previous models nonviable for the task of truly generic topic modeling. This thesis presents Topic Large Language Model (TLLM), a 7 billion parameter Large Language Model (LLM) that has been fine-tuned for topic modeling. The dataset used for training TLLM is a mixture of manual labeling and knowledge distillation from a larger LLM. Combined with the training procedure, this results in a model that is a general expert instead of a narrow specialist. Due to the usage of a LLM, the model is inherently multilingual and exhibits levels of general knowledge not present in prior approaches. TLLM consistently produces less unique topics when compared to its nearest counterparts while remaining accurate. This makes it more effective for analyzing large datasets and demonstrates the model's capabilities at separating important pieces of information from irrelevant ones. Our human evaluation shows that TLLM outperforms GPT-3.5-turbo both with and without prompt engineering. In the same evaluation, TLLM is approximately tied with a model 10 times its size, while producing the least failures out of all the models tested. We show that topic modeling of universal themes is possible with both general instruction following LLMs and LLMs trained for topic modeling. However, TLLM greatly improves on other LLMs in terms of predictability, topic granularity and computational efficiency, making it the first model capable of internet-scale topic modeling. As far as the authors are aware, this is the first work to fine-tune LLMs for the task of topic modeling.

Keywords Natural Language Processing, Large Language Models, Topic Modeling, Self-Supervised Learning, Fine-Tuning, Prompt Engineering, Low-Rank Adaptation

Aalto-yliopisto
Perustieteiden
korkeakoulu

Tekijä Roope Kajoluoto

Työn nimi Internetin mittakaavassa toteutettava aihemallinnus suurilla kielimalleilla

Koulutusohjelma Computer, Communication and Information Sciences

Pääaine Machine Learning, Data Science and Artificial Intelligence

Työn valvoja Prof. Pekka Marttinen

Työn ohjaaja TkT Janne Toivola, TkL Kenneth Oksanen

Päivämäärä 22.04.2024 **Sivumäärä** 80+5 **Kieli** englanti

Tiivistelmä

Aihemallit analysoivat luonnollisen kielen latentteja aiheita. Perinteisesti näitä aiheita ovat edustaneet koulutusdokumenteista valikoidut n-grammit. Internet-dataa käsiteltäessä tämä aiheiden määritelmä on kuitenkin usein liian rajoittava. Nykyiset aihemallit kohtaavat myös merkittäviä haasteita lyhyiden, monikielisten ja epäformaalien dokumenttien kanssa, jonka lisäksi niiden aiheiden täsmällisyyttä on vaikea hallita aiheiden määrän kasvaessa. Nämä haasteet, yhdessä muiden kanssa, tekevät nykyisistä lähestymistavoista riittämättömiä aidosti geneeriseen aihemallinnukseen. Tämä työ esittelee TLLM:n (Topic Large Language Model), seitsemästä miljardista parametrista koostuvan suuren kielimallin joka on koulutettu aihemallinnukseen. TLLM:n koulutusdatan luomisessa on yhdistetty suuremman kielimallin osaamista ja manuaalista annotointia. Yhdistettynä koulutusprosessiin, tuloksena on geneerinen asiantuntija perinteisen, kapean spesialistin sijaan. Suuren kielimallin käytön ansiosta malli on luonnostaan monikielinen ja yleistiedoiltaan huomattavasti aikaisempia malleja edistyneempi. TLLM tuottaa johdonmukaisesti vähiten uniikkeja aiheita verrattuna sen lähimpiin vastineisiin, samalla ylläpitäen aiheiden oikeellisuuden. Tämä helpottaa suurten datamassojen analysoimista ja tarjoaa todisteita siitä, että TLLM kykenee erottelemaan dokumenttien tärkeät yksityiskohdat tarpeettomista. Ihmisevaluaatiomme osoittaa, että TLLM menestyy paremmin kuin GPT-3.5-turbo sekä normaalilla että optimoidulla kehotteella. Samassa evaluaatiossa TLLM on pisteiltään keskimäärin tasoissa itseään kymmenen kertaa suuremman mallin kanssa, samalla tuottaen vähiten suoranaisia epäonnistumisia kaikista koestetuista malleista. Osoitamme, että yleismaailmallisten aiheiden mallinnus on mahdollista sekä yleispätevillä että tarkoitukseen koulutetuilla kielimalleilla. TLLM on kuitenkin muihin kielimalleihin verrattuna huomattavasti ennustettavampi, nopeampi ja tuottaa helpommin analysoitavia aiheita. Tämä tekee siitä ensimmäisen mallin, joka kykenee internetin mittakaavassa toteutettavaan aihemallinnukseen. Kirjoittajien parhaan tietämyksen mukaan tämä on ensimmäinen tutkimus, joka hienosäätää kielimalleja aihemallinnukseen.

Avainsanat Luonnollisen kielen käsittely, Suuret kielimallit, Aihemallinnus, Itseohjattu oppiminen, Kielimallin hienosäätö, Kehotteiden suunnittelu, Matala-asteinen mukautus

Contents

Symbols and abbreviations

Symbols

$\boldsymbol{x}$	Bold lowercase letters indicate vectors
$\boldsymbol{X}$	Bold uppercase letters indicate matrices
$x_i, X_{i,j}$	Selecting a value from a vector or a matrix by index
$\boldsymbol{x}_{i:j} \boldsymbol{X}_{i:j,k}$	Selecting a range of indices from a vector or a matrix

Operators

$p(x), p(X = x)$	The probability of an event
$\log(x), \ln(x), \log_n(x)$	Base 10 logarithm, natural logarithm and base n logarithm
$\|\|\boldsymbol{x}\|\|, \|\|\boldsymbol{x}\|\|_n$	Euclidean norm (2-norm), n-norm
$\|\boldsymbol{x}\|$	The size of a vector, number of entries
$S_c(\boldsymbol{x}, \boldsymbol{y}) = \frac{\boldsymbol{x} \cdot \boldsymbol{y}}{\|\|\boldsymbol{x}\|\|\|\|\boldsymbol{y}\|\|}$	Cosine similarity
$D_c(\boldsymbol{x}, \boldsymbol{y}) = 1 - S_c(\boldsymbol{x}, \boldsymbol{y})$	Cosine distance
$\text{softmax}(\boldsymbol{x}) = \frac{e^{x_i}}{\sum_{j=1}^{\|\boldsymbol{x}\|} e^{x_j}}$	The softmax operator

Abbreviations

BERT	Bidirectional encoder representations from transformers
CoT	Chain-of-thought
DTM	Document term matrix
GPT	Generative pre-trained transformer
HBM	High bandwidth memory
ICL	In-context learning
LDA	Latent Dirichlet allocation
LLM	Large language model
LSA	Latent semantic analysis
LSI	Latent semantic indexing
LSTM	Long short-term memory
LoRA	Low-rank adaptation
MHA	Multi-head attention
NLP	Natural language processing
PEFT	Parameter-efficient fine-tuning
QA	Question answering
RLHF	Reinforcement learning from human feedback
RNN	Recurrent neural network
RoPE	Rotary position embedding
SFT	Supervised fine-tuning
SRAM	Static random-access memory
SVD	Singular value decomposition
TDM	Term-document matrix
TF-IDF	Term frequency-inverse document frequency
TLLM	Topic large language model
ToT	Tree-of-thoughts
VQA	Visual question answering

1 Introduction

In the context of information retrieval, topic modeling comprises of approaches that infer the latent themes of documents [1]. This provides an automatic way to organize, summarize, visualize or otherwise understand large quantities of unstructured data. In essence, given a collection of N natural language documents $\boldsymbol{d} = \{d_1, ..., d_N\}$, the goal is to infer their underlying topics $z = \{z_1, ..., z_N\}$. A topic is an abstraction of a semantic concept that is also human-interpretable and uncovered in an unsupervised fashion. Topic modeling is applicable to a multitude of data formats, such as gene sequences [83] and images [136], but this thesis will focus solely on data that is in the form of written natural language. Furthermore, in this thesis, any sequence of written natural language is referred to as a *document*: books, social media posts, news articles, reviews or even online comments. This is similar to the historical definition by Jurafsky et al., which includes all units of text indexed in a system and available for retrieval [70].

Previous work has mostly focused on topics that are composed of different n-grams included in a collection of training documents [1, 15, 53, 31]. Furthermore, some authors define topic models as extracting groups of words from documents [13]. However, topics that are represented by collections of n-grams are often difficult to interpret and require domain knowledge. For instance, names of people, products and species are often likely candidates for topics, as they are highly specific to certain types of documents. Although thematic to specific types of documents, these topics do not provide insight unless the human interpreter is familiar with their meanings. Collections of n-grams can also be overly restrictive. For instance, they might provide poor results if the documents are written in informal speech, are noisy or do not contain repeating instances of the same, information-rich words. Additionally, a large proportion of internet documents are short, low quality and contain no latent topic. This can pose further challenges to models that rely on an accurate document clustering to produce topics, such as BERTopic [53]. Consequently, current topic models face significant obstacles when applied to internet data, instead of the more sanitized datasets often used for their evaluation. This thesis adopts a more loose definition of topics: a categorization of a document in natural language that is at most 3 words long. In essence, a topic is a heavily compressed summary of the document's core message.

With this definition in mind, we study using Large Language Models (LLM) for topic modeling. Our main contributions are as follows: (1) Outline the challenges in applying prior topic models to internet material. (2) Provide a process for creating fine-tuning datasets and training topic modeling LLMs with them. Its steps are designed to maximize the breadth of topics understood by the resulting model and to allow for modifying the granularity of the resulting model's topics. (3) Show that topic modeling of universal themes is possible with both purpose-made LLMs and LLMs trained for generic instruction following, given a sufficiently optimized prompt and computational resources. (4) Present Topic Large Language Model (TLLM), a 7 billion parameter LLM trained for topic modeling. Due to its computational efficiency, multilinguality, topic granularity and general knowledge, TLLM is the first model

capable of true internet-scale topic modeling. As far as the authors are aware, this is the first work to fine-tune LLMs for topic modeling.

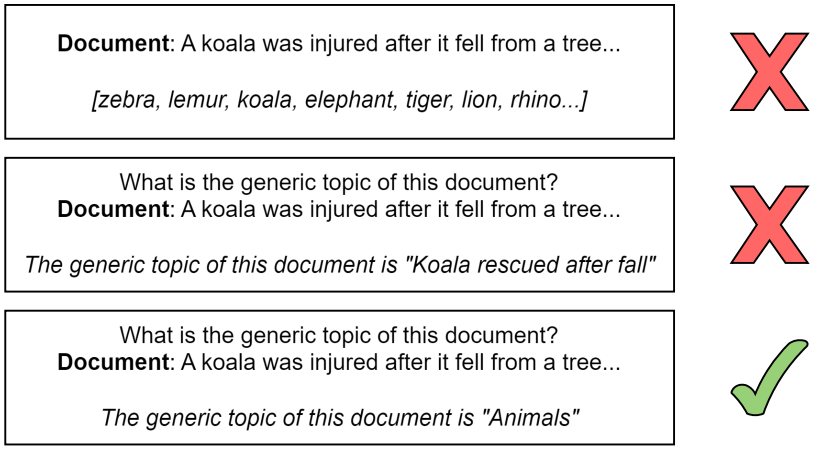


Figure 1: Examples of possible generations with different topic modeling approaches. (Top) Traditional topic models, such as LDA. (Middle) Contemporary LLMs, such as GPT-3.5-turbo. (Bottom) TLLM.

1.1 Key requirements

When defining an optimal internet-scale topic model, there are a few key requirements. Although many topic models have been proposed and are widely used today, none of them fit all these criteria at once. These requirements are in addition to general criteria that also applicable in other domains, such as topic accuracy.

(Generic) In order to analyze vast amounts of internet data, the generated topics should be generic. This is to say, generated topics should reflect the topic distribution of the internet as a whole and should not focus only on a narrow subset, such as medicine [196] or customer reviews [82]. Furthermore, topics should not be too specific or too generic. Overly specific topics lead to an unreasonable amount of unique topics, making for an ineffective analysis tool. It must be possible to analyze large quantities of internet data without having to investigate an unreasonable amount of topics. Conversely, overly generic topics lead to dissimilar documents receiving the same topic, in which case the topics seize to be trustworthy. For instance, the documents "A crocodile was seen in the lake" and "An alligator was seen in the lake" should receive the same topic. The generated topic should be related to animals in general, and not crocodiles or alligators specifically. Clearly, if the goal is to understand large corpuses as a whole, generating a unique topic for all different animals would be excessive.

(Compute-efficient) The model needs to be fast on relatively modest hardware. Not only does this allow for basic inference to be run with only a moderate hardware investment, but it also allows for scaling in a way that is not prohibitively expensive. As the model must be applicable to analysis of internet-scale data streams, the compute efficiency and inference speed of the model are crucial.

(Multilingual) As internet data is inherently diverse in terms of language, the model should be multilingual. In the early 2000s, only a third of internet users were English speakers [90]. More recently, Buck et al. studied a 35.23 TiB subset of the Common Crawl dataset, and found 73 languages with at least 1 GiB of uncompressed text [23]. Furthermore, the authors note that even Finnish, which made up only 0.17% of the dataset, accounted for 47.73 GiB of text when uncompressed. Training a separate topic model for every language of interest, while perhaps possible, would be time consuming and resource intensive. In this thesis, the model's proficiency in a language is required to be relative to its proportional frequency in internet data. This proportion is estimated from the Common Crawl dataset's statistics [28]. In short, the model should have an excellent understanding of English, an adequate understanding of languages such as German and Spanish, and be satisfactory at less spoken languages such as Finnish. In this thesis, we do not require the model to generate topics in languages other than English; it suffices for the model to be able to generate English topics for documents written in other languages.

(Streamable) The model must be able to generate topics for documents in isolation. This is to say, for the inference dataset $\boldsymbol{d} = \{d_1, ..., d_N\}$ the model must be able to generate $z_i \in \boldsymbol{z}$ using only d_i, without access to $d_j, \exists j \neq i$. This is due to the real-time nature of the internet, more data is constantly being created. The dataset is never static, unlike in many other domains.

(Length agnostic) The problems raised by document length are twofold: documents can be either too long or too short to be modeled effectively. Firstly, the model should be able to process documents that are at least in the range of thousands of words. This is a significant limitation with some topic models. For instance, BERTopic can only process documents of up to 512 tokens. Although there are methods for artificially increasing this limit, they are often ineffective and complicate inference. Secondly, the model should be able to produce topics for documents that only contain a few words. The internet has led to a rise of short-form textual content [99], which contains more sparse information when compared to longer documents [139]. It is known that this causes problems for current topic modeling approaches [134].

This thesis is structured as follows. Section 2 provides a literature review in the areas of topic modeling and large language models. Section 3 presents TLLM along with specifics related to its training. Section 4 examines TLLM and evaluates it against comparable models. Section 5 provides a short summary related to the main findings of this thesis and closing remarks.

2 Literature review

The analysis of latent themes in natural language has been studied at least since the 1990s [31]. Traditional approaches commonly use probabilistic models with varying degrees of complexity. From simply analyzing word frequencies to modeling the generative process behind documents, these approaches remain relatively popular [1]. In recent years, LLMs have shown exceptional promise on a multitude of natural language tasks, but have not seen wide adoption in the field of topic modeling. This section will cover related work in the fields of Natural Language Processing (NLP), topic modeling and LLMs.

Section 2.1 covers projecting text into a vector space. This is a requirement of all models discussed in subsequent chapters. As different types of models create vector representations using different methods, an overview of these different methods' strengths and weaknesses is provided. Section 2.2 covers previous work in the field of topic modeling. Influential or otherwise relevant models are discussed, which also works to motivate the differing approach taken in this thesis. Section 2.3 provides a brief introduction to language models as a whole. This discussion is continued in the subsequent section 2.4, which gives an overview of LLMs. As TLLM is implemented using a LLM, this section also works to motivate some of its design decisions.

2.1 Text representations

For documents to be processed by the topic models discussed in subsequent chapters, they need to be projected into a vector space. There are many ways in which this is commonly achieved. Traditional approaches generally produce a sparse vector representation in a discrete vector space. This is in contrast to more recent, often neural network based approaches such as GloVe [132], that produce a dense vector in a continuous vector space. The latter vector representations are commonly referred to as text embeddings, or simply embeddings. Embeddings can also be constructed by more conventional means. These include, among others, applying dimensionality reduction techniques such as Principal Component Analysis (PCA) [129, 62] on word co-occurrence matrices [86] and probabilistic models [45]. Unlike traditional, frequency-based approaches where words are only considered with respect to their occurrence frequencies, embeddings also account for context. As pointed out by Firth [40]: "You shall know a word by the company it keeps". This section covers notable approaches to projecting text into a vector space for the purpose of topic modeling.

2.1.1 Polysemy and synonymy

There are two prominent challenges that a good vector representation must overcome: polysemy and synonymy. *Polysemy* stands for the ability of words to have many, sometimes even mutually exclusive, meanings [70]. One example of a polysemous word is the word *paper*, as it can refer to the material but also the newspaper made out of it. Polysemy is closely related to *homonymy*, which stands for two different words having the same spelling and pronunciation, but different meanings. As pointed out

by Jurafsky et al., the words "bank" (*financial institution*) and "bank" (*sloping mound*) are an instance of homonyms [70]. In essence, the difference between homonymy and polysemy is whether the words are also related semantically. For a more full example, take the word *set*. The word, depending on context, can be a verb or a noun. It can be used to describe a collection of items, the act of setting a clock, the sun going below the horizon, and more. Some of these meanings are semantically distinct enough to be considered homonyms, while others are instances of the same, highly polysemous word. Many classical vector projection methods interpret words in isolation, without accounting for the context in which they appeared. Clearly, such methods face significant challenges when processing highly polysemous and homonymous words.

Synonymy, on the other hand, stands for two different words having a similar or even the exact same meaning. The word *lucrative* could be used synonymously with the word *profitable* and the word *couch* could be used synonymously with the word *sofa* [70]. However, synonymy is complicated by also being reliant on context. As a consequence of polysemy, two words can have certain meaning that are synonymous with each other but others that are even antonymous. For instance, the word *argument* can be synonymous with the word *dispute* when it stands for an expression of differing opinions. However, if the word is used to refer to a statement in support of a belief, it is no longer synonymous with *dispute* and is now synonymous with words such as *explanation*. Therefore, synonymy is an important factor in generating accurate vector representations, but it needs to be considered together with polysemy.

2.1.2 Frequency-based representations

Perhaps the simplest way to project documents into a vector space is through the Bag-Of-Words (BoW) approach. BoW vectors are generated by counting the frequencies of all known words contained within the document, and placing them in a vector [9]. The indices of different words are inferred from a static vocabulary of size N_v: $\boldsymbol{v} = \{v_1, ..., v_{N_v}\}$. For example, with a vocabulary $\boldsymbol{v} = \{\text{a, dog, cat, and, nice, giraffe}\}$, the document "a nice dog and a nice cat" would have the vector representation of $[2, 1, 1, 1, 2, 0]$. Clearly, BoW vectors only capture multiplicity and discard any notion of ordering or grammar present in the original document. A matrix created with this approach, where each row represents a document, each column represents a word and the entries represent different word frequencies is called a Document-Term Matrix (DTM) [121]. The transpose of this matrix is also commonly used, and is referred to as the term-document matrix (TDM) [9, 4]. The entries of these matrices can also be something other than words, such as bigrams, subwords or morphemes. BoW vectors are intuitive and easy to implement, but can have severe drawbacks.

Firstly, the vocabulary is static, so either unseen words can not be represented or previously created vector representations have to be updated when encountering a new word. The former entails discarding possibly crucial information and the latter makes parallelization difficult. Secondly, BoW vectors created for large collections of documents are often extremely sparse, which in turn leads to a similarly sparse DTMs. This can lead to negative effects associated with the curse of dimensionality [74] and

increases the computational complexity of subsequent vector and matrix operations. Thirdly, these vectors often provide misleading measures of document similarities. Say we have a dataset of three documents: $\boldsymbol{d}$ = {nice dog, good cat, outer space} and a vocabulary of $\boldsymbol{v}$ = {nice, good, dog, cat, outer, space}. Intuitively, the first document shares more similarities with the second than with the third. However, the documents' BoW vectors are $[1, 0, 1, 0, 0, 0]$, $[0, 1, 0, 1, 0, 0]$ and $[0, 0, 0, 0, 1, 1]$. All three vectors are orthogonal to each other. Consequently, common distance metrics such as cosine distance would claim that all three documents are equally dissimilar. In this sense, BoW vectors lose much of the information contained in the original documents. Lastly, words that are generally more common will have a high frequency, even if they are not used any more than is typical. Words with the highest frequencies will often be words such as *the*, solely because they are used to structure natural language, regardless of subject matter. This further obscures any measurements of document similarity, as the highest frequency words are not a good indicator of any latent topic.

The challenge of different words having different typical frequencies can be alleviated in a few ways. As pointed out by Leskovec et al., hundreds of the most common words in English are often removed from documents before they are processed [91]. These words are known as "stop words", and usually include the previously mentioned word *the* along with others, such as *but*. Another common modification to BoW vectors is to represent their entries with term frequency-inverse document frequency (TF-IDF) weighted values. In a TF-IDF vector, the frequency $f_{t,d}$ of a word t for document d is weighed by its occurrence rate in the whole dataset $\boldsymbol{d}$ [91]:

$$\begin{aligned} \text{tfidf}(t, d, \boldsymbol{d}) &= \text{tf}(t, d) \cdot \text{idf}(t, \boldsymbol{d}) \\ &= \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \cdot \log \frac{|\boldsymbol{d}|}{|\{d \in \boldsymbol{d} : t \in d\}|} \end{aligned} \tag{1}$$

There are of course many ways to alter the definitions of *term frequency* and *inverse document frequency* to better suit different tasks. For instance, term frequency can be scaled logarithmically to reflect that the repetition of words does not indicate a linear increase in importance [70]. However, the foundational intuition behind the weighing mechanism is that for a word to have a high value in a TF-IDF vector, it requires both a high level of importance for that specific document and a low level of importance in others. Therefore, TF-IDF heavily reduces the values of generic words that are common in all documents. This is desirable in topic modeling, where information that does not convey information about the document's latent topics can be considered redundant.

These modifications, while useful, do not directly assess many of the challenges faced by the BoW approach. Although they can assist in measuring document similarities, they do not alleviate the effects of synonymy and polysemy. Words are still represented by simple (weighted) frequencies, and therefore context is discarded. The vocabulary remains static and the removal of stop words has not changed the document vector sizes or sparsities significantly.

2.1.3 Embeddings

Embedding models are a newer approach to generating vector representations from both individual words and entire documents [173]. As mentioned previously, a key distinction between embeddings and traditional vector encodings is related to context. Whereas traditional frequency-based approaches assign static vector representations to words, embedding models also consider the context in which the word appeared. Concretely, the same word can have a different embeddings depending on the words that follow and precede it. Morris et al. found that modern neural embeddings can be reconstructed back into the original text with a high confidence [117]. This suggests that document embeddings are able to condense information exceptionally well and maintain contextual information.

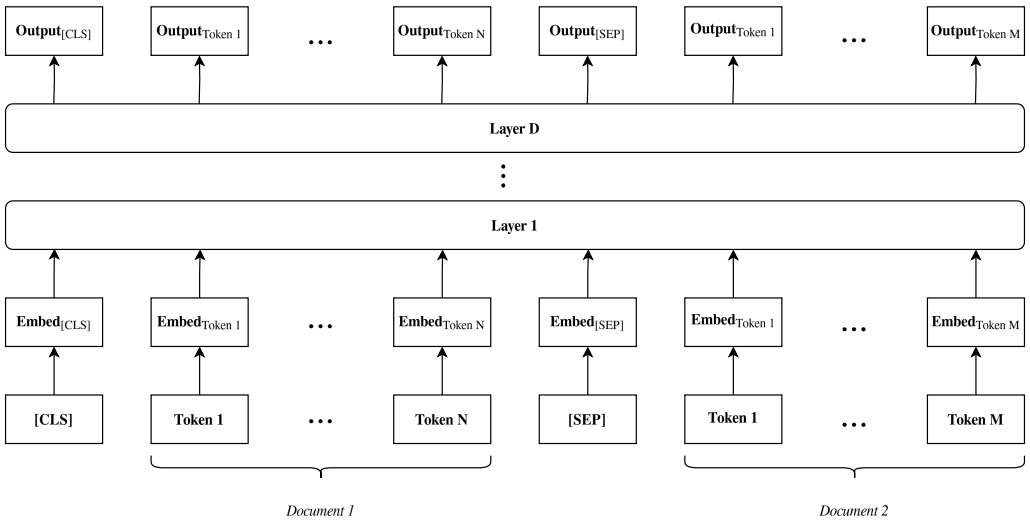


Figure 2: A visualization of BERT, when used for document similarity estimation.

Bidirectional Encoder Representations from Transformers (BERT) [34] and RoBERTa [105] set new benchmarks in sentence similarity detection. The BERT architecture, when used for document similarity estimation, is visualized in Fig. 2. Due to BERT's cross-encoder structure, fine-tuning it for calculating document similarities is straightforward. The model input consists of the documents separated by the separator token ([SEP]) and preceded by the classification token ([CLS]). The most common ways to estimate document similarities with BERT are to average the outputs of the final layer or to use the output of the classification token directly [146]. For the latter case, after a forward pass is performed, the output of the classification token is interpreted as a similarity score for the documents. A loss is calculated based on its correctness and used to train the model through backpropagation [73]. During inference, two documents can be passed through the model, leading to an estimate of their similarity. It is worth highlighting that this process does not create comparable vector representations, but generates a similarity estimate directly. This makes it a simple and intuitive extension of a pretrained BERT model. However, the primary challenge with this approach is the computational complexity of calculating

similarities for large corpuses. When similarities are calculated for multiple document pairs, every pair of documents has to be considered individually. This means that the number of calculations increases roughly proportionally to the square of the number of documents, leading to poor scalability. As noted by Reimers et al. [146], finding the document pair with the highest similarity in a dataset of $n = 10000$ documents requires $\frac{n(n-1)}{2} = 49995000$ forward passes of the model. For small datasets this is an inconvenience, while for large datasets it can be prohibitively expensive.

Instead of approximating similarities directly, a more generic approach is to generate document embedding vectors. This way, a simple distance metric can be used to arrive at a similarity score between any two of them. Although BERT and RoBERTa are trained on text pairs, a single document can also be used as input. After the forward pass, a fixed-size embedding can be derived by either averaging the token outputs or using the CLS token [138, 195]. As the underlying model was trained to capture semantic meaning from documents to be used in a downstream task, the idea is that this information could also be used directly. However, these embeddings have been found to be poor for calculating semantic similarities between documents [146]. A more common approach to achieve an embedding for a document is to pass its individual words through a word embedding model such as GloVe [132] or Word2Vec [114, 115], and then average or sum the results. As shown by Reimers et al. [146], averaging GloVe embeddings often leads to more accurate document embeddings than using the output layer of a BERT model.

An approach that aims to directly create neural document embeddings was proposed by Reimers et al. in 2019, called Sentence-BERT (SBERT) [146]. The training fine-tunes a pretrained BERT or RoBERTa model specifically for generating sentence embeddings. Training data consists of a million sentence pairs that have been annotated with a textual entailment label: entailment, neutral or contradiction [19, 183]. During training, three distinct objective functions are used to update the model: the classification objective function, the regression objective function and the triplet objective function. SBERT can be trained in less than 20 minutes, which is a considerable reduction to other sentence embedding models that start from a random initialization [68, 75]. The resulting sentence embeddings can be compared with cosine distance, dot-product, Manhattan distance or Euclidean distance, depending on the specific model used. SBERT embeddings provided a new benchmark for sentence embedding performance, and were subsequently used by BERTopic. In later studies, different SBERT models have been trained with more data and improvements to the training procedure. These studies have produced models with new capabilities, such as multilinguality [147]. However, embeddings created by SBERT are not always accurate and the models have some notable drawbacks, such as their limits on document length. The original BERT model has a context length of 512 tokens [34], meaning that SBERT embeddings can only be generated for relatively short documents.

Much of the subsequent research into sentence embeddings has also utilized BERT. For example, Saleh et al. [61] apply a pooling function to multiple layer representations instead of only the last one. The authors observed an increase in embedding quality using this approach on multiple baseline models, including SBERT. More concretely,

for an input sequence of N tokens, $\boldsymbol{w} = \{w_1, ..., w_N\}$, a BERT forward pass yields the tensor $\boldsymbol{T} \in \mathbb{R}^{L' \times N \times H}$. Here, H is the hidden size of the model, $L' = L + 1$ and L is the number of intermediate layers, as layer 0 is also included. Then, a layer set $\boldsymbol{s}$ is selected, for which a pooling function is applied. The classical approach of averaging the last layer representations would entail having the layer set $\boldsymbol{s} = \{L\}$. However, the authors show increased accuracy of sentence embeddings when allowing this set to change, depending on which combination of layers performs best.

2.2 Topic modeling

Topic modeling involves uncovering the underlying latent themes in natural language. The field of topic modeling could be seen to have started with Latent Semantic Indexing (LSI), also known as Latent Semantic Analysis (LSA) [31, 1]. Since then, many different algorithms have made progress in improving modeling capacity. Consequently, modern topic models encompass a broad range of different use cases and approaches. This chapter covers related work in the field of topic modeling.

2.2.1 LSA

Latent Semantic Analysis (LSA) [31] is perhaps the most foundational topic model. LSA takes as input a TDM $\boldsymbol{M} \in \mathbb{R}^{m \times n}$ that is optionally TF-IDF weighted, where m is the size of the vocabulary and n is the number of documents. LSA then performs Singular Value Decomposition (SVD), which factorizes the matrix $\boldsymbol{M}$ into the product of three separate matrices:

$$\begin{aligned} \boldsymbol{M} &= \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T \\ \text{where } \boldsymbol{U} &\in \mathbb{R}^{m \times m}, \\ \boldsymbol{\Sigma} &\in \mathbb{R}^{m \times n}, \\ \boldsymbol{V} &\in \mathbb{R}^{n \times n} \end{aligned} \tag{2}$$

Here, $\boldsymbol{U}$ and $\boldsymbol{V}$ are real orthogonal matrices while $\boldsymbol{\Sigma}$ is a rectangular diagonal matrix with only non-negative values. LSA is closely related to PCA, which can be performed by applying SVD on a covariance matrix [9, 160]. When SVD is applied to a complex instead of a real matrix, $\boldsymbol{U}$ and $\boldsymbol{V}$ are complex unitary matrices. However, as TDMs are always real matrices, the notation of $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ is used here for clarity instead of the more general $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$, where $\boldsymbol{V}^*$ stands for the conjugate transpose of $\boldsymbol{V}$. Next, the $\boldsymbol{\Sigma}$ matrix is truncated to a square matrix with t columns and rows: $\boldsymbol{\Sigma}_{1:t,1:t} \in \mathbb{R}^{t \times t}$. As the values in this matrix are ordered based on magnitude, this effectively discards all but the t largest values. Here, the hyperparameter t represents the number of latent topics in the corpus. Consequently, the $\boldsymbol{U}$ and $\boldsymbol{V}$ matrices are truncated to t columns: $\boldsymbol{U}_{1:m,1:t} \in \mathbb{R}^{m \times t}$ and $\boldsymbol{V}_{1:n,1:t} \in \mathbb{R}^{n \times t}$. This produces the end result of $\boldsymbol{M}_t = \boldsymbol{U}_{1:m,1:t}\boldsymbol{\Sigma}_{1:t,1:t}\boldsymbol{V}^T_{1:n,1:t}$, where $\boldsymbol{M}_t \in \mathbb{R}^{m \times n}$ is an approximation of $\boldsymbol{M}$ with rank t.

With this SVD factorization, the similarity of two documents can be quantified by calculating the cosine similarity of their respective columns in $\boldsymbol{M}_t$. Directly

comparing the columns in the original TDM would be problematic due to polysemy and matrix sparsity, which are alleviated by using the low-rank approximation $\boldsymbol{M}_t$. More concretely, similar words are merged into one, abstract vector representation. Comparing how much each document relates to these non-interpretable representations is more stable than comparing their word frequencies directly.

In summary, LSA is concerned with realizing the relationships between words in a corpus of text. In relation to topic modeling, LSA allows for clustering and calculating document similarities using these abstract topic representations. However, LSA does not produce human interpretable topics, as it is only used to approximate similarities. The approach is not streamable, as performing SVD requires access to a static corpus of all the documents to be analyzed. Furthermore, the approach does not guarantee the mitigation of polysemy and can still suffer from its effects.

2.2.2 Mixture of unigrams

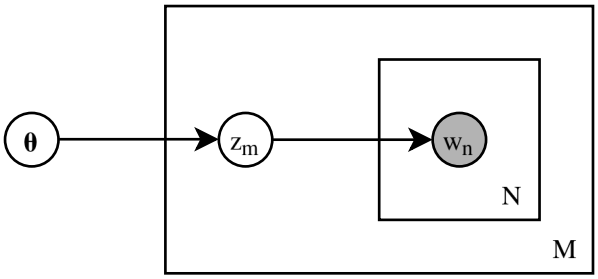


Figure 3: Mixture of unigrams

The mixture of unigrams provides a simple approach to latent topic discovery. In this model, each document is assumed to belong to one in a set of possible topics, each of which defines a multinomial distribution over the word vocabulary. Document generation entails choosing a topic z and then generating mutually independent unigrams from its word distribution until the document length reaches a termination point. Consequently, the marginal probability of any document $\boldsymbol{w} = \{w_1, ..., w_N\}$ is as follows [15]:

$$p(\boldsymbol{w}) = \sum_z p(z) \prod_{n=1}^{N} p(w_n|z) \tag{3}$$

Nigam et al. define a further variable $\boldsymbol{\theta} \in \mathbb{R}^k$, which is a vector defining the multinomial distribution over k topics [122]. With this extension, the marginal posterior probability of documents becomes:

$$p(\boldsymbol{w}|\boldsymbol{\theta}) = \sum_z p(z|\boldsymbol{\theta}) \prod_{n=1}^{N} p(w_n|z, \boldsymbol{\theta}) \tag{4}$$

The mixture of unigrams approach suffers from the assumption that documents are only comprised of one topic [15]. Furthermore, as pointed out by the authors, many

of the model's assumptions are also violated in real world data. These include the assumptions of word independence and independence between a document's length and its topic.

2.2.3 pLSA

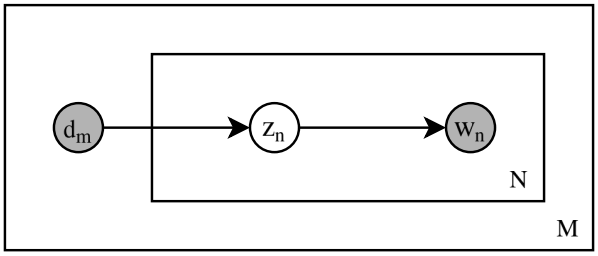


Figure 4: probabilistic Latent Sematic Allocation (pLSA)

Similarly to LSA, Probabilistic Latent Semantic Analysis (pLSA) [60] is a method for analyzing relationships between documents and the words they contain. Unlike LSA, the document generation process is explicitly modeled in pLSA. In pLSA, every document is assumed to have been generated one word at a time, by first sampling a topic from a multinomial topic distribution and then sampling a word from that topic's multinomial word distribution. In this way, the latent topics of the document generation process are explicitly encoded as the hidden variables. This provides interpretability, as each document maintains a topic distribution, which in turn maintains a word distribution. This is in contrast to LSA, where the resulting topic vectors have no such easily interpretable meaning and the mixture of unigrams, where documents can only exhibit one topic. With the pLSA model, the marginal probability of a document $\boldsymbol{w} = \{w_1, ..., w_N\}$ is:

$$p(\boldsymbol{w}) = p(d) \sum_{z} p(z|d) \prod_{n=1}^{N} p(w_n|z) \tag{5}$$

Here, d is a dummy index for the document in the training corpus. It should be noted that pLSA requires estimating a large number of variables that also grows linearly with the size of the corpus: $kM + kV$. Here, k is the number of topics, M is the size of the corpus and V is the size of the vocabulary. This leads to added computational complexity and possibility of overfitting [15]. Furthermore, as pointed out by Blei et al., a major difficulty with pLSA is that there is no natural way to assign topic to documents which were not present during training [15]. In essence, there is no clear method for performing post-training inference with the pLSA approach without retraining the model.

2.2.4 LDA

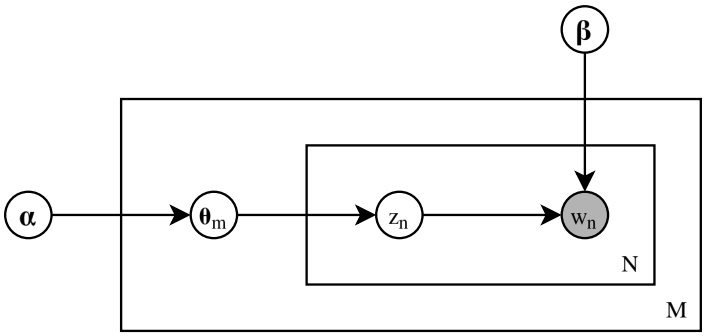


Figure 5: Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation (LDA) [15] is one of the most widely studied probabilistic topic models [1]. LDA assumes that each document is a mixture over a set of possible topics, whereas topics are an infinite mixture over an underlying set of word probabilities. Suppose we have a vocabulary of size V and a set of k topics. Words $w \in \mathbb{N}^V$ and topics $z \in \mathbb{N}^k$ are one-hot encoded vectors over the vocabulary and topic sets. As these vectors only have a single component equal to one and all others equal to zero, we will denote them with unbolded characters. This way, documents are sequences of words $\boldsymbol{w} = \{w_1, ..., w_N\}$, where N is the length of the document, and documents form a corpus $\boldsymbol{d}$ of size M: $\boldsymbol{d} = \{\boldsymbol{w}_1, ..., \boldsymbol{w}_M\}$. LDA models documents as the result of a generative process that produces them one word at a time, similarly to pLSA. First, the number of words is sampled from a Poisson distribution: $N \sim \text{Poisson}(\epsilon)$. Additionally, a topic distribution $\boldsymbol{\theta}_d \in \mathbb{R}^k$ is sampled from a shared k-dimensional Dirichlet distribution $\boldsymbol{\theta}_d \sim \text{Dir}(\boldsymbol{\alpha})$. For the corpus as a whole, these distributions form a topic matrix $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_M) \in \mathbb{R}^{M \times k}$. Then, for each word $n \in [1, ..., N]$, the process is assumed to first sample a topic $\boldsymbol{z}_n \sim \text{Multinomial}(\boldsymbol{\theta}_d)$ from a k-dimensional multinomial distribution. Subsequently, the word itself is sampled from a V-dimensional multinomial distribution $w_n \sim \text{Multinomial}(\boldsymbol{\beta}_{z_n,1:V})$, conditionally on both z_n and $\boldsymbol{\beta}$. Note that $\boldsymbol{\alpha} \in \mathbb{R}^k$ and $\boldsymbol{\beta} \in \mathbb{R}^{k \times V}$ are the priors, initially set by the user, that are optimized during training. On the other hand, $\epsilon \in \mathbb{R}$ is a static variable, which parametrizes the distribution of document lengths. The Poisson distribution chosen by the authors is not crucial to other steps, so a more accurate distribution could be used instead. LDA only contains one observable variable $\boldsymbol{w}$, while pLSA also includes the training document indices $\boldsymbol{d}$. Pseudocode describing the dataset generation process is given in Algorithm 1.

The process has three layers of variables. Firstly, $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are corpus-level variables, they are shared between all the documents in the corpus. Secondly, the $\boldsymbol{\theta}$ variables are document-level variables, as they are sampled once for each document. Lastly, z and w are word-level variables, which are sampled for each generated word. An important distinction to make is that under the pLSA and LDA assumptions, documents can contain multiple topics, unlike in the mixture of unigrams model. In fact, every word could be generated conditionally on a different topic. In this case, the document $\boldsymbol{w} \in \boldsymbol{d}$

Algorithm 1 Corpus generation assumption of LDA

Require: α, β, ϵ
  while $d \leq |\boldsymbol{d}|$ **do**
    $N_d \sim \text{Poisson}(\epsilon)$
    $\boldsymbol{\theta}_d \sim \text{Dir}(\boldsymbol{\alpha})$
    while $n \leq N_d$ **do**
      $z_{d,n} \sim \text{Multinomial}(\boldsymbol{\theta}_d)$
      $w_{d,n} \sim \text{Multinomial}(\boldsymbol{\beta}_{z_{d,n},1:V})$
    end while
  end while

is a mixture of $|\boldsymbol{w}|$ unique topics.

Given the parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, the joint probability of a topic mixture $\boldsymbol{\theta}$, a sequence of N words $\boldsymbol{w}$ and the words' respective N topic memberships $\boldsymbol{z}$ is:

$$p(\boldsymbol{\theta}, \boldsymbol{z}, \boldsymbol{w}|\boldsymbol{\alpha}, \boldsymbol{\beta}) = p(\boldsymbol{\theta}|\boldsymbol{\alpha}) \prod_{n=1}^{N} p(z_n|\boldsymbol{\theta}) p(w_n|z_n, \boldsymbol{\beta}) \tag{6}$$

The document's marginal probability can be obtained by integrating over $\boldsymbol{\theta}$ and summing over z:

$$p(\boldsymbol{w}|\boldsymbol{\alpha}, \boldsymbol{\beta}) = \int p(\boldsymbol{\theta}|\boldsymbol{\alpha}) \prod_{n=1}^{N} \sum_{z_n} p(z_n|\boldsymbol{\theta}) p(w_n|z_n, \boldsymbol{\beta}) d\boldsymbol{\theta} \tag{7}$$

Equation 7 provides the posterior probability of an observed document given some estimated hyperparameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. To obtain the marginal probability of an entire corpus $\boldsymbol{d}$, we can simply calculate the product of its individual documents' marginal probabilities. Continuing from Eq. 6, applying the chain rule to the left side gives:

$$p(\boldsymbol{\theta}, \boldsymbol{z}, \boldsymbol{w}|\boldsymbol{\alpha}, \boldsymbol{\beta}) = p(\boldsymbol{\theta}, \boldsymbol{z}|\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}) p(\boldsymbol{w}|\boldsymbol{\alpha}, \boldsymbol{\beta}) \tag{8}$$

Rearranging, the joint posterior distribution of the hidden variables $\boldsymbol{\theta}$ and $\boldsymbol{z}$ (given the words $\boldsymbol{w}$) is as follows:

$$p(\boldsymbol{\theta}, \boldsymbol{z}|\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{p(\boldsymbol{\theta}, \boldsymbol{z}, \boldsymbol{w}|\boldsymbol{\alpha}, \boldsymbol{\beta})}{p(\boldsymbol{w}|\boldsymbol{\alpha}, \boldsymbol{\beta})} \tag{9}$$

We arrive at the posterior distribution of an observed document's topic mixture $\boldsymbol{\theta}$ and its words' topic memberships $\boldsymbol{z}$, with Eq. 7 in the denominator. Calculating this posterior distribution provides the central task to inferring topics in LDA. However, Blei et al. prove that the posterior distribution in Eq. 7 is intractable [15]. Therefore, the LDA inference presented in Eq. 9 is reliant on statistical inference. The authors propose a convexity-based Variational Inference (VI) algorithm that makes use of Jensen's inequality, and further discuss the possibility of using Laplace approximations, higher-order variational techniques or Monte Carlo methods such as Gibbs sampling. In essence, the proposed VI training updates $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ in a way that maximizes the

log probability of the training data, $\log p(\boldsymbol{d}|\boldsymbol{\alpha}, \boldsymbol{\beta})$, until convergence. A benefit of LDA is that once the parameters have been estimated, topics can be generated for new documents in isolation. This is not possible with pLSA, as the training produces an empirical distribution over the topics that is directly tied to the training documents [15]. Moreover, the number of optimizable parameters does not grow with respect to the size of the training corpus. This is because LDA models the topic mixture weights $\boldsymbol{\alpha}$ as a fixed-length hidden variable instead of linking them to the training data. In total, LDA optimizes $kV + k$ parameters, the total number of parameters in $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. In addition to reducing the computational complexity of training, this also mitigates the possibility of overfitting, which is a notable problem with pLSA.

LDA has been used by researchers on a wide range on problem settings. Du et al. used LDA to study whether the Common Crawl dataset is a representative subset of internet data [38]. Li et al. proposed a topic-based spam detector that utilized LDA [94]. Zhang et al. used LDA to approximate themes within medical documents for use in personalized recommendations [196]. LDA has even been used to generate document embeddings, an approach coined Lda2Vec [116]. In summary, LDA has become a de facto approach for all applications within topic modeling.

However, LDA faces challenges in certain domains. As observed by Phan et al., LDA and pLSA both demonstrate reduced modeling performance with short documents [135, 134]. LDA generates a static set of possible topics during training, making it incapable of inferring topics that were not present in the training data. Furthermore, LDA requires that the number of topics is static and specified prior to training. This makes it unsuitable for applications where the number of topics is difficult to estimate. LDA also makes many simplifying assumptions about the underlying data generation process. Words within a document are assumed to be generated independently of one another, an assumption that is violated in essentially all natural language. Additionally, LDA suffers from the order effect, where changing the order of the documents can change the generated topics [1]. The topics generated by LDA are represented by collections of words and are not necessarily interpretable.

As a result of its shortcomings, there have been many proposed modifications to LDA. For the word independence assumption, the authors themselves point out that the Dirichlet distribution could be extended into a Dirichlet-multinomial distribution over trigrams [15]. Balikas et al. proposed SentenceLDA [8], where topics are sampled for sentences instead of individual words. Consequently, words within the same sentence are always assumed to have been generated by the same topic [1]. Contextualized Topic Model (CTM) is a variant of LDA which assumes that topics contained within the same document are correlated [14, 1]. Intuitively, documents could be assumed to rarely combine certain topics, while others are likely to appear together. In Hierarchical Dirichlet Process (HDP), the number of topics is not required a priori but is left as a learnable parameter [164]. In the similarly named hierarchical LDA (hLDA), a hierarchical structure for the topics is learned from the data [52].

2.2.5 Neural topic modeling

As noted by Abdelrazek et al. [1], the focus of topic modeling research has shifted to neural models in roughly the past five years, despite probabilistic models maintaining their popularity. One example of a neural network based approach to topic modeling is BERTopic [53]. The algorithm consists of four primary steps. First, the documents are embedded. Although there are many possible choices for embedding models, the authors propose using SBERT. To combat the high dimensionality of the SBERT embeddings, a dimensionality reduction is performed in the second step. The authors note that the concept of spatial locality becomes ill-defined in high-dimensional space, as the distance to the nearest data point approaches the distance to the furthest data point. This is commonly referred to as the curse of dimensionality [2], and although there are clustering algorithms that combat it [128], the authors opt for simply reducing the dimensionality of the embeddings. The authors propose using Uniform Manifold Approximation and Projection (UMAP) [112], an alternative to the popular t-distributed Stochastic Neighbor Embedding (t-SNE) [169] technique. Although it would also be possible to use the more traditional method of PCA, its assumption of linearity between data features is too strict for this use case. However, some of PCA's extensions such as kernel PCA could also be viable candidates, as they are also effective at separating nonlinear data [153]. In the third step, the embeddings are clustered using HDBSCAN [113], which is a hierarchical variant of DBSCAN [39]. Two of the primary benefits of using HDBSCAN are that the number of clusters is dynamic instead of static, and that documents can be marked as outliers. This reduces the requirements for data sanitization, as a minority of poor quality documents might not reduce the quality of the clustering. The last step generates topics for all the clusters. The authors propose a class-based term frequency-inverse document frequency (c-TF-IDF) procedure. It is a modification of the well established TF-IDF technique, which works on cluster-level instead of the document-level. In short, c-TF-IDF generates topics for each cluster, and each document inherits the topics of its respective cluster. Consequently, documents are often assigned topics with n-grams that are not present in the document itself.

BERTopic, while innovative and useful, has some drawbacks. Firstly, the model is only capable of inferring topics that were present in the training data. This can be an overly strict requirement if the goal is to model a large amount of topics. Secondly, the procedure contains multiple steps and is complicated compared to traditional approaches. This also makes the training slow and obfuscates the hyperparameter tuning process when using larger datasets. Thirdly, BERTopic is reliant on the capabilities of the embedding model. SBERT models will likely produce inaccurate embeddings for documents with multiple unrelated topics, highly polysemous words or sarcasm, which will inevitably mislead topic generation with c-TF-IDF. Lastly, as the approach relies on clustering, data quality and the balance between topic frequencies must be accounted for. Despite the use of HDBSCAN, low quality documents might still interfere with the clustering.

2.3 Language models

The foundational function of Language Models (LM) is to assign probabilities to sequences of text or their continuations [70]. This is to say, given a sequence of N words $\{w_1, ..., w_N\}$, a language model is able to either approximate its probability $p(w_1, ..., w_N)$ or the probability of the next word w_{N+1} conditionally on the previous ones: $p(w_{N+1}|w_1, ..., w_N)$. Both functionalities can also be applied to subsets of text in the form of a sliding window with size k. Then, the probability of a text window is $p(w_{N-k+1}, ..., w_N)$ and the probability of the next word conditionally on the window is $p(w_{N+1}|w_{N-k+1}, ..., w_N)$.

The joint probability of the words can also be defined as the product of the individual words' conditional probabilities:

$$p(\boldsymbol{w}) = \prod_{i=1}^{|\boldsymbol{w}|} p(w_i|w_1, ..., w_{i-1}) \tag{10}$$

Equation 10 relies on the fact that natural language, regardless of its otherwise unpredictable and evolving complexions, has a natural ordering [141]. In fact, despite its seemingly infinite intricacies, all natural language tends to at least have a start, an end and a progression from the former to the latter. Therefore, it can be assumed that a word in a sequence of natural language is generated conditionally only on its predecessors, not on its successors.

Perhaps the simplest language models are in the family of n-gram language models. In n-gram language models, a word's probability is inferred from the probability of n words appearing subsequently. This includes the inherent assumption that a word's probability can be approximated with only a few of its predecessors, regardless of how many words actually precede it [70]. In a 1-gram model, also known as a unigram model, the probability of each word is independent of all other words. In this simplistic model, the marginal probability of a document can be calculated with Eq. 11.

$$p(\boldsymbol{w}) = \prod_{i=1}^{|\boldsymbol{w}|} p(w_i) \tag{11}$$

Consequently, in a 2-gram (bigram) model, the probability of each word is conditional on the previous word. This leads to the generalized n-gram model, where the probability of a word is conditional on n of its predecessors, as shown in Eq. 12.

$$p(\boldsymbol{w}) = \prod_{i=1}^{|\boldsymbol{w}|} p(w_i|w_{i-n}, ..., w_{i-1}) \tag{12}$$

Language modeling has gone through multiple phases, during which different architectures have gained and lost popularity. Recurrent Neural Networks (RNNs) [3] are a class of neural networks designed for sequential data, which makes them a natural candidate for language modeling. When RNNs are applied to language, a hidden state is carried over and updated at each word. Connections form directed cycles

between different words, allowing them exhibit dynamic temporal behavior, unlike in simple feed-forward neural networks. This hidden state functions as a "memory", enabling the model to "remember" things about previous sections of the text, which is crucial for implementing understanding of past events. RNNs are hindered by the fact that the hidden state is updated only at the end of the forward pass. Consequently, a full forward pass is required for each word before the next word can be considered. Later improvements to the RNN architectures include Long Short-Term Memory (LSTM) [58] and Gated Recurrent Units (GRU) [25]. These modifications alleviate the vanishing gradient problem present in traditional RNNs and reduce the number of required parameters. Bi-directional LSTMs are a further advancement of RNNs, allowing information to flow both forwards and backwards [49, 50]. Bahdanau et al. [6] proposed an additive attention mechanism with a similar motivation to the scaled dot-product attention used in transformers [172]. However, unlike the latter, it is not parallelizable. Generally speaking, all previous language modeling architectures have been superseded by the currently dominant transformers [172]. Consequently, this thesis will focus solely on the transformer architecture.

2.4 Large language models

Large Language Models (LLM) are a class of language models that utilize a decoder-only transformer architecture [172], a large corpus of unannotated text and billions of trainable parameters. The resulting models are autoregressive and generate coherent continuations to existing text [21]. LLMs have shown astonishing performance in a wide range of natural language tasks such as conversational chatbots [123], mathematics [177, 5] and programming [151]. LLMs have also been applied to multimodal settings outside text-to-text generation, such as image-to-text generation [95], text-to-image generation [77], text-to-sound generation [44] and even any-to-any generation [186]. This sections discusses the broader field of LLMs, while focusing on the details that are most relevant to TLLM.

2.4.1 Tokenization

Large language models work with the same modeling assumptions as other language models. However, they usually do not split text into words but more specialized parts, commonly referred to as *tokens*. A token can be a word, but also a subword, a single character or even multiple words or symbols strung together. This is due to the restrictiveness of using entire words. The Oxford English dictionary [127], which contains only a fraction of all naturally occurring words, has over 300 000 main entries. Yet, LLMs are able to generate virtually any word, despite their vocabularies containing only a number of tokens in the range of 30 000-50 000 [34, 167, 21]. For example, the word *alike* is not in the vocabulary of Llama 2 [167]. Consequently, the model is unable to generate the word directly, regardless of preceding tokens. However, it could generate the tokens *a* and *like*, forming the word as the combination of two tokens that are present in its vocabulary. Alternatively, the model could generate the tokens *ali* and *ke*. If a word is exceptionally rare, a LLM might not even be able

to generate any of its bigrams as a single token. However, as long as the individual characters that make up the word are in its vocabulary, the model can generate it. Continuing the previous example, Llama 2 could produce the characters a, l, i, k and e individually. In fact, as its vocabulary contains the Latin alphabet, it is capable of generating any word that is made up of only Latin characters.

LLM vocabularies are created with algorithms that aim to make understanding language as easy as possible for the model while compressing repetitive information. After all, if a document consisting of N characters can be compressed into $0.5N$ tokens, the computational complexity of training on the document is roughly halved. A commonly used algorithm is Byte-Pair Encoding (BPE) [158], which was popularized by Sennrich et al. [156]. BPE has since been used by many LLMs, including Llama 2 [167], GPT-1 [140], GPT-2 [141] and GPT-3 [21]. BPE begins with a vocabulary $\boldsymbol{v}$ of individual characters and merges the most frequent pairs of tokens in a corpus until a predefined vocabulary size is reached. The motivation is to generate a vocabulary of common and possibly long strings of characters, while maintaining the ability to also generate rare words at a more granular level. Pseudocode for BPE is shown in Alg. 2. The way in which the vocabulary is generated is crucial to the capabilities of the model. In addition to BPE, there are a few alternative algorithms worth noting. These include WordPiece [154] and UnigramLM [81], the former of which is used by BERT. Some researchers have also questioned the dominance of BPE in language modeling; Bostrom et al. argue that UnigramLM leads to more morphologically accurate subwords and either matches or outperforms BPE [17].

Algorithm 2 Byte-Pair Encoding

Require: Corpus $\boldsymbol{d}$, Vocabulary size V_{max}
  $\boldsymbol{v} \leftarrow$ Unique characters in $\boldsymbol{d}$
  while $|\boldsymbol{v}| < V_{max}$ **do**
    $(t_1, t_2) \leftarrow$ Most frequent bigram in $\boldsymbol{d}$
    $t_{new} \leftarrow t_1 \bigcup t_2$
    $\boldsymbol{v} \leftarrow \boldsymbol{v} \bigcup t_{new}$
    $\boldsymbol{d} \leftarrow \boldsymbol{d}$ with all (t_1, t_2) replaced with t_{new}
  end while

2.4.2 Transformers

The transformer is a neural network architecture originally proposed by Vaswani et al. [172]. It utilizes an encoder-decoder structure and a parallelizable scaled dot-product attention mechanism. The transformer consists of two main components: the encoder and the decoder. The encoder maps a token $x \in \mathbb{N}_+$ into a vector of continuous values $\boldsymbol{z} \in \mathbb{R}^H$, where H is size of the model's hidden vectors. The full input sequence is usually longer than one token, in which case the sequence of i tokens $\boldsymbol{x} = \{x_1, ..., x_i\}$ is processed in parallel. Similarly to when processing one token, the encoder maps these tokens into a sequence of i vectors of continuous values $\{\boldsymbol{z}_1, ..., \boldsymbol{z}_i\}$. Starting from an empty sequence, the decoder generates an output sequence of o tokens one

at a time: $\boldsymbol{y} = \{y_1, ..., y_o\}$. When generating each token, it uses the sequence of vectors generated by the encoder within its cross-attention layers. For a concrete example, the input to the encoder could be a sentence written in English, while the desired output would be its German translation. Clearly, these two sentences can have different lengths: $i \neq o$. The English sentence would be tokenized, embedded and passed through the encoder. Afterwards, the decoder would generate the tokens of the German translation sequentially, while utilizing the output of the encoder. As the transformer architecture is central to TLLM, the remainder of this section provides a more detailed description of its design.

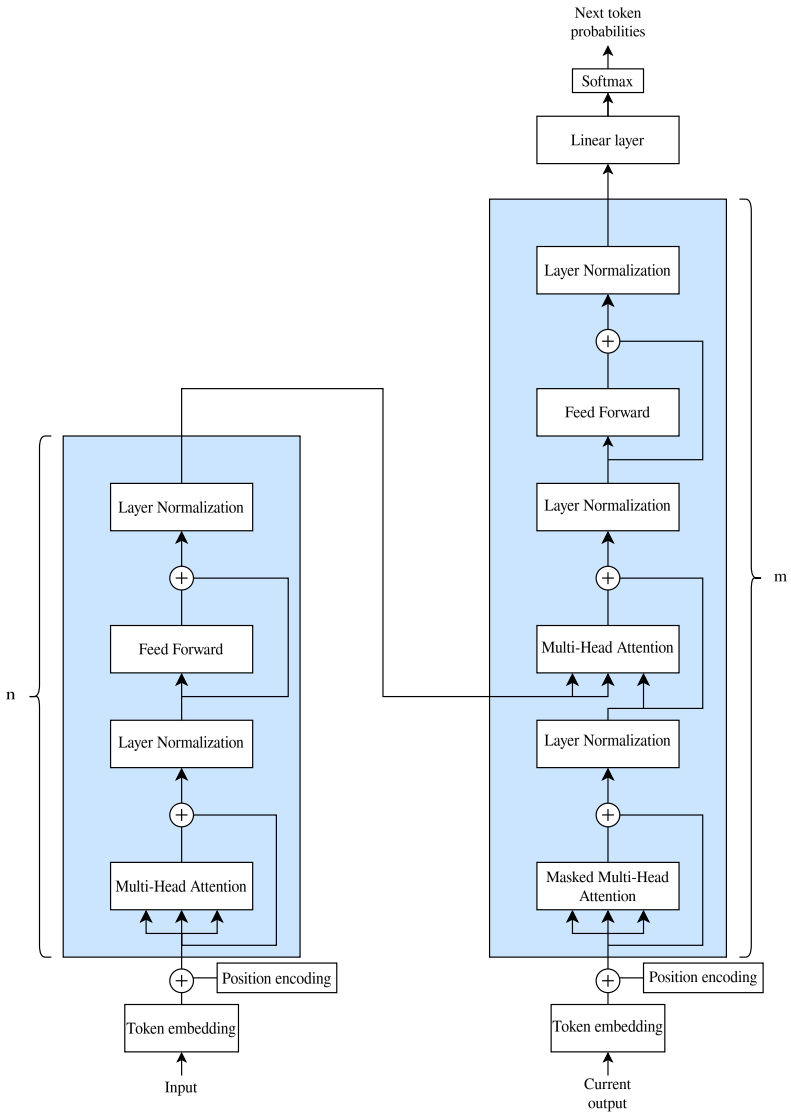


Figure 6: The transformer architecture

Embedding occurs before the first encoder and decoder blocks. The first step is tokenization, where the text is split greedily into tokens recognized by an input vocabulary V. Unrecognized tokens, if present, are mapped to a specific "unknown"

token. Then, a simple lookup from an embedding matrix $E \in \mathbb{R}^{|V| \times H}$ is used to map the tokens into their embeddings, where $|V|$ is the size of the vocabulary and H is the hidden size. This matrix is learned during training with backpropagation, identically to all other parameters of the model. Vaswani et al. used $H = 512$ [172], but subsequent research has often increased this to values such as $H = 768$ [34, 140, 141], $H = 1024$ [141] and $H = 4096$ [167, 69]. The input and output vocabularies are often identical, which allows for sharing the embedding matrix between the encoder and decoder. Obviously, this is not possible if the vocabularies differ, for example due to being monolingual vocabularies of different languages. The learned embeddings do not include any information about the tokens' relative placements within the input text. Instead, this information is communicated through a position encoding. In the original design by Vaswani et al., each dimension of the token embeddings is summed with a sine or a cosine function at a different frequency. Further improvements to this process are discussed in section 2.4.5. The authors also experimented with learned position encodings, but found that this did not increase the model's predictive capabilities.

Scaled dot-product attention, also known simply as attention, maps a query and a set of key-value pairs to an output [172]. In other words, it calculates a weighted average of feature representations with the weight proportional to a similarity score between pairs of representations [187]. A more intuitive interpretation is that self-attention calculates an association score between different token representations passing through the layer. In this way, the attention mechanism gives the model understanding of the connections between different tokens in its input. In a *masked self-attention layer*, the resulting matrix inside the softmax is also masked such that the attention scores between all tokens and their subsequent tokens is set to $-\infty$. This is done to prevent leftward flow of information. The attention mechanism is used inside multi-head attention layers, and its formal definition is shown in Eq. 13.

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}})\boldsymbol{V} \tag{13}$$

Multi-Head Attention (MHA) layers are perhaps the most complex part of the transformer. Each MHA layer consists of $3h + 1$ linear layers, h scaled dot-product attention layers and a concatenation step. Here, h is the number of attention heads, which Vaswani et al. set to $h = 8$. Subsequent research has often used a higher value, for instance $h = 16$ [34] or $h = 32$ [167]. For each scaled dot-product attention head at index i, there are three weight matrices that represent the weights of the linear layers with no bias:

$$\begin{aligned} \boldsymbol{W}_i^Q &\in \mathbb{R}^{H \times d_q} \\ \boldsymbol{W}_i^K &\in \mathbb{R}^{H \times d_k} \\ \boldsymbol{W}_i^V &\in \mathbb{R}^{H \times d_v} \end{aligned} \tag{14}$$

In the MHA layer, three matrices consisting of queries, keys and vectors are given as input and passed through different attention "heads" concurrently. The different heads learn different representation subspaces, which allow them to attend to different

types of information. The MHA output is computed as follows:

$$\begin{aligned} \text{MultiHeadAttention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) &= \text{Concat}(\text{head}_1, ..., \text{head}_h)\boldsymbol{W}^O \\ \text{where head}_i &= \text{Attention}(\boldsymbol{Q}\boldsymbol{W}_i^Q, \boldsymbol{K}\boldsymbol{W}_i^K, \boldsymbol{V}\boldsymbol{W}_i^V), \\ \boldsymbol{W}^O &\in \mathbb{R}^{(d_v \cdot h) \times H} \end{aligned} \tag{15}$$

The last matrix multiplication with $\boldsymbol{W}^O$ represents the final linear layer with no bias. This projects the output of the concatenation step, which has $d_v \cdot h$ columns, into the correct dimensionality of $\mathbb{R}^H$. Vaswani et al. [172] used $d_k = d_v = H/h = 64$, which leads to a significant dimensionality reduction. The authors note that the total computational cost of MHA using $d_k = 64$ is similar to single-head attention using $d_k = H$.

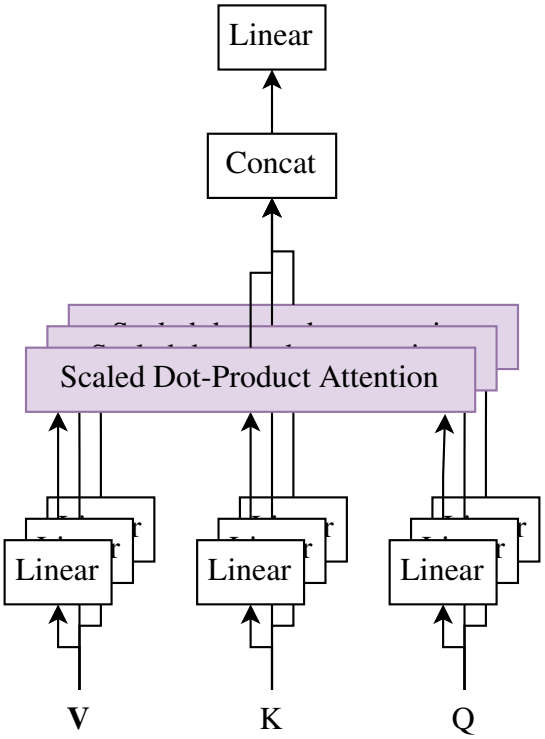


Figure 7: Multi-Head Attention (h=3)

The *encoder*, as the name suggest, work to encode information about the input in a way that is useful for the decoder. The encoder is made up of n encoder blocks that consist of a multi-head attention layer, a simple two-layer fully connected feed-forward neural network along with two summations and layer normalizations. The first encoder block of a transformer takes the sequence of token embeddings summed with their position encodings as input. Subsequent blocks use the result of the previous block as input. The last encoder block produces a sequence of vectors that contain the original tokens' continuous feature representations, which is treated as the output of the encoder.

The *decoder* is an autoregressive component that generates the model's output. It is made up of m decoder blocks that contain both a masked and an unmasked multi-head attention layer, along with a 2-layer feed forward network. The encoder and decoder can have different depths, and their sizes are an important hyperparameter for the performance of the model [188]. While the first attention layer is masked, the subsequent one is unmasked. The output of the encoder is used as the keys and values in the unmasked multi-head attention layers. The first decoder block takes the previously generated token as input, or a special *start* token if this is the beginning of the output. Consequent decoder blocks use the result of the previous decoder block as input. The last decoder block's output is passed through a linear layer that projects it to the dimensionality of the vocabulary, $\mathbb{R}^{|V|}$. A softmax operation then turns the resulting vector into a probability distribution over the vocabulary, after which a sampling method can be used to generate the next token. After generating a token, the next forward pass can be performed with the updated output. It is worth noting that during training, the previously generated token is not used as the decoder's input. Instead, the ground-truth token that is present in the training data is used, making the decoder's training non-autoregressive, unlike its inference. This is also the case when the model is used for tasks other than language generation, such as text classification.

Transformers have been applied to a broad range of fields within machine learning, including NLP [21], visual tasks such as optical flow estimation [71] and even weather forecasting [84]. Although transformers were originally applied to machine translation, they can be applied to virtually any sequence-to-sequence or classification task by modifying the input and output vocabularies. In essence, transformers have become exceptionally universal and the standard model architecture for a wide range of machine learning tasks [168].

The training procedure of transformer models is commonly split into two parts: pretraining and fine-tuning [16]. Pretraining produces a generic model which can then be fine-tuned for more specific downstream tasks. Similar approaches within the broader field of machine learning are known as transfer learning [110]. In essence, fine-tuning a pretrained model for downstream tasks is more data efficient and less computationally expensive than training the same model from a random initialization. The full transformer, also known as an encoder-decoder transformer, is still used by notable models such as BART [93] and T5 [143]. It also remains popular in pure sequence-to-sequence tasks such as translation [101]. However, many recent models rely solely on either the encoder or the decoder blocks, as shown in Fig. 8. These models are referred to as encoder-only and decoder-only transformers.

2.4.3 Encoder-only transformers

Encoder-only transformer models omit the decoder. Identically to the full transformer, the encoder blocks are placed sequentially and each block is fed the output of the previous one. The output of the last block is used as the output of the model without being passed to a decoder. The most notable encoder-only language models are in the BERT family [34]. Additionally, there have been some proposed variants of BERT. One notable example is RoBERTa [105], which modifies the models' training

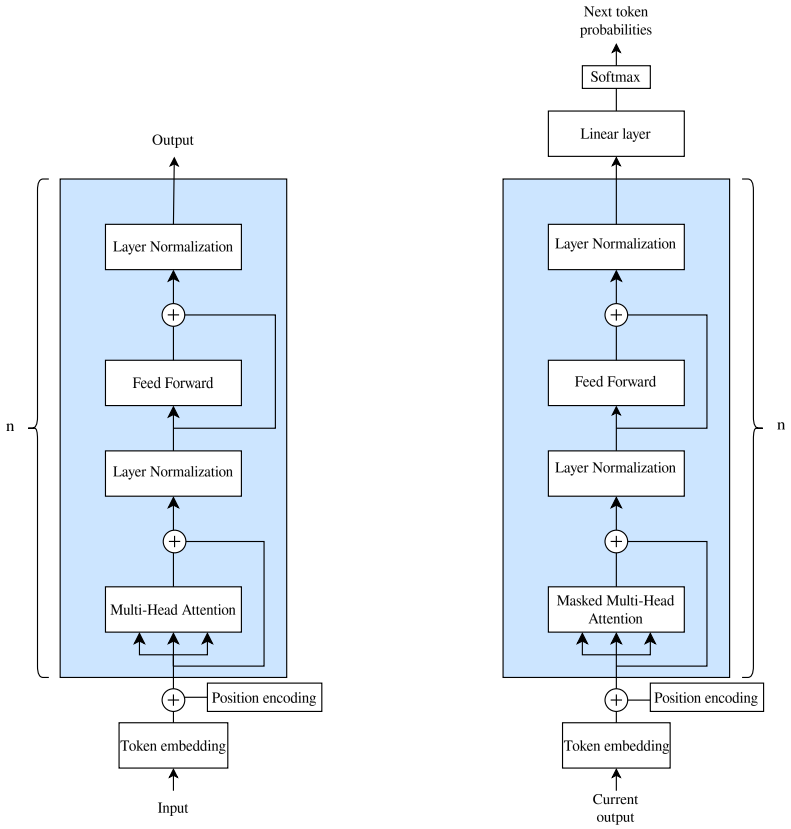


Figure 8: (Left) Encoder-only transformer, such as the one used in BERT (n=12) [34]. (Right) Decoder-only transformer, such as the one used in GPT-1 (n=12) [140] and GPT-2 (n=12, n=24, n=36, n=48) [141].

procedure. BERT and RoBERTa models have been pretrained in many different sizes, all of which can be fine-tuned for a range of more specific use cases. Encoder-only transformers have been found to be excellent at understanding language and context [34]. Consequently, they are often applied to Natural Language Understanding (NLU) tasks, such as text classification [101]. Unlike in decoder-only transformers, the self-attention layer is not masked. This means that when processing a sequence of tokens, the attention from each token is split among all other tokens, including its successors. A diagram of an encoder-only transformer is shown in Fig. 8 (left).

2.4.4 Decoder-only transformers

On the other hand, decoder-only transformers omit the encoder. This also entails removing the unmasked cross-attention, which in the original transformer receives its keys and values from the encoder. Consequently, decoder-only language models only use a masked multi-head self-attention layer, which makes the component inherently autoregressive. While encoder-only models excel at language understanding, decoder-only language models have been found to be excellent at language generation [21, 101]. A diagram of a decoder-only transformer is shown in Fig. 8 (right). As TLLM is

implemented using a decoder-only transformer, further sections will focus exclusively on this architecture.

Notable decoder-only transformer models include Llama [166], Llama 2 [167], Chinchilla [59], Mistral [69] and the GPT models created by OpenAI [140, 141, 21]. Radford et al. presented a framework for pretraining and task-specific fine-tuning of decoder-only transformer models [140]. In this framework, a decoder-only transformer model is first pretrained for next-token prediction on a large corpus, after which it can be fine-tuned to perform a downstream task. Models that adhere to this paradigm, possess a sufficiently large amount of parameters, and have been trained on a large corpus, are commonly referred to as LLMs. A pretrained LLM is often referred to as a base model [124] or, more commonly, a foundation model [16]. In essence, LLMs are trained to mimic their training data during inference. This training is done via optimizing a loss function on the model's ability to predict tokens, for which the cross-entropy loss is a common choice. The pretraining of LLMs is a form of unsupervised learning, as no human annotation is required. More specifically, this training is referred to as self-supervised learning. Self-supervised learning is a subset of unsupervised learning that, unlike other forms of unsupervised learning, still optimizes a loss towards some ground truth. However, this ground truth is not the result of human annotation like in supervised learning, but inferred automatically from the data. The foundation model generated by pretraining is then further updated with Supervised Fine-Tuning (SFT) [166, 167, 69] and possibly Reinforcement Learning from Human Feedback (RLHF) [21, 59, 125]. RLHF is especially crucial for training behaviors that, unlike the generic structure of language, are difficult to express as a loss function, such as safety [7]. Reinforcement Learning from AI Feedback provides an alternative to the costly process of human annotation, replacing it with off-the-shelf LLMs while maintaining similar performance [88]. Recently, Direct Preference Optimization (DPO) [142] has gained popularity as it replaces RLHF with a considerably simpler optimization objective.

2.4.5 Modifications

The original encoder-decoder transformer introduced many of the techniques still used today. However, there have also been numerous proposed improvements to its individual components. As discussed previously, the position encoding gives the model information about the tokens' positions within the input. Subsequent improvements to this procedure have surpassed the predictive performance of the original, simplistic implementation on many benchmarks. One notable example is the Rotary Position Encoding (RoPE) [162]. RoPE encodes the positional information with a rotation matrix, leading to improved language generation abilities, especially with longer token sequences. RoPE is used by Llama 2 [167], PaLM [26] and Mistral [69], among others. Another prevalent position embedding technique that deviates from the original is called ALiBi [137]. It has been employed by influential models, including the 176B multilingual BLOOM [185] and its fine-tuned versions, such as the BLOOMZ models [119].

Two notable challenges with transformer models are their limited context size and the self-attention layers' time- and space-complexity increasing quadratically with

respect to the length of the input [72]. These are not addressed by small modifications to components such as the position encoding. However, due to the popularity of transformer models, there have been fundamentally different architectures that attempt to mitigate these drawbacks. Three architectures stand out: Receptance Weighted Key Value (RWKV) [131], retentive networks [163] and Mamba [54]. These, and other similar architectures, might prove superior to the transformer in the future, but will not be discussed further in this thesis.

2.4.6 Prompting

The quality of the text produced by LLMs depends heavily on the composition of the preceding tokens, colloquially known as the *prompt*. This notion applies to both foundation models [180] and LLMs fine-tuned for instruction following [78]. The prompt is placed in the sequence of tokens that is used for next token prediction, known as the model's context window [21], or simply its *context*. The context also includes the potential system prompt, along with previous user prompts and LLM responses. For foundation models, the model is only trained to continue text in a way that is reminiscent of its training data. This is to say, foundation models have no inherent abilities regarding user interaction or instruction following. However, these models can also be used for Question Answering (QA) tasks by formatting the prompt in a way that elicits such behavior. For instance, Radford et al. condition GPT-2 for translation by placing pairs of English and French text in its context [141]. The context ends with only an English text without its French counterpart, after which the model is used to greedily sample the continuation. The authors use the first whole sentence that is generated as the translation. A similar approach is often taken for QA, where the model is conditioned with questions and answers, after which the last question is left unanswered. Wang et al. use this prompting technique for three different objectives: creating training tasks for LLMs, labeling them based on whether they involve classification, and producing their completions [178]. This way of prompting directs the LLM towards the distribution of training data that is similar to the desired task. In other words, it conditions the LLM to produce text that is similar to parts of its training data that relate to the desired behaviour. Intuitively, the training corpus most likely includes questions and their respective answers, translations, and other pairs of connected information.

There are an infinite number of different ways to form the prompt for any practical task, but some prompts produce better outputs than others. This has spawned the study of prompt engineering [104], producing the best performing prompts for given tasks. Many different prompting techniques have been found to improve LLM generation results either generally or within a narrow use-case. Many such prompting methods were studied by Bsharat et al., along with evaluations of their effectiveness on contemporary models [22]. It is worth making the distinction that prompting techniques do not modify the model in any way, as they do not involve gradient updates. Instead, the prompt simply leads the same model towards more desirable generation paths.

When prompting LLMs that have been fine-tuned for instruction following,

conditioning the model with examples of correct generations is not strictly required. However, it has been found to often improve the quality of LLM generations. In fact, this technique, referred to as In-Context Learning (ICL) [21], can allow LLMs to solve previously difficult or impossible tasks [180]. When demonstrations of the task are provided to the model during inference, this is also referred to as one-shot or few-shot learning, depending on the number of demonstrations [21]. This is in contrast to zero-shot learning, where no such examples are given. This ability of LLMs to learn to perform tasks based on tokens in their context is widely recognized as an emergent ability [179], as it is an unexpected behavior that was not targeted in the model's training. Furthermore, the prevalence of ICL capabilities has been observed to increase along with model size [21, 179]. Researchers have adopted different definitions for ICL. The definition used by Dong et al. states that: "In-context learning is a paradigm that allows language models to learn tasks given only a few examples in the form of demonstration" [37]. Li et al. use ICL synonymously with prompting, defining both as including task demonstrations [97]. Meanwhile, Brown et al. define ICL more loosely as conditioning the LLM on instructions and/or some demonstrations [21]. Despite its successes on some tasks, some authors have raised doubts about the extent of ICL's capabilities. Kossen et al. [79] conclude that although ICL can elicit better performance, it is unable to fully overcome information learned during pretraining. The authors include information in the prompt that is contradictory to the information that was trained on during pretraining, and find that the model does not fully transition to the newly given information.

Another popular prompting style is Chain-Of-Thought (CoT) prompting [180] that modifies the prompt to encourage responses that walk through the answer one step at a time. This approach was further explored by Kojima et al. [78], who added the prefix "Let's think step by step" to different prompts. The authors refer to this prompting style as Zero-shot-CoT, and evaluated it on many different tasks such as arithmetics and symbolic reasoning. The authors observed a performance increase from 10.4% to 40.7% on GSM8K [27] and 17.7% to 78.7% on MultiArith [149] while using an InstructGPT [126] model. CoT prompting has since been used by many LLMs directly in their system prompt. For instance, Mukherjee et al. utilize the system prompt of the Orca LLMs to align the model with the user's desired answer length and specificity [120]. One of the authors' recommended system prompts is as follows: "You are an AI assistant that helps people find information. User will you give you a question. Your task is to answer as faithfully as you can. While answering think step-by-step and justify your answer". Following this advice, Lian et al. use the following system prompt for MistralOrca [98]: "You are MistralOrca, a large language model trained by Alignment Lab AI. Write out your reasoning step-by-step to be sure you get the right answers!".

Prompt optimization can also be automated [159], often through a LLM producing the prompt candidates. Zhou et al. propose the Automatic Prompt Engineer (APE) framework, where the algorithm iteratively removes prompts that lead to poor generations on specific QA pairs from a set of initial candidates $\mathcal{U}$ [199]. In case the initial set of generated prompts does not contain a sufficiently performant prompt, the framework also allows resampling $\mathcal{U}$. Furthermore, Yang et al. propose

Optimization by PROmpting (OPRO) [190]. OPRO modifies APE to generate new prompts during every iteration. These new prompts aim to improve test accuracy based on a trajectory estimated from previously generated prompts.

2.4.7 Fine-tuning

All training that is not part of a model's pretraining is classified as fine-tuning. Fine-tuning can be done in a multitude of ways, the most traditional of which is full fine-tuning [33, 48], alternatively known as full parameter fine-tuning [108]. Originally called just model tuning, it was proposed by Howard and Ruder [64]. Full fine-tuning updates all of the network's weights using the same classical backpropagation used in pretraining, but with a task-specific dataset and possibly a new loss function. This approach has several drawbacks. Firstly, the process is often prohibitively expensive to perform: 16-bit full fine-tuning of LLaMA 65B requires more than 780GB of GPU memory [33], while full fine-tuning of GPT-3 requires 1.2TB of GPU memory [66]. This is usually performed on clusters of hundreds of GPUs and therefore out of reach for most individuals and organizations, similarly to pretraining. There are also considerations of inefficiency, as even when full fine-tuning is feasible, it remains an exceptionally large commitment of resources such as electricity and hardware. Secondly, the approach suffers from a phenomenon coined catastrophic interference, also known as catastrophic forgetting [111, 144]. Catastrophic forgetting is a phenomenon where deep networks forget previously learned tasks when they are retrained [63]. In the context of LLMs this means that updating the model's weights for a specific task, such as QA, makes it forget things it learned during pretraining [106, 80]. This might include information about people, places, words or even the structure of language. Thirdly, full fine-tuning can lead to overfitting, causing suboptimal generalization [24]. Even in the absence of catastrophic forgetting, task-specific training data will turn a general model into only a narrow specialist unless carefully constructed. All these challenges make full fine-tuning relatively difficult and inefficient. As a mitigation strategy to the problems faced by full fine-tuning, Lv et al. proposed LOw-Memory Optimization (LOMO), which reduces the memory requirements of full fine-tuning a 65B parameter model by 10.8% [108]. LOMO modifies the process by performing the gradient computation and parameter update in one, unified step. Lv et al. built on this work in a further algorithm, called low-memory optimization with adaptive learning rate (AdaLomo), which frees the learning rate to adapt during training [107].

Recently, fundamentally different approaches that do not perform full fine-tuning have received increasing attention. These include a range of different algorithms, which are all categorized under the unifying term of Parameter Efficient Fine-Tuning (PEFT) [36]. The first notable PEFT method was proposed in 2019 by Houlsby et al., called adapter modules [63]. In adapter modules, the base model is kept frozen while updating the weights of added adapter layers. It is worth noting that this idea of frozen weights has also been used in previous research unrelated to LLMs, for instance in ELMo [133]. Furthermore, adapter modules have been applied to non-transformer models prior to their implementation in transformers [145]. Houslby et al. demonstrate

adapter modules nearly matching the performance of full fine-tuning on some tasks by adding a number of parameters equal to 3.6% of the original model [63]. However, adding layers deepens the model, making inference slower. Changing the structure of the model also leads to further complexity during inference, as the models' weights are no longer interchangeable. Moreover, it fails to reliably match the performance of full fine-tuning.

To answer these challenges, Low-Rank Adaptation (LoRA) training was proposed in 2021 by Hu et al. [66]. Similarly to the approach taken with adapter modules, the algorithm keeps the base model frozen. Unlike adapter modules, LoRA does not add new parameters to the model. For an existing weight matrix $\boldsymbol{W}_0 \in \mathbb{R}^{a \times b}$, LoRA updates $\boldsymbol{A} \in \mathbb{R}^{a \times r}$ and $\boldsymbol{B} \in \mathbb{R}^{r \times b}$, whose product forms a r-rank approximation of an update matrix: $\boldsymbol{BA} = \Delta \boldsymbol{W}_0$. This update matrix is then summed with the original weights prior to evaluation and inference. Typically, every transformer block's query-, key- and value matrices are updated, but the technique can also be applied to the parameters of a linear layer. For example, a linear layer $\boldsymbol{h} = \boldsymbol{W}_0 \boldsymbol{x}$ with the original weights $\boldsymbol{W}_0$ is updated to perform the following:

$$\boldsymbol{h} = \boldsymbol{W}_0 \boldsymbol{x} + \Delta \boldsymbol{W}_0 \boldsymbol{x} = (\boldsymbol{W}_0 + \Delta \boldsymbol{W}_0) x = \boldsymbol{W} \boldsymbol{x} \tag{16}$$

The approach taken by LoRA is reminiscent of the technique proposed by Zhang et al., which trains and ultimately sums the weights of a secondary model with the weights of a pretrained model [194]. It is worth noting that the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are small when compared to the full weight matrix $\boldsymbol{W}_0$, as they form a low-rank approximation of it. This makes the training much less resource intensive, as it effectively updates only a fraction of the number of parameters in the full model. Unlike in full fine-tuning, the gradient updates only affect $\Delta \boldsymbol{W}_0$ and not $\boldsymbol{W}_0$, which alleviates the effects of catastrophic forgetting. The authors note that for GPT-3, the number of trainable parameters can set as small as 0.01% of the full model size.

Hu et al. find that LoRA training of GPT-3 outperforms all previous methods on WikiSQL, MNLI-m and SAMSum validation datasets by training only a number of parameters equal to 0.2% of the full model [66]. This evaluation also included full fine-tuning. Even when training with only 0.03% of the full model's parameters, LoRA performs better than full fine-tuning on MNLI-m and SAMSum, getting within the error margin from full fine-tuning on WikiSQL. Additionally, the method makes changing between fine-tuned models simple. One LoRA fine-tuned model can be switched to another by subtracting the current LoRA update weights $\Delta \boldsymbol{W}_0$ from $\boldsymbol{W}_0$ and adding the LoRA update weights of another.

A further advancement of LoRA was proposed in 2023, called QLoRA [33]. It works largely the same way, but quantizes the model parameters to 4 bits during training using a novel 4-bit NormalFloat datatype. QLoRA also uses double quantization, a method where the quantization constants are also quantized and dequantized when needed for further reductions to its memory footprint. As noted by the authors, QLoRA training of a 16-bit model with 65B parameters requires less than 48Gb of GPU memory without slowing down inference or degrading predictive performance when compared to full fine-tuning.

Another prominent PEFT method is called *prefix tuning* [97]. In addition to its applicability on decoder-only models, it can also be used to fine-tune encoder-decoder models. In prefix tuning, the model weights are kept frozen identically to LoRA. However, no additional weights are added, removed or updated. Instead, a matrix of embeddings $\boldsymbol{P} \in \mathbb{R}^{k \times H}$ is learned for each of its layers, where k is the number of trainable embeddings and H is the model's embedding size. Then, for each transformer block, prefix tuning prepends these embeddings to its activations. When applied to encoder-decoder models, separate prefixes are learned for both the encoder and decoder. The authors reparametrize these embeddings such that $\boldsymbol{P}_{i,1:H} = \text{MLP}(\boldsymbol{P}'_{i,1:j})$. Here, MLP is a large neural network, and $\boldsymbol{P}'$ is a matrix with an identical number of rows but less columns compared to $\boldsymbol{P}$: $j < H$. This is done mainly to stabilize the learning process. Unlike the representations of token embeddings, these embeddings remain unchanged throughout the forward pass for all prompts. An important distinction is that while prompt engineering techniques add tokens represented by discrete values, prefix tuning optimizes continuous word embeddings through backpropagation. As these generated embeddings are not constrained by the embeddings of existing natural language tokens, this approach is more expressive. *Soft prompt tuning*, also known simply as prompt tuning, is similar to prefix tuning, but only prepends tokens to the beginning of the input [92]. The prior name will be used in this thesis to better distinguish it from prompt engineering. Given a sequence of n input tokens, the LLM first embeds them normally to create the matrix $\boldsymbol{X} \in \mathbb{R}^{n \times H}$. Then, the soft prompt $\boldsymbol{P} \in \mathbb{R}^{k \times H}$ of k of tunable embeddings is prepended to $\boldsymbol{X}$, creating $\{\boldsymbol{P}, \boldsymbol{X}\} \in \mathbb{R}^{(n+k) \times H}$, which then passes through the encoder as a whole. During later layers, the representations of these embeddings are calculated identically to those of token embeddings. The authors apply soft prompt tuning to the encoder-decoder model T5, but it can be applied similarly to decoder-only LLMs. According to the authors, soft prompt tuning benefits from the model being able to update the embeddings' intermediate-layer representations. Clearly, soft prompt tuning also trains less parameters, as it only prepends learned embeddings to the beginning of the encoder's input and does not require the reparametrization technique used in prefix tuning. As both prefix tuning and soft prompt tuning train embeddings that occupy parts of the input, is should be highlighted that they reduce the maximum length of the user prompt. This can make them unviable in some use cases.

Fine-tuning, while having been empirically found to be helpful, lacks theoretical guarantees and is not universally applicable. Zhou et al. proposed the *superficial alignment hypothesis*, which states that fine-tuning mainly aligns the model in terms of answering style, not in terms of capabilities or knowledge [198]. The authors fine-tune a model with only a thousand question-answer pairs, and find that it generalizes well to a range of tasks that were not present during fine-tuning. According to the authors, this possibility of achieving performance comparable to models that have been fine-tuned on significantly larger datasets supports the superficial alignment hypothesis. The trade-off between training data quantity and model size has also been previously studied by Kirstain et al. [76]. The authors find that some tasks, such as multiple choice tasks with restricted output spaces, benefit much more from added training data. In open-ended QA tasks, the authors observe notably smaller improvements

from enlarging the training dataset. Khotha et al. [80] find that fine-tuning harms pretrained capabilities on prompts that are similar but not the same as the type of data used for fine-tuning. The authors find that modifying prompts to be more dissimilar to the fine-tuning data distribution, while retaining their semantic meanings, alleviates this effect. One proposed way to accomplish this is to translate them into different languages. Berglund et al. propose the *reversal curse* [12]. The authors find that a LLM fine-tuned on the logical relationship between A and B does not necessarily learn the relationship from B to A. More concretely, fine-tuning a model with the information that Olof Scholz was the ninth Chancellor of Germany did not teach the model to answer the question: "Who was the ninth Chancellor of Germany?". This highlights one of the challenges faced by contemporary pretraining and fine-tuning methods.

2.4.8 Explainability

A noteworthy downside of LLMs is that their results are usually not explainable; their reasoning can not be presented in human-understood terms [197]. Consequently, there is no mechanism for knowing if the output generated by a LLM is correct. One approach to approximate LLM trustworthiness is to use its own confidence as a proxy. This can be achieved by taking the log probabilities of the generated tokens and averaging them [165]. Achiam et al. note that the GPT-4 foundation model is highly calibrated, meaning its likelihood scores for generated tokens correlate strongly with the probability of the model's output being correct [125]. However, the authors note that the level of calibration suffers during the RLHF process. Chen et al. proposed an alternative approach, called ASPIRE [24]. ASPIRE is a framework that adapts LLMs for a specific task and self-evaluation simultaneously. The training consists of three steps. First, the foundation model's original weights $\boldsymbol{\theta}$ are frozen and the model is fine-tuned for the desired task with some PEFT approach. The authors employ soft prompt tuning, optimizing the parameters $\boldsymbol{\theta}_p$. In the second step, the model is prompted to generate many different answers to a set of questions relating to the desired task. The sampling utilizes beam search to generate high-likelihood outputs. These answers are then labeled as either correct or incorrect using the metric $M(y, \dot{y})$, where y is the generated output and $\dot{y}$ is the reference answer. The authors utilize ROUGE-L [100], interpreting a sufficiently high value as an indication of correctness. Thirdly, with the base model and the new soft prompt tuning parameters frozen, a second round of fine-tuning is performed for new parameters $\boldsymbol{\theta}_s$. The objective of this round of fine-tuning is to add a confidence score to the LLM's outputs. Ideally, $\boldsymbol{\theta}_p$ together with $\boldsymbol{\theta}$ should form the QA behavior of the model, while $\boldsymbol{\theta}_s$ should not interfere with this process. Instead, $\boldsymbol{\theta}_s$ should only train the model's ability to evaluate its own answers. The authors also perform this step with soft prompt tuning. The resulting model is able to generate answers to questions while also approximating its own confidence.

2.4.9 Inference

As discussed in section 2.4.4, LLM inference generates next token distributions autoregressively. A token is sampled and the process repeats, until the LLM generates a specific token that indicates the end of generation. Due to the combination of this simplistic inference method and a training approach that makes the model to mimic its training data, some authors have labeled LLMs "stochastic parrots" [11]. This name conveys the opinion that even though LLMs have demonstrated exceptional capabilities, they do not understand the text they are producing. Instead, it is argued that LLMs are simply parroting their training corpus in a stochastic manner. This sentiment is not universal, as many authors have pointed out the emergence of novel world models inside LLMs that suggest some higher level understanding [55, 96]. Additionally, some inference techniques add further steps to trade speed for improved long-term thinking. Firstly, the classical method of *beam search* is also applicable to LLMs [41]. In beam search, multiple "beams" of n tokens are generated. Then, the first token from the sequence of tokens that had the highest total log likelihood is chosen. The motivation is to allow the model to investigate if a token, although promising at the current time step, would ultimately lead down an undesirable path. Although beam search has been effective in translation models, it has been observed to lead to repetitive text with low entropy and a lack of coherence during story generation [181]. This is a natural consequence of maximizing long-term log probability during open-ended tasks. Beam search is still applicable in situations where low-entropy generations are not an issue, such as in the previously mentioned ASPIRE framework. Secondly, *Tree-Of-Thoughts* (TOT) inference [191] produces a tree of different "thoughts"; partial solutions to the task at hand. The authors use the example tasks of creative writing, game of 24 and crosswords for evaluation. This tree is generated, evaluated and navigated either in a Breadth First Search (BFS) or Depth First Search (DFS) manner, leading to an inference technique that is deliberately incremental and multi-layered. Although the approach is complex and requires more computation when compared to basic inference or even CoT, the authors demonstrate significantly improved generation quality in multiple tasks. Thirdly, Goyal et al. propose adding a specific *pause token* to LLM vocabularies [47]. The LLM is free to generate these tokens as part of its output, but they are ultimately removed from the result as they are not human-interpretable. The benefit is that the pause tokens provide more paths of computation for subsequent, real token predictions. The authors found that this improved the answering capabilities of LLMs on QA tasks, although their experiments were limited to models with at most one billion parameters. Intuitively, this approach can be thought of in the same way as impulse control in humans: the LLM is able to generate pause tokens when using more computation is deemed necessary. This approach suffers from the fact that it requires the foundation model to be pretrained with specifically crafted data that includes pause tokens.

An interesting approach to LLM inference can be found in formal grammars [43]. Formal grammars manipulate the vocabulary that is available to the LLM at each token generation step. Consequently, the LLM is only capable of producing answers that follow the desired format. Inference that utilizes a formal grammar is referred

to as Grammar-Constrained Decoding (GCD) [42]. GCD can be especially useful if the desired output has a specific structure, such as JSON or chess moves. In such situations, generations that do not follow the predefined grammar can be rejected without further evaluation. Some authors have also proposed modifying the formal grammar based on the input [42] or generating it with a LLM [175].

2.4.10 Accelerating inference

LLM inference is usually performed on either a designated Graphics Processing Unit (GPU) or a purpose-built Tensor Processing Unit (TPU). These components are specifically designed to be fast at performing instructions such as matrix multiplication. This provides a notable boost in efficiency when performing tasks that make heavy use of these instructions, such as LLM inference. There have been many advances in making inference less resource intensive. FlashAttention [30] and the subsequent FlashAttention 2 [29] accelerate inference by better utilizing the faster Static Random-Access Memory (SRAM) on modern GPUs instead of the slower High Bandwidth Memory (HBM). In essence, this modification significantly improves parallelism during matrix multiplications, and can be up to 10 times faster than standard attention implementations. Sliding Window Attention (SWA) is an alternative attention mechanism originally proposed by Beltagy et al. as part of the Longformer architecture [10]. SWA uses a fixed-size window attention, where given a window size w, each token attends to $0.5w$ tokens on either side. In SWA, computational complexity increases linearly $O(n)$ instead of quadratically $O(n^2)$ in document length. SWA is employed by Mistral [69], among others.

Many authors have proposed the classical approach of *pruning* [87] to be used on LLMs. Using pruning, Van et al. were able to reduce the parameter count of Llama-2-7b by 20%-30% with negligible loss in performance [170]. *Quantization* is another way in which inference can be made more efficient. The previously mentioned 4-bit NormalFloat datatype can be used for inference as well as training [33], and this is supported by many libraries such as HuggingFace transformers [184]. Additionally, Dettmers et el. proposed a routine for quantizing transformer parameters to 8-bits for matrix multiplications [32].

Another approach to accelerating inference is known as *knowledge distillation* [57]. The idea behind knowledge distillation is to transfer knowledge from a larger model to a smaller one. Ideally, this would produce a smaller and faster model that maintains a reasonable amount of the larger and less efficient model's capabilities. Related to knowledge distillation, Zhang et al. [192] study *the curse of capacity gap*, which states that a larger teacher language model does not always lead to a better student language model [193]. The authors theorize that there exists an optimal capacity gap between each teacher and student model. Interestingly, the authors find that this gap remains consistent throughout different student sizes and architectures.

2.4.11 LLMs in topic modeling

Despite their lack of popularity within the field of topic modeling, there are still some notable examples of research that combines LLMs with topic modeling. Wang et al. use a novel prompting technique that relies heavily on ICL to generate topics with pretrained LLMs [176]. Similarly to traditional topic models, the authors use sequences of n-grams to represent topics. Prakash et al. used LLMs as part of a multimodal architecture to generate topics for images [136]. Xu et al. use an encoder-decoder transformer, prefix tuning and diffusion to train DeTiME [189], a model that generates highly clusterable topic embeddings. Peña et al. use GPT-2 to analyze topics in public affairs documents [130]. Instead of filling the role of a generative model, the authors use the LLM to generate feature representations together with encoder-only models. In essence, this approach transformed topic generation into a multi-label classification task, relying on a subset of the documents already having a manually crafted topic. Rijken et al. used a LLM to interpret non-LLM topics by compressing them into short descriptions [148]. In their model evaluation, a domain expert was tasked with manually creating short summaries for medical documents. These short summaries were also generated by analyzing LDA topics with ChatGPT [123], after which the domain expert was asked to estimate how useful the automatically generated summaries were in comparison to theirs. The domain expert's opinion was that 50% of the automatically generated summaries were useful. However, only 5% of them were better and 30% equally good when compared to the manually created summaries.

In conclusion, while there have been some studies on combining LLMs and topic modeling, it is still a largely novel application, and all previous studies have opted to use pretrained models, such as ChatGPT. As far as the authors are aware, there have been no studies on fine-tuning LLMs for topic modeling.

3 Research material and methods

This section examines the implementation details of TLLM. First, we outline the process of creating SFT datasets for topic modeling. This process enables the creation of balanced and generic datasets, whose topic granularity can be altered easily. Then, we motivate the decisions behind the chosen foundation model and fine-tuning method. Finally, the section ends with a discussion related to inference and prompting.

3.1 Dataset

SFT requires a dataset of question-answer pairs that the model can be trained on. In the case of TLLM, these pairs consist of documents and their desired LLM replies, which include the correct topic. For this study, the documents are collected from the Reddit TL;DR [174] and OpenWebText datasets [46], a dataset comprised of Wikipedia entries [182] and datasets containing articles by CNN and the DailyMail [155, 56]. The Wikipedia articles were collected from the *20220301.en* dataset provided by HuggingFace [67] and truncated to only include the first chapter, as later sections often diverged from the original topic of the article. All documents were further truncated to 512 characters. These datasets were chosen to represent a wide range of different internet material. The Reddit TL;DR and OpenWebText datasets contain casual internet conversations, often written in first person. Meanwhile, the Wikipedia dataset contains factual and neutral information written in third person. Lastly, the other two datasets contain news, which are also a sizeable part of the internet and whose token distributions are typically different from the other training documents. Generally, any predictive model should be trained on data that is similar to the data it will be given during inference, as this relaxes the requirements for out-of-distribution generalization. Therefore, the training datasets were chosen with the goal of making the training data distribution maximally similar to the data distribution seen during inference. A notable detail is that all of the training data is written in English. Despite the lack of multilingual training data and the small parameter count of the model, TLLM exhibits understanding on a range of different languages. This suggests that transfer learning from the model's multilingual pretraining alone is capable of enabling multilingual topic modeling.

The documents are annotated manually, but with assistance from automatically generated labelings. The automatic labels were generated with a larger LLM: Llama-2-70b-chat. This is comparable to weakly labeled data and reminiscent of knowledge distillation from larger LLMs. The automatic labeling process used a formal grammar to guarantee the correctness of the output format, as provided by llama.cpp [43]. The automatic topics were generated with the following process, which is also visualized in Fig. 9. First, approximately 145 000 documents were given topics with Llama-2-70b-chat. This process labeled documents from the different datasets in a round-robin process, so each of the five datasets was equally represented. These documents received 15317 unique topics, which were generally deemed too specific for the desired answering style of TLLM. Next, all topics were embedded using a sentence embedding model. Then, the topic embeddings were clustered using HDBSCAN, which created

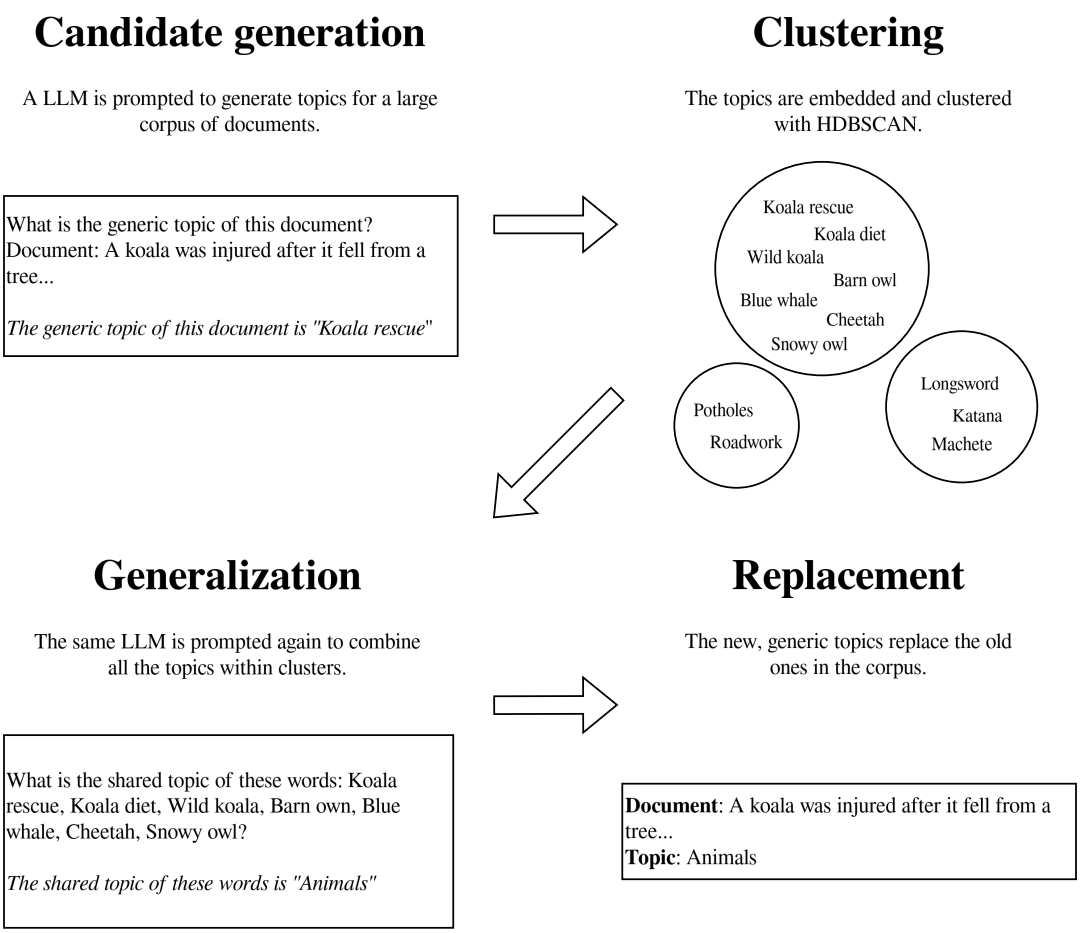


Figure 9: A visualization of the dataset creation process.

3849 unique topic clusters. It is also possible to reduce the embedding dimensionality prior to HDBSCAN with UMAP, but this was found to not be crucial for the resulting clusters. It is worth noting that the parameters of HDBSCAN can be altered to modify the amount and size of the clusters, which will directly affect the granularity of the final topics. The same also applies to UMAP, in case it is utilized. After clustering, the topics within each cluster were used to create a general, unifying topic with the previously used LLM. The prompt for this made heavy use of ICL and can also be found in Appendix A. The old, specific topics were then replaced with these more general topics. Originally, it was considered that the pairs of documents and their new topics could be used as training data directly. However, many of the documents were of low quality and needed to be filtered out, on top of which the clusters were not homogeneous in terms of true latent topics. For instance, the topic *minesweeper strategies* was clustered with topics related to explosives instead of video games and the topic *formula one* was clustered with topics related to mathematics instead of motorsports. Additionally, the differences between topics such as *riot*, *protest*, *strike*, *crime* and *politics* were too narrow to be consistently recognized. Previous research has shown that data consistency and quality are more important than quantity in LLM

fine-tuning [198] and pretraining [109]. Therefore, the decision was made to use the automatically generated topics only as assistance during manual labeling.

We also conducted a subsequent experiment, where clustering was performed using document embeddings instead of topic embeddings. However, the resulting clusters were either accurate and small (leading to a majority of the documents being labeled as outliers) or big and noisy (which led to inaccurate general topics). This was attributed to the embedding model attempting to fully represent the large amount of information in the documents, leading to inaccuracies. Furthermore, due to the subjectivity of the topics, seemingly related documents sometimes received mutually dissimilar topics when evaluated in isolation. These mistakes sabotaged the creation of general topics in the following step. This process also required reducing the number of processed documents for computational reasons. HDBSCAN requires the construction of a distance matrix, whose number of entries grows proportionally to the square of the number of documents. This leads to slow iterations and a large memory footprint. This is unlike in the prior approach, as there only exists a total of 15317 unique topics. Possible solutions include running the process iteratively, only utilizing a subset of the total documents or leaving a majority of the documents with overly specific topics. As none of these options were appealing or guaranteed to result in a good labeling, we chose the approach of clustering based on topic embeddings.

The automatically generated topics were used as guidance for both the human annotator and to select which documents will be manually labeled. First, the frequency of each topic $z_i \in \boldsymbol{z}$ was calculated: $\boldsymbol{f} = \{f_1, ..., f_{|\boldsymbol{z}|}\}$. Then, n_i random documents were chosen from each topic z_i in order from the most frequent to the least frequent. The equation for calculating n_i was as follows:

$$n_i = \max\left(1, \lceil \frac{10 \cdot f_i}{||\boldsymbol{f}||_{inf}} \rceil\right) \tag{17}$$

Here, $\lceil \cdot \rceil$ stands for the ceiling function. Using Eq. 17, documents with the most frequent topic are labeled 10 times. Documents with less frequent topics are labeled in lesser amounts based on the topics' general occurrence rates in the whole dataset, but always at least once. This technique allows for creating a dataset that reflects the global occurrence rates of different topics within the training data. It is hypothesized that this should reduce overfitting and produce a more general model. Mimicking the topic distribution of the larger, 145 000 document dataset with the much smaller, manually annotated dataset would not be possible without these automatically generated topics. The resulting training data consists of conversations that are based on the TLLM prompting template, which can also be found in Appendix A. The interface for manual labeling can be seen in Fig. 10.

It is worth noting that the training data consisted of only 500 document-topic pairs. With such a low number of document-topic pairs, fine-tuning is computationally inexpensive and only takes an hour on a consumer-grade GPU. Furthermore, if annotating a single document takes 20 seconds on average, this dataset could be created in under 3 hours. Therefore, the granularity of the fine-tuned model's topics can be adjusted relatively quickly by creating a new dataset or modifying the previous one. Despite this seemingly easy procedure to fine-tune the model for different topic

granularities, it is worth clarifying that all answering styles might not be equally easy to converge towards with fine-tuning.

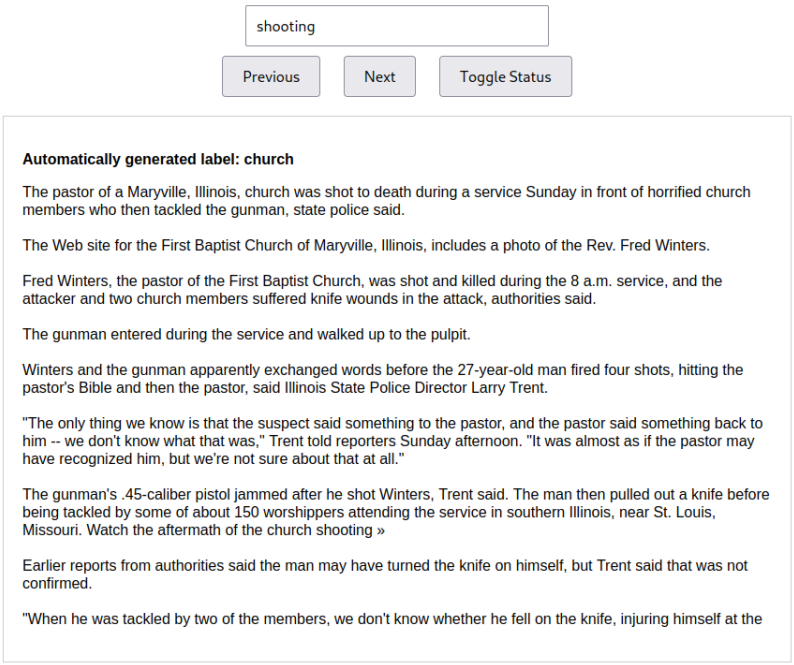


Figure 10: Interface for manual labeling of documents. The text box at the top can be used to input the correct labeling. The automatically generated labeling is shown in bold below the buttons. *(Previous)* Move to previous document. *(Next)* Move to next document. *(Toggle status)* Toggle whether the document should be marked as deleted.

3.2 Model

Llama 2 [167] is a large language model released by Meta in 2023. Three sizes of Llama 2 models have been released: models with 7, 13 and 70 billion parameters. All three sizes have both a foundation model and a chat-variant, which has been fine-tuned for instruction following. All variants of the model have a context length of 4096 tokens, which allows them to generally process prompts with thousands of words. The Llama 2 foundation models have become a popular choice within machine learning research due to their broad capabilities and a permissive usage license. The 7 billion parameter variant is used as the foundation model for TLLM. Early experiments were also conducted with the 7B-chat variant, but they proved unsuccessful. This was attributed to the low entropy of the fine-tuned model's outputs and the difficulty of removing its specific answering style without incurring catastrophic forgetting.

3.3 Training

The training was performed with QLoRA using a rank of 8, alpha of 32 and a learning rate of $2e^{-4}$. QLoRA was used to update the queries, keys and values of all attention layers, as is standard. The training was run for 4 epochs with a batch size of 2 and 8 steps of gradient accumulation, making the effective batch size 16. These hyperparameters were chosen to allow for training using a consumer-grade GPU with limited memory. The training used 200 warmup steps and a static dropout of 0.05, utilizing the transformers library by HuggingFace [184]. The model was trained for a total of 7 epochs. When trained for more epochs, the model became unlikely to produce topics that were not included in the training data, indicative of possible overfitting. After training, the model was transformed into the GGUF-format and all parameters were reduced to 8 bits with the conversion script provided by llama.cpp [43].

3.4 Inference

During training, the model learns to follow the answering style of *The general topic of this document is "**1-3 words**"*. For instance, the output of the model could be as follows:

```
### HUMAN:
What is the generic topic of this document?
Document: On December 13th last year, Parisians had to squint their
    eyes to be able to make out the iron girders of the normally
    imposing Eiffel Tower amid an unusually thick blanket of grey fog.

### RESPONSE:
The generic topic of this document is "pollution"
```

Listing 1: An example of a TLLM generation. The document was collected from the OpenWebText dataset [46] and truncated in the prompt for brevity.

The user prompt starts from "### HUMAN:" and ends in "### RESPONSE:", whereas the rest is generated by the model. However, as everything before the first quotation always remains unchanged, it can be either included in the prompt or prepended to the model's answer, which effectively achieve the same result. We also conducted experiments where the LLM was forced to answer with only the topic, omitting the prior declaration of "The generic topic of this document is". In these experiments, the model was unable to produce consistent topics. Goyal et al. find that providing LLMs a wider computational pathway improves generation quality [47]. The longer output could also be seen as an application of CoT, where the model is able to produce the answer one step at a time. Clearly, forcing the LLM to output only the topic goes against these principles. Another consideration is the LLM's answering style during unrestricted inference. Forcing the LLM to produce text in a style that it would not otherwise follow will, by definition, lead it towards responses it deems to

have a low likelihood. Thus, the prompting template was designed to be as similar as possible to the style in which the foundation model would answer questions before fine-tuning. The combination of these two factors could help explain the improvement in model performance when lengthening its output with tokens that could otherwise be seen as redundant.

4 Results

This section covers the resulting TLLM model and its capabilities. First, an overview of the model's high-level targets and how well they are achieved is provided. Following sections contain the results of a series of evaluations, analyzing the model's language understanding and quantifying its topic quality relative to comparable models.

(Generic) As mentioned in previous sections, the topics generated by the model should not be too specific or too generic. Generating topics that are too specific would, in the most extreme case, mean generating a unique topic for every document. On the other hand, the most generic topic model would classify all documents under one topic, perhaps the topic *document* or *text*. The correct granularity of topics is somewhere in between these extremes, not too specific as to lose its value as an analysis tool and not too generic as to lose its ability to distinguish between sufficiently different documents. The results shown in Fig. 11 and Fig. 12 provide evidence that, given a large dataset, the number of topics will remain reasonable. Further study showed that documents which receive the same topic are semantically similar (Table 2) and that the topics generated by TLLM are seen as reliable and helpful (Fig. 14). As the model exhibited zero failures during human evaluation (Table 4), it is clear that all of the documents were within TLLM's capabilities, despite their highly varying latent topics.

(Compute-efficient) In comparison to approaches such as LDA and BERTopic, TLLM is more computationally expensive. However, unsuccessful experiments conducted on producing a sufficiently generic BERTopic model led to much longer training durations than those experienced by TLLM. This comparison is also ill-defined; traditional models are unable to fill the role of TLLM, as discussed in prior sections. Additionally, Llama-2-7b has an identical number of parameters but is unable to produce coherent topics on its own, as seen in Table 4. Although the parameter count of GPT-3.5 is not publicly known, GPT-3 has 175 billion parameters [21] and the Llama 2 model used in human evaluation has 70 billion parameters. Clearly, TLLM with only 7 billion parameters is much more compute-efficient. The comparatively low parameter count of TLLM, combined with 8-bit quantization, allows for inference to be run on low-end graphics cards.

(Multilingual) We find that multilingual pretraining of the foundation model is enough for the model to exhibit an understanding of many different languages. As discussed in section 1, the model should be excellent in English, but also proficient in many other languages. As seen in Table 1, TLLM is able to infer obscure contextual information from documents written in Finnish. This, in combination with the model's clear abilities in English, suggests that it will be satisfactory at intermediately popular languages as well. This has also been empirically observed on languages, such as Chinese and Spanish.

(Streamable) The model is able to process documents fully in isolation. A topic can be produced for a document through simply prompting TLLM with no access to any other information. Effectively, the only limit to inference throughput is in the amount of parallelizable computing resources available.

(Length agnostic) While the model can process much longer documents than for instance BERTopic, its length requirements can still be limiting. The maximum length

of documents that TLLM can generate topics for is 4051 tokens, assuming that the generated topic consists of only one token. The prompting template consumes 44 tokens from the context length, along with every additional token used for the topic. This could be improved in future work by substituting the foundation model with one that is capable of processing longer inputs. A natural candidate would be the Mistral foundation model [69], as it has a context size of 8092 tokens and manifests many of the advantages of Llama 2, including its general language understanding and low parameter count. As seen in Fig. 11, TLLM generates significantly less unique topics for shorter documents when compared to Llama-2-70b-chat. This provides evidence that TLLM will generalize better with the types of documents commonly encountered on the internet. The documents used in Fig. 12 are longer, but TLLM still generates less unique topics than all the models it was compared against.

In addition to these factors, the topics generated by TLLM are easy to interpret and do not exhibit multiple meanings. For instance, documents with the topic *race* are seemingly all discussions related to human races and ethnicities, and not about racing competitions. Not only does this indicate that the model understands the differences between the word "race" in different contexts, the model also prefers to use the word for only one meaning, even if the word itself is highly homonymous. Secondly, the model is not limited to a specific list of possible topics. Some researchers define topic models as generating probability distributions over a fixed, finite vocabulary [102], but this can be unnecessarily restrictive. For instance, LSA, pLSA and LDA require the number of topics to be specified before training and BERTopic only maintains a static set of possible topics during inference. TLLM, on the other hand, does not have such limitations.

4.1 Language understanding

TLLM can use its knowledge about people, places and context on a non-trivial level during inference. To demonstrate this, we can take the following document, written in Finnish: "Minusta on alentavaa oikeita mestareita kohtaan kun käytätte **name of a person** nimitystä mestari. Voisiko sen lopettaa". When translated into English, this document stands for: "I find it offensive towards real champions when you call **name of a person** a champion. Could it be stopped". The document itself does not necessarily refer to any specific topic, other than perhaps *opinions* or *statements*, which would be overly non-descriptive. However, modifying the name given in the document gives it context and hints at an underlying topic. Were this statement said about movie actor, it would intuitively be in reference to films. Conversely, if the name referred to a politician, this would be a political statement. TLLM recognizes this context from simply the name given in the document. Table 1 provides a breakdown of what topics TLLM generates when altering the name given in the document. From these experiments, it would appear that the model also has some level of understanding related to Finnish individuals. When no name is provided (*him/her*, Finnish *hänestä*), TLLM generates the topic *language*. This indicates that the model interprets the document as talking about the semantic meaning of the word "champion". Similar levels of understanding related to names of individuals would be present in a LDA

model only if the training data explicitly mentioned them. Similarly, a BERTopic model would be able to understand such nuances only if the embedding model was able to include information related to the individual in the document embedding. Even if this context was understood, these models would not generate a concise topic, but a sequence of n-grams.

Table 1: Topics inferred for different individuals by TLLM. The results have been generated greedily with top-k=1, choosing the most probable token at every generation step. The document used in the prompt and the cursive name were written in Finnish.

Name	**Topic**
Teemu Selänne	sports
Halla-Aho	politics
Steven Spielberg	film
Frank Sinatra	music
Grace Hopper	computer science
Buzz Aldrin	space
him/her/them	language

4.2 Evaluation

Although many previous topic models exist, none are directly comparable to TLLM, which complicates evaluation. Topic models are often evaluated on two metrics: prediction accuracy and topic interpretability [35]. For topic models that use collections of n-grams as topic representations, topic interpretability can be further split into topic cohesion and topic diversity [176, 13, 53, 18]. Intuitively, if the n-grams within the same topic are not related to each other, it is a strong indicator of poor topic quality. The current standard for evaluating topic cohesion is Normalized Mutual Pointwise Information (NPMI) [65, 18], where a score of 1 stands for perfect cohesion and 0 for no cohesion. On the other hand, topic diversity measures the breadth of the model's range of different topics. Dieng et al. [35] evaluated topic diversity as the percentage of unique n-grams in the first 25 n-grams of all topics [1]. None of these metrics can be used with TLLM, as topics are represented by a single coherent term consisting of 1-3 words. Furthermore, a list of possible topics could either be argued to not exist or be infinitely large, as the model is only limited by the amount of whitespaces and not by the amount of tokens used. For example, Rijken et al. [148], whose model produced topic descriptions with a LLM, relied solely on human evaluation in their testing. As a consequence of these factors, TLLM is evaluated on three different benchmarks: generalizability 4.2.1, clustering performance 4.2.2 and human evaluation 4.2.3.

The evaluations make use of two different datasets. Firstly, the *20 Newsgroups dataset* [85] consists of 18846 emails from 20 different newsgroups. The emails are not strictly related to the newsgroup they were sent to, which makes the dataset relatively noisy. The emails are also moderately short, 284 words on average. This dataset is

used for studying the topic generation process for larger quantities of shorter, noisier documents. Secondly, the *BBC dataset* consists of 2228 news articles published by the BBC between 2004-2005 [51]. All of the articles belong to one of five different categories: business, entertainment, politics, sport and tech. The articles are relatively long, 691 words on average. This dataset is used to compare TLLM to multiple other models, as the lower number of documents makes inference with larger models quicker and cheaper.

TLLM is evaluated against three LLMs: Llama-2-70b-chat, GPT-3.5-turbo-0125 and Llama-2-7b. GPT-3.5-turbo was further prompted using two templates; both with and without ICL. As discussed previously, traditional topic models have focused on topics in the form of n-gram sequences, and are not capable of internet-scale topic modeling. This, combined with the fact that they usually model documents as mixtures of multiple topics, would have made a comparison against them ill-defined. GPT-3.5-turbo is used through the OpenAI API [20], while the others were run locally on a consumer-grade GPU. Llama-2-70b-chat was quantized to 4 bits to make inference more feasible, while TLLM and Llama-2-7b were quantized to 8 bits. Topics were always generated with top-k of 1. This forced the models to always generate the token that was perceived as the most probable during inference. This reduces randomness and makes the results replicable. All prompting templates can be found in Appendix A.

4.2.1 Generalizability

An internet-scale topic model should produce only a moderate number of topics for a given dataset. Even for large datasets, the number of unique topics should not grow too large to be effectively analyzed. In this thesis, this capability is referred to as the model's generalizability. To study the model's generalizability, we plot the number of unique topics as a function of the total number of documents processed. A similar approach has been previously used to study the complexity of different natural languages by Varjokallio et al. [171]. Intuitively, the rate of increase in the number of unique topics should decrease as the number of topics increases. This is a consequence of the previously generated topics spanning a larger subset of the latent topics present in the documents. In this evaluation, a topic was generated for each document in the *20 Newsgroups dataset* individually by both TLLM and Llama-2-70b-chat. The results are visualized in Fig. 11. The plot includes the names of topics at the document indices, where they were generated for the tenth time. The number of topics shown is restricted to prevent visual overlap. The same randomized document order was shared by both models. In summary, TLLM created a total of 639 unique topics. This means that the number of topics was 3.4% of the total document count. In comparison, Llama-2-70b-chat generated 1855 unique topics, roughly 3 times as many.

Furthermore, Fig. 12 compares the number of unique topics generated by all the different models on the *BBC dataset*. As is evident in the figure, TLLM generated the least unique topics, followed by Llama-2-70b-chat and GPT-3.5-turbo with the ICL prompt. The most unique topics were generated by GPT-3.5-turbo with the non-ICL prompt and the Llama-2-7b foundation model. The difference between the number of

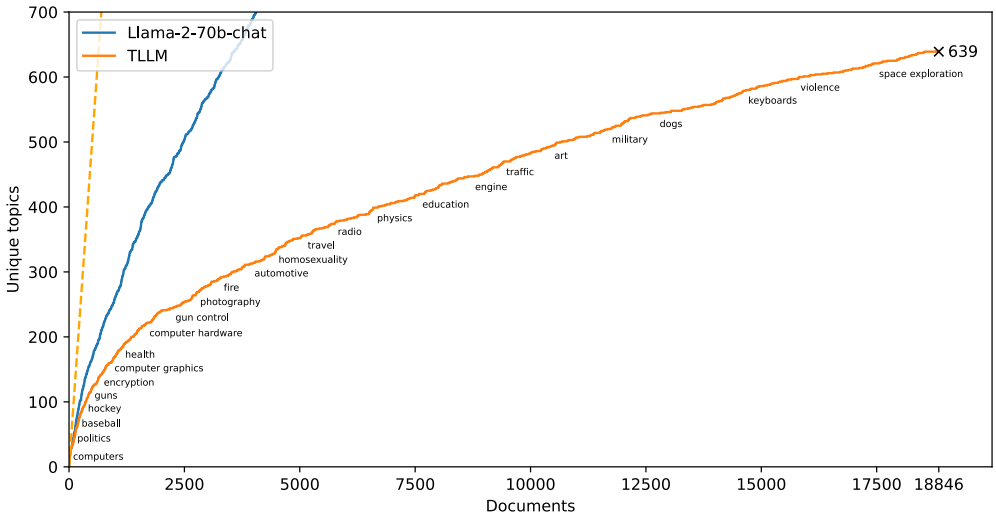


Figure 11: A plot of the number of unique topics generated by TLLM and Llama-2-70b-chat as a function of the total number of documents. The documents are from the *20 Newsgroups dataset*. The dotted orange line represents $y = x$, where every document would be generated a unique topic.

unique topics generated by TLLM and Llama-2-70b-chat is not as drastic as in the *20 Newsgroups dataset*. This could be attributed to the longer length of the documents: longer documents generally make the topic more obvious, which makes topic modeling easier. Similarly to the results shown in Fig. 11, the same randomized document order was shared by all the models.

4.2.2 Clustering performance

A critical assumption of any topic model is that it generates similar topics for similar documents, while dissimilar documents receive different topics. One way to estimate the model's performance in this regard is to generate topics for documents in a corpus, and then interpret the documents that received the same topic as being in the same cluster. Then, it is possible to analyze the intra-cluster similarities of the documents contrasted against their inter-cluster similarities. We take the approach of embedding the documents and then calculating similarities between them using cosine similarity. The embeddings are in high-dimensional space (768) and calculated with a SBERT model (all-mpnet-base-v2) [146, 147]. For calculating intra-cluster similarities, we simply average the similarities between all the documents within the same cluster. On the other hand, inter-cluster similarity is calculated by averaging the similarities between all documents inside the cluster with documents outside the cluster. For two distinct clusters of documents $\mathcal{A}$ and $\mathcal{B}$, with sizes $|\mathcal{A}|$ and $|\mathcal{B}|$, inter-cluster similarity is therefore measured as:

$$\frac{1}{|\mathcal{A}||\mathcal{B}|} \sum_{x \in \mathcal{A}} \sum_{y \in \mathcal{B}} S_c(x, y) \tag{18}$$

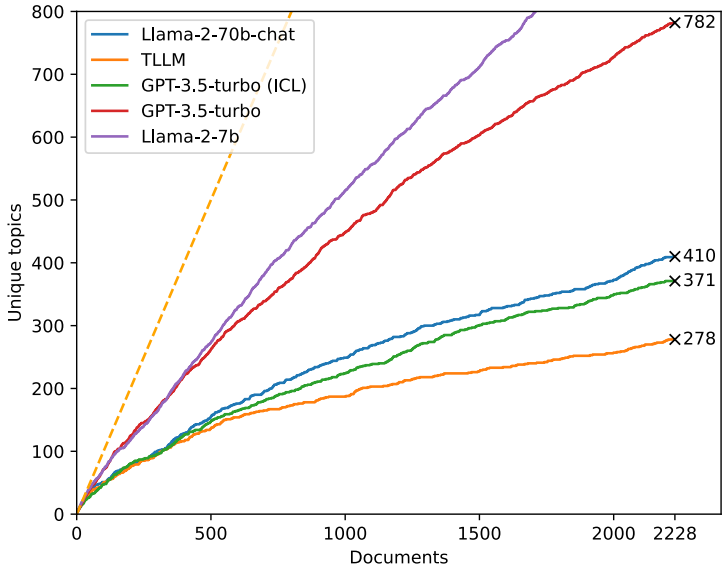


Figure 12: Number of unique topics generated as a function of number of documents processed for different models using the *BBC dataset*. The dotted orange line represents $y = x$, where every document would be generated a unique topic.

This is akin to the Unweighted Pair Group Method with Arithmetic mean (UPGMA), alternatively known as the average linkage method, used in agglomerative clustering [161]. These similarities can then be averaged over the entire clustering. Eq. 19 is used to calculate the mean intra-cluster similarity for the whole dataset, while Eq. 20 is used to calculate the mean inter-cluster similarity.

$$\frac{1}{|C|}\sum_{\mathcal{A}\in C}\frac{1}{|\mathcal{A}|^2}\sum_{x\in\mathcal{A}}\sum_{y\in\mathcal{A}}S_c(x,y) \tag{19}$$

$$\frac{1}{|C|(|C|-1)}\sum_{\mathcal{A}\in C}\sum_{\mathcal{B}\in C\setminus A}\frac{1}{|\mathcal{A}||\mathcal{B}|}\sum_{x\in\mathcal{A}}\sum_{y\in\mathcal{B}}S_c(x,y) \tag{20}$$

Intuitively, the intra-cluster similarity should be higher than inter-cluster similarity; the larger the difference, the better. However, a significant difference is not realistic given the dataset and desired granularity of topics. A significant difference would mean that the topics are specific and detailed, which is not desirable. For the *20 Newsgroups dataset*, TLLM generates a clustering where the mean intra-cluster similarity is 0.74 and mean inter-cluster similarity is 0.31. On the other hand, Llama-2-70b-chat creates a clustering with a mean intra-cluster similarity of 0.84 and a mean inter-cluster similarity of 0.32. These results indicate that TLLM generates slightly less coherent clusters when compared to Llama-2-70b-chat, which is expected given its significantly lower number of clusters, as seen in Fig. 11.

Table 2 shows the mean intra and inter-cluster similarities of different models on the *BBC dataset*. As is apparent in the table, the largest intra-cluster similarities were achieved by the models that generated the most unique topics. This is expected: any

topic that was generated for only one document has the intra-cluster similarity of 1. Furthermore, the rate of decrease in intra-cluster similarities appears to closely follow the total number of clusters. It is worth highlighting that TLLM is able to maintain an intra-cluster similarity of 0.74 with 32% less unique topics when compared to Llama-2-70b-chat.

Table 2: Intra and inter-cluster similarities of topic clusterings generated by different models on the *BBC dataset*.

Model	**Unique topics**	**Intra-cluster similarity**	**Inter-cluster similarity**
TLLM	278	0.74	0.10
GPT-3.5-turbo (ICL)	371	0.75	0.10
Llama-2-70b-chat	410	0.80	0.10
GPT-3.5-turbo	782	0.83	0.11
Llama-2-7b	1011	0.94	0.10

4.2.3 Human evaluation

A human evaluation was conducted as a final measure of model capabilities. The evaluation included a total of 100 documents, sampled randomly from the *BBC dataset*. Five different topics were generated for each document using the same models and prompting templates described previously. In the evaluation interface, the user was presented with the document, along with the topic candidates in a random ordering. The evaluator was not informed which model generated which topic candidate. Each evaluator was asked to give each topic candidate a score between 1 and 3 with the following instructions:

> Assume that there are tens/hundreds of millions of online documents being analyzed. Grade the following topic candidates in terms of how helpful they would be.
> The grades are as follows:
> **1** Bad, not useful or even misleading
> **2** OK, possibly useful.
> **3** Good, probably useful.
> If a topic is too specific ("stock marker disaster" etc), you'd have to investigate many topics to get the desired result. If a topic is too broad ("opinion" etc), the documents within it will often not be related to each other.

The evaluation interface is shown in Fig. 13. The users performed the labeling independently and with no further guidance. The human evaluation was performed by four different people, with varying degrees of familiarity with this thesis. The

evaluators were selected by sending an open invitation to a group of roughly 15 people whose work relates to data analytics, and choosing the first three people who responded to it. The fourth evaluator was the main author of this thesis. Consequently, all respondents had prior familiarity with analyzing large collections of text.

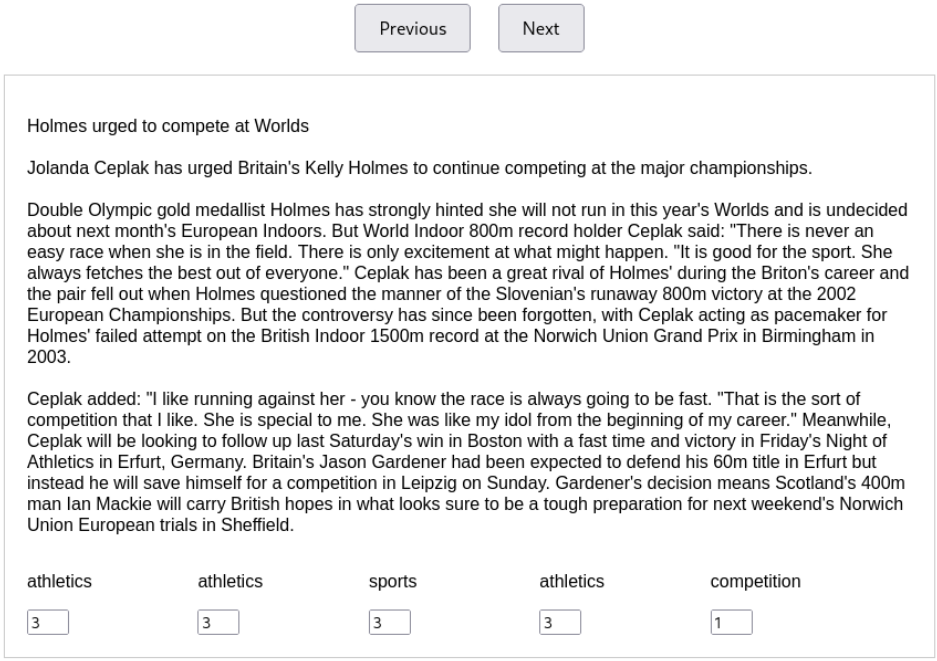


Figure 13: The user interface for evaluation topic candidates. The order in which different models' topic candidates are shown is randomized for each document and hidden from the user.

The results are shown in Table 3 and visualized in Fig. 14. In summary, the best average scores were achieved by Llama-2-70b-chat and TLLM. They were followed by GPT-3.5-turbo with and without ICL. The worst average score was received by Llama-2-7b, the foundation model used to train TLLM. The standard deviations and standard error means are calculated from the document-wise mean scores. More specifically, they are the standard deviations and standard error means of 100 mean scores, calculated from the four different evaluations for each generated topic. This is done in an effort to mitigate the effects of the evaluators' differing average grades. In essence, these values tell the variance of the model's expected performance, when averaged over disagreements between evaluators.

Table 3: Results of the human evaluation for all the models. The standard deviations and standard error means are calculated from document-wise mean scores.

Model	Mean score	Standard deviation	Standard error mean
Llama-2-70b-chat	2.66	0.28	0.03
TLLM	2.62	0.27	0.03
GPT-3.5-turbo (ICL)	2.53	0.42	0.04
GPT-3.5-turbo	2.42	0.50	0.05
Llama-2-7b	1.82	0.78	0.08

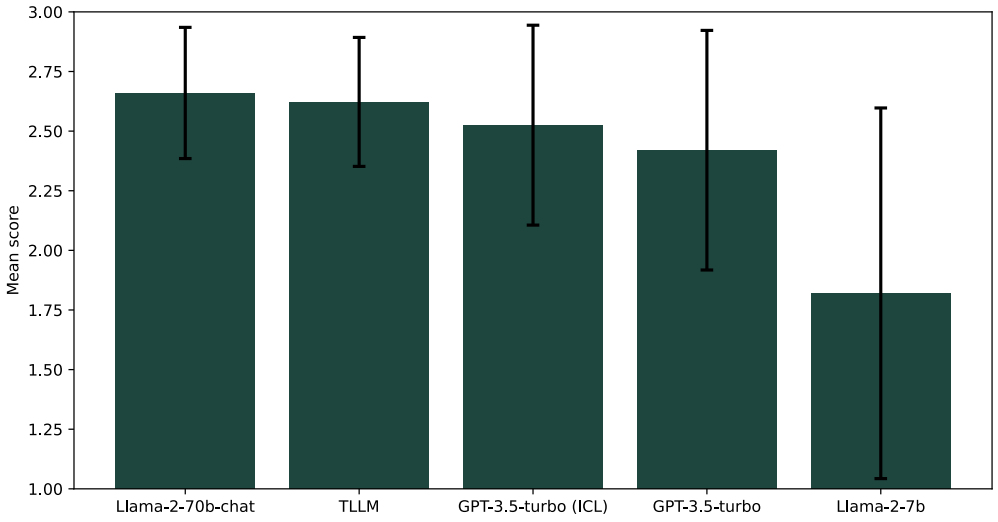


Figure 14: The mean scores of each topic model. The whiskers drawn in black denote the standard deviations of document-wise mean scores.

The results indicate that despite its relatively low number of parameters, TLLM is a very capable topic model. It was surpassed with only a minor margin by a model ten times its size. Furthermore, as TLLM maintained lower topic count on the *20 Newsgroups dataset*, the results may have been different if the human evaluation was performed with shorter documents. On the other hand, these results also indicate that it could be beneficial to retrain TLLM with a larger foundation model, if the added computational complexity during inference is deemed acceptable. GPT-3.5-turbo also performed quite well, but its scores were hindered by the model being too specific. For instance, given a news story about a law that bans hunting with dogs, GPT-3.5-turbo generated the topic *hunting ban law*. Such topics were deemed too precise by the evaluators. This could be a result of the answering style learned during RLHF training, as all other models in the evaluation have only been trained with SFT. For instance, Shen et al. conclude that RLHF introduces language biases with respect to response length, making the model prefer longer responses [157]. This conclusion has also

been drawn by other authors [152, 89]. In the case of GPT-3.5-turbo, modifying the prompt to include ICL improved its scores noticeably.

Fig. 15 shows the models' score distributions. As is evident, TLLM received slightly less scores of one point (12) when compared to Llama-2-70b-chat (14), but in turn had more scores of two points (TLLM: 127, Llama-2-70b-chat: 108). The other models' distributions reinforce the results from previous analysis: ICL improved the results of GPT-3.5-turbo and Llama-2-7b is unable to produce accurate topics. Notably, the number of worst scores received by GPT-3.5-turbo dropped from 49 to 29 with the more performant ICL prompt.

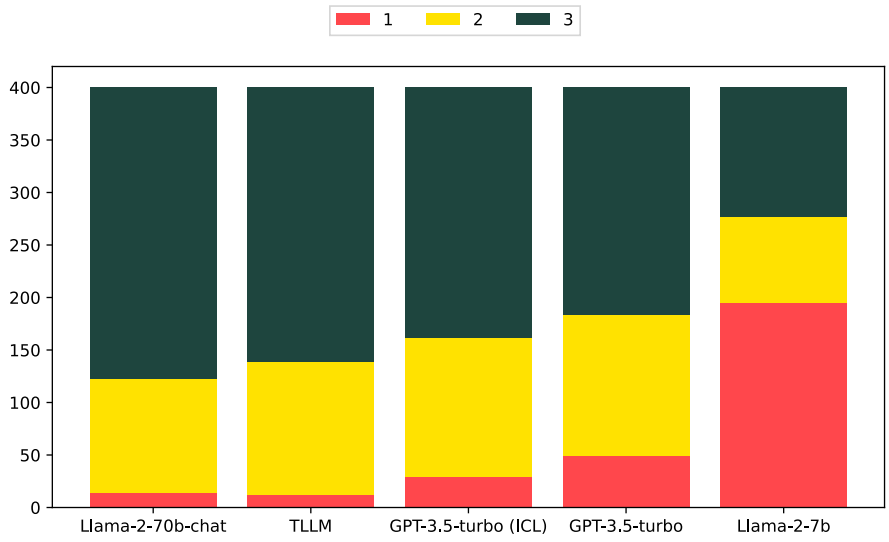


Figure 15: Visualization of the models' score distributions.

Fig. 16 further showcases the differences in how the four evaluators perceived the topics generated by the models. While some evaluators preferred either TLLM or Llama-2-70b-chat, the order of the other three models remained mostly static. Evaluator 3 was an exception to this rule. They gave the topics of GPT-3.5-turbo the best score more frequently when the prompt did not use ICL, seemingly preferring the more specific topics. However, they also gave the topics of GPT-3.5-turbo the lowest score more frequently when prompted without ICL. This means that the model's topics were not strictly better; they were more concentrated between the two extremes.

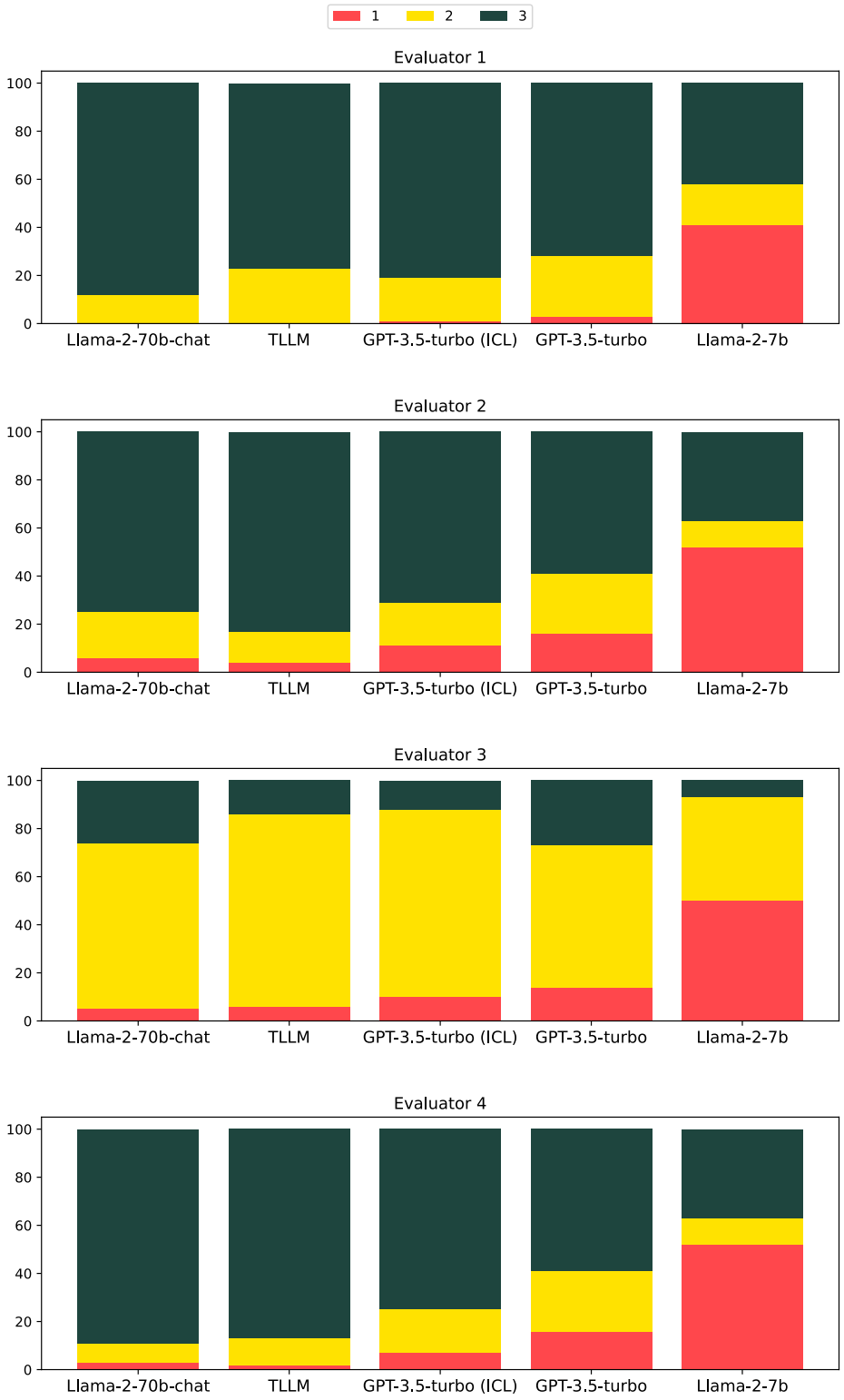


Figure 16: Score distributions for each evaluator.

Table 4 shows the number of *failures* for each model. A failure occurs when the model's topic candidate receives the worst score from at least half of the evaluators. The worst performing model, Llama-2-7b, had a failure rate of 50%. This is unsurprising, as the model would often simply repeat words from the document or combine mutually unrelated terms. TLLM exhibited zero failures, while Llama-2-70b-chat had two. The failure rate of GPT-3.5 was significantly improved by ICL, which reduced it by half. Despite this, GPT-3.5-turbo still experienced 8 failures with the more performant prompt. Another finding of the human evaluation results is that a relatively modest time investment was able to improve the foundation model in such a drastic way.

Table 4: The number of failures for each model in the 100 document human evaluation. A failure occurs when at least half of the human evaluators give the model's generated topic a score of 1.

Model	**Number of failures**
TLLM	0
Llama-2-70b-chat	2
GPT-3.5-turbo (ICL)	8
GPT-3.5-turbo	16
Llama-2-7b	50

4.3 Limitations

Recent research suggests that LLMs do not use the entire context equally [79] and ignore especially the middle of the prompt [103]. This can pose problems if information crucial to the document's topic is present in the middle of the document. Similar limitations have not been found for the other topic modeling approaches discussed in this thesis. Furthermore, TLLM can only generate topics for documents with a maximum length of approximately 4000 tokens. Documents longer than this limit have been left out of the scope of this thesis.

During model evaluation and generation of the training data, more sophisticated ways of document truncation could have been used. The training documents were truncated to 512 characters, which sometimes led to unfinished sentences and incomplete words. During evaluation, documents were truncated to 3900 characters. This means that the full context of the LLMs was not always used, and the prior problem of incomplete sentences and words was also present.

The topics generated by TLLM are not explainable and there is no guaranteed way to estimate its accuracy. A direction worth pursuing could be the implementation of the ASPIRE framework into TLLM training. Furthermore, it could be studied whether the generated tokens' log probabilities are indicative of topic correctness. Related to accuracy, methods that improve LLM question answering capabilities at the price of computational complexity might prove an interesting future research direction for TLLM. These include the previously mentioned beam search and pause tokens, among others. Some topic models have the ability to approximate the ratios of different topics

present in the document. While these approaches allow for a document to contain multiple topics in differing amounts, TLLM has been trained to only generate one topic for each document.

A significant drawback with TLLM is that it is not able to infer when a document contains no topic. This causes especially noisy documents to gravitate towards the same topic. In previous iterations of TLLM, this topic has been either *politics* or *business*, leading to these topics being saturated with low-quality documents. This could be partially alleviated by setting a minimum length for documents, in addition to a maximum length. However, this is not a perfect solution as long documents could still consist of nonsense and short documents could have very distinct topics. This drawback is also shared by all other topic models discussed in this thesis. Documents which have only a vague topic can also be over-represented when analyzing certain topics. The document *I would like that very much* might be given the topic *opinions*, even though the document itself does not hold any meaningful information about any opinion. It could be beneficial modify the TLLM training data to include a label for when there is no topic. However, such experiments are left for further study.

TLLM suffers from the fact that topics are not unequivocal or unambiguous. A news story about a basketball team winning a match could have one of many topics. The topic *basketball* might be the most intuitive, but *sports*, *championships*, *competitions*, *sports news*, *basketball news* are also reasonable. This is especially apparent in sports, where the training topics were created such that frequently discussed sports like football, basketball and hockey would have their own topic; less popular sports would either be grouped together into a larger category or just labeled *sports*. For example, formula one racing was labeled *motorsports* and long jumping was labeled *athletics*, while snooker was simply labeled *sports*. The idea behind this labeling scheme was to split the large *sports* cluster into coherent and somewhat equally sized subtopics. However, this behavior proved difficult to replicate consistently during inference. In other words, documents about basketball might be given the topic *basketball*, but also *sports* when labeled with TLLM. Similarly, documents about ice hockey might receive the topic *ice hockey*, *hockey* or *sports*. Therefore, extra care has to be taken when using the topics to analyze a dataset. The same applies to singular and plural topics, as TLLM might generate either the topic *sport* or *sports*, *animal* or *animals*, and so on. The topics used for training were consistently plural, but this behavior is not perfectly replicated by the model. It is worth noting that these challenges are present in all topic models discussed in this thesis to varying degrees.

TLLM has blind spots in terms of information and is not knowledgeable about everything. Some of these blind spots could most likely be alleviated with more training data, but others are most likely a result of the model's low parameter count. Furthermore, the model needs to be updated to maintain its ability to recognize recent events, people and places. As the topics are very generic, this will most likely not be a problem in the near future, but in time the model will inevitably fall behind on relevant information. A natural way to achieve this would be to change the foundation model to one that has been trained with more recent data. As long as new foundation models are released frequently, this should not pose a problem, especially given the low time investment required to retrain TLLM once the dataset has been acquired.

TLLM is more opinionated compared to models that rely solely on word frequencies or whose topics are represented by collections n-grams from training documents. Therefore, the internal biases of the LLM might affect the generated topics. Rozado et al. use ChatGPT to fill different political surveys, and find that it has a tendency to favour the politics of progressive, left-leaning libertarians [150]. More recently, Motoki et al. find that ChatGPT has a significant political bias towards the Democrats in the US and the Labour Party in the UK [118]. In terms of TLLM, this could affect the probabilities of generating topics such as *terrorism*, *dictator* and *corruption*, as such definitions are naturally dependent on the opinions, cultural understanding and political views of the respondent. Studies of the model's biases should be conducted on the foundation model; it is feasible to curate the fine-tuning data for such biases, unlike the much larger pretraining dataset.

5 Conclusions

This thesis presented TLLM, the first topic model capable of internet-scale analysis. We defined a topic as a human-interpretable description of a document's contents that is at most three words long. With this definition, we proposed an approach to generating balanced and generic datasets of document-topic pairs for LLM fine-tuning. This method was used to fine-tune Llama-2-7b for the task of topic modeling. The resulting model, TLLM, generalizes exceptionally well to large datasets of unseen data and exhibits levels of multilinguality and general knowledge not present in prior approaches. TLLM also generates topics which group large quantities of documents under the same term, enabling corpus-wide analysis. Human evaluation revealed that TLLM either matches or slightly exceeds the performance of Llama-2-70b-chat and GPT-3.5-turbo, models much larger than itself. Out of all tested models, TLLM was the only model to never produce a failure during human evaluation. TLLM also showed promising results in its ability to produce topics for shorter documents when compared to Llama-2-70b-chat.

In summary, this thesis proves two factors. First, sufficiently generic modeling is possible using general-purpose instruction following LLMs with both openly available and restricted weights, given an optimized prompting template. Especially the use of ICL appears to assist in topic generation. Secondly, it is possible to train a capable topic modeling LLM with only a modest time investment and seven billion parameters. The benefits of the latter are most apparent in topic granularity and compute efficiency. TLLM also has some weaknesses, such as its inability to estimate its own accuracy, process documents longer than approximately 4000 tokens and recognize when a document has no discernible topic. Although the possible solutions to these drawbacks are left for future study, there are clear ideas to mitigating all major challenges faced by TLLM. Furthermore, these mitigations are not necessary for the model to fulfill its primary functions; rather, they merely serve to improve its capabilities. As the use of LLMs within topic modeling is largely a novel application and fine-tuning LLMs for topic modeling has not been studied before, this work outlines a possible future direction for the field in the era of transformers.

References

[1] Aly Abdelrazek, Yomna Eid, Eman Gawish, Walaa Medhat, and Ahmed Hassan. Topic modeling algorithms and applications: A survey. *Information Systems*, page 102131, 2022.

[2] Charu C Aggarwal, Alexander Hinneburg, and Daniel A Keim. On the surprising behavior of distance metrics in high dimensional space. In *Database Theory—ICDT 2001: 8th International Conference London, UK, January 4–6, 2001 Proceedings 8*, pages 420–434. Springer, 2001.

[3] Shun-ichi Amari. Learning Patterns and Pattern Sequences by Self-Organizing Nets of Threshold Elements. *IEEE Transactions on Computers*, C-21:1197–1206, 1972.

[4] Murugan Anandarajan, Chelsey Hill, Thomas Nolan, Murugan Anandarajan, Chelsey Hill, and Thomas Nolan. Term-document representation. *Practical Text Analytics: Maximizing the Value of Text Data*, pages 61–73, 2019.

[5] Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen Marcus McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. Llemma: An Open Language Model for Mathematics. In *The Twelfth International Conference on Learning Representations*, 2024.

[6] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR*, abs/1409.0473, 2014.

[7] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback. *arXiv preprint arXiv:2204.05862*, 2022.

[8] Georgios Balikas, Massih-Reza Amini, and Marianne Clausel. On a topic model for sentences. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, page 921–924, New York, NY, USA, 2016. Association for Computing Machinery.

[9] David Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.

[10] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The Long-Document Transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[11] Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? FAccT '21, page 610–623, New York, NY, USA, 2021. Association for Computing Machinery.

[12] Lukas Berglund, Meg Tong, Maximilian Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. The Reversal Curse: LLMs trained on "A is B" fail to learn "B is A". In *The Twelfth International Conference on Learning Representations*, 2024.

[13] Federico Bianchi, Silvia Terragni, and Dirk Hovy. Pre-training is a Hot Topic: Contextualized Document Embeddings Improve Topic Coherence. In *Annual Meeting of the Association for Computational Linguistics*, 2020.

[14] Federico Bianchi, Silvia Terragni, Dirk Hovy, Debora Nozza, and Elisabetta Fersini. Cross-lingual Contextualized Topic Models with Zero-shot Learning. In *Conference of the European Chapter of the Association for Computational Linguistics*, 2020.

[15] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022, 2003.

[16] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[17] Kaj Bostrom and Greg Durrett. Byte Pair Encoding is Suboptimal for Language Model Pretraining. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4617–4624, Online, November 2020. Association for Computational Linguistics.

[18] Gerlof Bouma. Normalized (pointwise) mutual information in collocation extraction. *Proceedings of the Biennial GSCL Conference 2009*, 30:31–40, 2009.

[19] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 632–642, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[20] Greg Brockman, Peter Welinder, Mira Murati, and OpenAI. OpenAI: OpenAI API, 2020. URL https://openai.com/blog/openai-api. (Accessed 27-March-2024).

[21] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language Models are Few-Shot Learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

[22] Sondos Mahmoud Bsharat, Aidar Myrzakhan, and Zhiqiang Shen. Principled Instructions Are All You Need for Questioning LLaMA-1/2, GPT-3.5/4. *arXiv preprint arXiv:2312.16171*, 2023.

[23] Christian Buck, Kenneth Heafield, and Bas van Ooyen. N-gram Counts and Language Models from the Common Crawl. In *International Conference on Language Resources and Evaluation*, 2014.

[24] Jiefeng Chen, Jinsung Yoon, Sayna Ebrahimi, Sercan Ö. Arik, Tomas Pfister, and Somesh Jha. Adaptation with Self-Evaluation to Improve Selective Prediction in LLMs. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

[25] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics.

[26] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. PaLM: Scaling Language Modeling with Pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

[27] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[28] Common Crawl statistics, 2024. URL https://commoncrawl.github.io/cc-crawl-statistics/plots/languages. (Accessed 08-March-2024).

[29] Tri Dao. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. In *The Twelfth International Conference on Learning Representations*, 2024.

[30] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness.

In *Advances in Neural Information Processing Systems*, volume 35, pages 16344–16359. Curran Associates, Inc., 2022.

[31] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[32] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. GPT3.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems*, 2022.

[33] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient Finetuning of Quantized LLMs. *ArXiv*, abs/2305.14314, 2023.

[34] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *North American Chapter of the Association for Computational Linguistics*, 2019.

[35] Adji B Dieng, Francisco JR Ruiz, and David M Blei. Topic Modeling in Embedding Spaces. *Transactions of the Association for Computational Linguistics*, 8:439–453, 2020.

[36] Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5(3):220–235, 2023.

[37] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, Lei Li, and Zhifang Sui. A Survey for In-context Learning. *ArXiv*, abs/2301.00234, 2023.

[38] Yuheng Du, Alexander Herzog, Andre Luckow, Ramu Nerella, Christopher Gropp, and Amy Apon. Representativeness of latent dirichlet allocation topics estimated from data samples with application to common crawl. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 1418–1427. IEEE, 2017.

[39] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databaseswith Noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996.

[40] John Firth. A Synopsis of Linguistic Theory, 1930-1955. *Studies in linguistic analysis*, pages 10–32, 1957.

[41] Markus Freitag and Yaser Al-Onaizan. Beam Search Strategies for Neural Machine Translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 56–60, Vancouver, August 2017. Association for Computational Linguistics.

[42] Saibo Geng, Martin Josifoski, Maxime Peyrard, and Robert West. Grammar-constrained decoding for structured NLP tasks without finetuning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 10932–10952, 2023.

[43] Georgi Gerganov. llama.cpp, 2023. URL https://github.com/ggerganov/llama.cpp. (Accessed 17-January-2024).

[44] Deepanway Ghosal, Navonil Majumder, Ambuj Mehrish, and Soujanya Poria. Text-to-Audio Generation using Instruction-Tuned LLM and Latent Diffusion Model. *arXiv preprint arXiv:2304.13731*, 2023.

[45] Amir Globerson, Gal Chechik, Fernando Pereira, and Naftali Tishby. Euclidean Embedding of Co-occurrence Data. *Journal of Machine Learning Research*, 8(76):2265–2295, 2007.

[46] Aaron Gokaslan, Vanya Cohen, Ellie Pavlick and Stefanie Tellex. OpenWebText Corpus, 2019. URL http://Skylion007.github.io/OpenWebTextCorpus. (Accessed 29-September-2023).

[47] Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. Think before you speak: Training Language Models With Pause Tokens. In *The Twelfth International Conference on Learning Representations*, 2024.

[48] Sachin Goyal, Ananya Kumar, Sankalp Garg, Zico Kolter, and Aditi Raghunathan. Finetune like you pretrain: Improved finetuning of zero-shot vision models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19338–19347, 2023.

[49] Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. Bidirectional LSTM networks for improved phoneme classification and recognition. In *International Conference on Artificial Neural Networks*, pages 799–804. Springer, 2005.

[50] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5):602–610, 2005. IJCNN 2005.

[51] Derek Greene and Pádraig Cunningham. Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 377–384, 2006.

[52] Thomas Griffiths, Michael Jordan, Joshua Tenenbaum, and David Blei. Hierarchical topic models and the nested Chinese restaurant process. *Advances in Neural Information Processing Systems*, 16, 2003.

[53] Maarten Grootendorst. BERTopic: Neural topic modeling with a class-based TF-IDF procedure. *arXiv preprint arXiv:2203.05794*, 2022.

[54] Albert Gu and Tri Dao. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[55] Wes Gurnee and Max Tegmark. Language Models Represent Space and Time. In *The Twelfth International Conference on Learning Representations*, 2024.

[56] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching Machines to Read and Comprehend. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[57] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[58] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9:1735–1780, 1997.

[59] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Thomas Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karén Simonyan, Erich Elsen, Oriol Vinyals, Jack Rae, and Laurent Sifre. An empirical analysis of compute-optimal large language model training. In *Advances in Neural Information Processing Systems*, volume 35, pages 30016–30030. Curran Associates, Inc., 2022.

[60] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, page 50–57, New York, NY, USA, 1999. Association for Computing Machinery.

[61] MohammadSaleh Hosseini, Munawara Saiyara Munia, and Latifur Khan. BERT has more to offer: BERT layers combination yields better sentence embeddings. In *The 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

[62] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6):417, 1933.

[63] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly.

Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*, pages 2790–2799. PMLR, 2019.

[64] Jeremy Howard and Sebastian Ruder. Universal Language Model Fine-tuning for Text Classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[65] Alexander Miserlis Hoyle, Pranav Goel, and Philip Resnik. Improving Neural Topic Models Using Knowledge Distillation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

[66] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

[67] HuggingFace. Wikipedia Dataset. URL https://huggingface.co/datasets/wikipedia. (Accessed 20-September-2023).

[68] Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. Poly-encoders: Architectures and Pre-training Strategies for Fast and Accurate Multi-sentence Scoring. In *International Conference on Learning Representations*, 2020.

[69] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7B. *arXiv preprint arXiv:2310.06825*, 2023.

[70] Dan Jurafsky. *Speech and Language Processing*. Pearson Education India, 2000.

[71] Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and C. Rupprecht. CoTracker: It is Better to Track Together. *ArXiv*, abs/2307.07635, 2023.

[72] Feyza Duman Keles, Pruthuvi Mahesakya Wijewardena, and Chinmay Hegde. On The Computational Complexity of Self-Attention. In *International Conference on Algorithmic Learning Theory*, pages 597–619. PMLR, 2023.

[73] Henry J Kelley. Gradient Theory of Optimal Flight Paths. *ARS Journal*, 30(10):947–954, 1960.

[74] Eamonn Keogh and Abdullah Mueen. Curse of Dimensionality. In *Encyclopedia of Machine Learning and Data Mining*, pages 314–315. Springer US, Boston, MA, 2017.

[75] Ryan Kiros, Yukun Zhu, Russ R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. *Advances in Neural Information Processing Systems*, 28, 2015.

[76] Yuval Kirstain, Patrick Lewis, Sebastian Riedel, and Omer Levy. A Few More Examples May Be Worth Billions of Parameters. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1017–1029, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.

[77] Jing Yu Koh, Daniel Fried, and Russ R Salakhutdinov. Generating Images with Multimodal Language Models. In *Advances in Neural Information Processing Systems*, volume 36, pages 21487–21506. Curran Associates, Inc., 2023.

[78] Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large Language Models are Zero-Shot Reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213. Curran Associates, Inc., 2022.

[79] Jannik Kossen, Yarin Gal, and Tom Rainforth. In-Context Learning Learns Label Relationships but Is Not Conventional Learning. In *The Twelfth International Conference on Learning Representations*, 2024.

[80] Suhas Kotha, Jacob Mitchell Springer, and Aditi Raghunathan. Understanding Catastrophic Forgetting in Language Models via Implicit Inference. In *The Twelfth International Conference on Learning Representations*, 2024.

[81] Taku Kudo. Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[82] Hye-Jin Kwon, Hyun-Jeong Ban, Jae-Kyoon Jun, and Hak-Seon Kim. Topic Modeling and Sentiment Analysis of Online Review for Airlines. *Information*, 12(2):78, 2021.

[83] Massimo La Rosa, Antonino Fiannaca, Riccardo Rizzo, and Alfonso Urso. Probabilistic topic modeling for the analysis and classification of genomic sequences. *BMC Bioinformatics*, 16(6):1–9, 2015.

[84] Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, Timo Ewalds, Zach Eaton-Rosen, Weihua Hu, Alexander Merose, Stephan Hoyer, George Holland, Oriol Vinyals, Jacklynn Stott, Alexander Pritzel, Shakir Mohamed, and Peter Battaglia. Learning skillful medium-range global weather forecasting. *Science*, 382(6677):1416–1421, 2023.

[85] Ken Lang. NewsWeeder: Learning to Filter Netnews. In *Machine Learning Proceedings 1995*, pages 331–339. Morgan Kaufmann, San Francisco (CA), 1995.

[86] Rémi Lebret and Ronan Collobert. Word Embeddings through Hellinger PCA. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 482–490, Gothenburg, Sweden, April 2014. Association for Computational Linguistics.

[87] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in Neural Information Processing Systems*, 2, 1989.

[88] Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Lu, Thomas Mesnard, Colton Bishop, Victor Carbune, and Abhinav Rastogi. RLAIF: Scaling Reinforcement Learning from Human Feedback with AI Feedback. *arXiv preprint arXiv:2309.00267*, 2023.

[89] Sangkyu Lee, Sungdong Kim, Ashkan Yousefpour, Minjoon Seo, Kang Min Yoo, and Youngjae Yu. Aligning Large Language Models by On-Policy Self-Judgment. *arXiv preprint arXiv:2402.11253*, 2024.

[90] Sirpa Leppänen and Saija Peuronen. Multilingualism on the Internet. *The Routledge Handbook of Multilingualism*, pages 384–402, 2012.

[91] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of Massive Datasets*. Cambridge university press, 2020.

[92] Brian Lester, Rami Al-Rfou, and Noah Constant. The Power of Scale for Parameter-Efficient Prompt Tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3045–3059, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[93] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdel rahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Annual Meeting of the Association for Computational Linguistics*, 2019.

[94] Jiwei Li, Claire Cardie, and Sujian Li. TopicSpam: a Topic-Model based approach for spam detection. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 217–221, 2013.

[95] Junnan Li, Dongxu Li, Silvio Savarese, and Steven C. H. Hoi. BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models. In *International Conference on Machine Learning*, 2023.

[96] Kenneth Li, Aspen K Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Emergent world representations: Exploring a sequence model trained on a synthetic task. In *The Eleventh International Conference on Learning Representations*, 2023.

[97] Xiang Lisa Li and Percy Liang. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online, August 2021. Association for Computational Linguistics.

[98] Wing Lian and Bleys Goodson and Guan Wang and Eugene Pentland and Austin Cook and Chanvichet Vong and Teknium. MistralOrca: Mistral-7B Model Instruct-tuned on Filtered OpenOrcaV1 GPT-4 Dataset, 2023. URL https://huggingface.co/Open-Orca/Mistral-7B-OpenOrca. (Accessed 27-March-2024).

[99] S Likhitha, BS Harish, and HM Keerthi Kumar. A Detailed Survey on Topic Modeling for Document and Short Text Data. *International Journal of Computer Applications*, 178(39):1–9, 2019.

[100] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[101] Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. A Survey of Transformers. *AI Open*, 3:111–132, 2021.

[102] Lin Liu, Lin Tang, Wen Dong, Shaowen Yao, and Wei Zhou. An overview of topic modeling and its current applications in bioinformatics. *SpringerPlus*, 5(1):1–22, 2016.

[103] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the Middle: How Language Models Use Long Contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 02 2024.

[104] Vivian Liu and Lydia B Chilton. Design Guidelines for Prompt Engineering Text-to-Image Generative Models. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pages 1–23, 2022.

[105] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *ArXiv preprint arXiv::1907.11692*, 2019.

[106] Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An Empirical Study of Catastrophic Forgetting in Large Language Models During Continual Fine-tuning. *arXiv preprint arXiv:2308.08747*, 2023.

[107] Kai Lv, Hang Yan, Qipeng Guo, Haijun Lv, and Xipeng Qiu. AdaLomo: Low-memory Optimization with Adaptive Learning Rate. *arXiv preprint arXiv:2310.10195*, 2023.

[108] Kai Lv, Yuqing Yang, Tengxiao Liu, Qinghui Gao, Qipeng Guo, and Xipeng Qiu. Full Parameter Fine-tuning for Large Language Models with Limited Resources. *arXiv preprint arXiv:2306.09782*, 2023.

[109] Pratyush Maini, Skyler Seto, He Bai, David Grangier, Yizhe Zhang, and Navdeep Jaitly. Rephrasing the Web: A Recipe for Compute and Data-Efficient Language Modeling. *arXiv preprint arXiv:2401.16380*, 2024.

[110] Aditya Malte and Pratik Ratadiya. Evolution of Transfer Learning in Natural Language Processing. *arXiv preprint arXiv:1910.07370*, 2019.

[111] Michael McCloskey and Neal J. Cohen. Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. volume 24 of *Psychology of Learning and Motivation*, pages 109–165. Academic Press, 1989.

[112] Leland McInnes and John Healy. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *ArXiv*, abs/1802.03426, 2018.

[113] Leland McInnes, John Healy, and Steve Astels. hdbscan: Hierarchical density based clustering. *J. Open Source Softw.*, 2(11):205, 2017.

[114] Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. In *International Conference on Learning Representations*, 2013.

[115] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. In *Neural Information Processing Systems*, 2013.

[116] Christopher E. Moody. Mixing Dirichlet Topic Models and Word Embeddings to Make lda2vec. *CoRR*, abs/1605.02019, 2016.

[117] John X. Morris, Volodymyr Kuleshov, Vitaly Shmatikov, and Alexander M. Rush. Text Embeddings Reveal (Almost) As Much As Text. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

[118] Fabio Motoki, Valdemar Pinho Neto, and Victor Rodrigues. More human than human: Measuring ChatGPT political bias. *Public Choice*, pages 3—-23, 2023.

[119] Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, Stella Biderman, Teven Le Scao, M Saiful Bari, Sheng Shen, Zheng-Xin Yong, Hailey Schoelkopf, Xiangru Tang, Dragomir R. Radev, Alham Fikri Aji, Khalid Almubarak, Samuel Albanie, Zaid Alyafeai, Albert Webson, Edward Raff, and Colin Raffel. Crosslingual Generalization through Multitask Finetuning. In *Annual Meeting of the Association for Computational Linguistics*, 2023.

[120] Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. Orca: Progressive Learning from Complex Explanation Traces of GPT-4. *arXiv preprint arXiv:2306.02707*, 2023.

[121] Eric Nguyen. Chapter 4 - Text Mining and Network Analysis of Digital Libraries in R. In *Data Mining Applications with R*, pages 95–115. Academic Press, Boston, 2014.

[122] Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. Text Classification from Labeled and Unlabeled Documents using EM. *Machine Learning*, 39:103–134, 2000.

[123] OpenAI: Introducing ChatGPT, 2022. URL https://openai.com/blog/chatgpt. (Accessed 27-March-2024).

[124] OpenAI. OpenAI model page, 2024, URL https://platform.openai.com/docs/models/gpt-base. (Accessed 31-January-2024).

[125] OpenAI. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774*, 2023.

[126] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

[127] Oxford English Dictionary. About the OED, 2023. URL https://www.oed.com/information/about-the-oed/. (Accessed 03-April-2024).

[128] Divya Pandove, Shivan Goel, and Rinkl Rani. Systematic review of clustering high-dimensional and large datasets. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 12(2):1–68, 2018.

[129] Karl Pearson. LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[130] Alejandro Peña, Aythami Morales, Julian Fierrez, Ignacio Serna, Javier Ortega-Garcia, Íñigo Puente, Jorge Córdova, and Gonzalo Córdova. Leveraging Large Language Models for Topic Classification in the Domain of Public Affairs. In Mickael Coustaty and Alicia Fornés, editors, *Document Analysis and Recognition – ICDAR 2023 Workshops*, pages 20–33, Cham, 2023. Springer Nature Switzerland.

[131] Bo Peng, Eric Alcaide, Quentin Gregory Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Nguyen Chung, Leon Derczynski, Xingjian Du, Matteo Grella, Kranthi Kiran GV, Xuzheng He, Haowen Hou, Przemyslaw Kazienko, Jan Kocon, Jiaming Kong, Bartłomiej

Koptyra, Hayden Lau, Jiaju Lin, Krishna Sri Ipsit Mantri, Ferdinand Mom, Atsushi Saito, Guangyu Song, Xiangru Tang, Johan S. Wind, Stanisław Woźniak, Zhenyuan Zhang, Qinghua Zhou, Jian Zhu, and Rui-Jie Zhu. RWKV: Reinventing RNNs for the transformer era. In *The 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

[132] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[133] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[134] Xuan-Hieu Phan, Cam-Tu Nguyen, Dieu-Thu Le, Le-Minh Nguyen, Susumu Horiguchi, and Quang-Thuy Ha. A Hidden Topic-Based Framework toward Building Applications with Short Web Documents. *IEEE Transactions on Knowledge and Data Engineering*, 23(7):961–976, 2010.

[135] Xuan-Hieu Phan, Le-Minh Nguyen, and Susumu Horiguchi. Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *Proceedings of the 17th international conference on World Wide Web*, pages 91–100, 2008.

[136] Nirmalendu Prakash, Han Wang, Nguyen Khoi Hoang, Ming Shan Hee, and Roy Ka-Wei Lee. PromptMTopic: Unsupervised multimodal topic modeling of memes using large language models. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 621–631, 2023.

[137] Ofir Press, Noah Smith, and Mike Lewis. Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation. In *International Conference on Learning Representations*, 2022.

[138] Yifan Qiao, Chenyan Xiong, Zhenghao Liu, and Zhiyuan Liu. Understanding the Behaviors of BERT in Ranking. *arXiv preprint arXiv:1904.07531*, 2019.

[139] Zhangcheng Qiu and Hong Shen. User clustering in a dynamic social network topic model for short text streams. *Information Sciences*, 414:102–116, 2017.

[140] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving Language Understanding by Generative Pre-Training. 2018.

[141] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language Models are Unsupervised Multitask Learners. *OpenAI blog*, 1(8):9, 2019.

[142] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[143] Colin Raffel, Noam M. Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2019.

[144] Roger Ratcliff. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological Review*, 97(2):285, 1990.

[145] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. *Advances in Neural Information Processing Systems*, 30, 2017.

[146] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China, November 2019. Association for Computational Linguistics.

[147] Nils Reimers and Iryna Gurevych. Making Monolingual Sentence Embeddings Multilingual using Knowledge Distillation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 11 2020.

[148] Emil Rijcken, Floortje Scheepers, Kalliopi Zervanou, Marco Spruit, Pablo Mosteiro, and Uzay Kaymak. Towards Interpreting Topic Models with ChatGPT. In *The 20th World Congress of the International Fuzzy Systems Association*, 2023.

[149] Subhro Roy and Dan Roth. Solving General Arithmetic Word Problems. In Lluís Màrquez, Chris Callison-Burch, and Jian Su, editors, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[150] David Rozado. The Political Biases of ChatGPT. *Social Sciences*, 12(3):148, 2023.

[151] Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. Code Llama: Open Foundation Models for Code. *arXiv preprint arXiv:2308.12950*, 2023.

[152] Keita Saito, Akifumi Wachi, Koki Wataoka, and Youhei Akimoto. Verbosity Bias in Preference Labeling by Large Language Models. In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*, 2023.

[153] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear Component Analysis as a Kernel Eigenvalue Problem. *Neural Computation*, 10(5):1299–1319, 1998.

[154] Mike Schuster and Kaisuke Nakajima. Japanese and Korean voice search. In *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5149–5152. IEEE, 2012.

[155] Abigail See, Peter J. Liu, and Christopher D. Manning. Get To The Point: Summarization with Pointer-Generator Networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[156] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics.

[157] Wei Shen, Rui Zheng, Wenyu Zhan, Jun Zhao, Shihan Dou, Tao Gui, Qi Zhang, and Xuanjing Huang. Loose lips sink ships: Mitigating Length Bias in Reinforcement Learning from Human Feedback. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2859–2873, Singapore, December 2023. Association for Computational Linguistics.

[158] Yusuxke Shibata, Takuya Kida, Shuichi Fukamachi, Masayuki Takeda, Ayumi Shinohara, Takeshi Shinohara, and Setsuo Arikawa. Byte Pair Encoding: a Text Compression Scheme That Accelerates Pattern Matching. 1999.

[159] Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online, November 2020. Association for Computational Linguistics.

[160] Jonathon Shlens. A Tutorial on Principal Component Analysis. *ArXiv preprint ArXiv:1404.1100*, 2014.

[161] Robert R. Sokal and Charles Duncan Michener. A statistical method for evaluating systematic relationships. *University of Kansas science bulletin*, 38:1409–1438, 1958.

[162] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with Rotary Position Embedding. *Neurocomputing*, 568:127063, 2024.

[163] Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive Network: A Successor to Transformer for Large Language Models. *arXiv preprint arXiv:2307.08621*, 2023.

[164] Yee Teh, Michael Jordan, Matthew Beal, and David Blei. Sharing clusters among related groups: Hierarchical Dirichlet processes. *Advances in Neural Information Processing Systems*, 17, 2004.

[165] Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher D Manning. Just Ask for Calibration: Strategies for Eliciting Calibrated Confidence Scores from Language Models Fine-Tuned with Human Feedback. In *The 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

[166] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. LLaMA: Open and Efficient Foundation Language Models. *arXiv preprint arXiv:2302.13971*, 2023.

[167] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open Foundation and Fine-Tuned Chat Models. *arXiv preprint arXiv:2307.09288*, 2023.

[168] Lewis Tunstall, Leandro Von Werra, and Thomas Wolf. *Natural Language Processing with Transformers*. O'Reilly Media, Inc., 2022.

[169] Laurens van der Maaten and Geoffrey Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

[170] Tycho F. A. van der Ouderaa, Markus Nagel, Mart Van Baalen, and Tijmen Blankevoort. The LLM Surgeon. In *The Twelfth International Conference on Learning Representations*, 2024.

[171] Matti Varjokallio, Sami Virpioja, and Mikko Kurimo. Morphologically motivated word classes for very large vocabulary speech recognition of Finnish and Estonian. *Computer Speech & Language*, 66:101141, 2021.

[172] 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.

[173] Ike Vayansky and Sathish AP Kumar. A review of topic modeling methods. *Information Systems*, 94:101582, 2020.

[174] Michael Völske, Martin Potthast, Shahbaz Syed, and Benno Stein. TL;DR: Mining Reddit to learn automatic summarization. In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 59–63, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[175] Bailin Wang, Zi Wang, Xuezhi Wang, Yuan Cao, Rif A Saurous, and Yoon Kim. Grammar Prompting for Domain-Specific Language Generation with Large Language Models. *Advances in Neural Information Processing Systems*, 36, 2024.

[176] Han Wang, Nirmalendu Prakash, Nguyen Khoi Hoang, Ming Shan Hee, Usman Naseem, and Roy Ka-Wei Lee. Prompting Large Language Models for Topic Modeling. In *2023 IEEE International Conference on Big Data (BigData)*, pages 1236–1241, 2023.

[177] Peiyi Wang, Lei Li, Zhihong Shao, RX Xu, Damai Dai, Yifei Li, Deli Chen, Y Wu, and Zhifang Sui. Math-Shepherd: A Label-Free Step-by-Step Verifier for LLMs in Mathematical Reasoning. *arXiv preprint arXiv:2312.08935*, 2023.

[178] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-Instruct: Aligning Language Models with Self-Generated Instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508, Toronto, Canada, July 2023. Association for Computational Linguistics.

[179] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent Abilities of Large Language Models. *Transactions on Machine Learning Research*, 2022.

[180] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

[181] Gian Wiher, Clara Meister, and Ryan Cotterell. On decoding strategies for neural text generators. *Transactions of the Association for Computational Linguistics*, 10:997–1012, 2022.

[182] Wikimedia Foundation. Wikimedia Downloads. URL https://dumps.wikimedia.org. (Accessed 27-March-2024).

[183] Adina Williams, Nikita Nangia, and Samuel Bowman. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,*

Volume 1 (Long Papers), pages 1112–1122, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[184] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. HuggingFace's Transformers: State-of-the-Art Natural Language Processing. *arXiv preprint arXiv:1910.03771*, 2019.

[185] BigScience Workshop, Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, et al. BLOOM: A 176B-Parameter Open-Access Multilingual Language Model. *arXiv preprint arXiv:2211.05100*, 2022.

[186] Shengqiong Wu, Hao Fei, Leigang Qu, Wei Ji, and Tat-Seng Chua. NExT-GPT: Any-to-Any Multimodal LLM. *ArXiv*, abs/2309.05519, 2023.

[187] Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nyströmformer: A Nyström-Based Algorithm for Approximating Self-Attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 14138–14148, 2021.

[188] Hongfei Xu, Josef van Genabith, Qiuhui Liu, and Deyi Xiong. Probing Word Translations in the Transformer and Trading Decoder for Encoder Layers. In *North American Chapter of the Association for Computational Linguistics*, 2021.

[189] Weijie Xu, Wenxiang Hu, Fanyou Wu, and Srinivasan H. Sengamedu. DetiME: Diffusion-enhanced topic modeling using encoder-decoder based LLM. In *The 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

[190] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large Language Models as Optimizers. In *The Twelfth International Conference on Learning Representations*, 2024.

[191] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. *Advances in Neural Information Processing Systems*, 36, 2024.

[192] Chen Zhang, Dawei Song, Zheyu Ye, and Yan Gao. Towards the Law of Capacity Gap in Distilling Language Models. *ArXiv*, abs/2311.07052, 2023.

[193] Chen Zhang, Yang Yang, Jiahao Liu, Jingang Wang, Yunsen Xian, Benyou Wang, and Dawei Song. Lifting the Curse of Capacity Gap in Distilling Language Models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4535–4553, Toronto, Canada, July 2023. Association for Computational Linguistics.

[194] Jeffrey O. Zhang, Alexander Sax, Amir Zamir, Leonidas J. Guibas, and Jitendra Malik. Side-Tuning: A Baseline for Network Adaptation via Additive Side Networks. In *European Conference on Computer Vision*, 2019.

[195] Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. BERTScore: Evaluating Text Generation with BERT. In *International Conference on Learning Representations*, 2020.

[196] Yin Zhang, Min Chen, Dijiang Huang, Di Wu, and Yong Li. iDoctor: Personalized and professionalized medical recommendations based on hybrid matrix factorization. *Future Generation Computer Systems*, 66:30–35, 2017.

[197] Haiyan Zhao, Hanjie Chen, Fan Yang, Ninghao Liu, Huiqi Deng, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, and Mengnan Du. Explainability for Large Language Models: A Survey. *ACM Transactions on Intelligent Systems and Technology*, 15(2):1–38, 2024.

[198] Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, LILI YU, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. LIMA: Less Is More for Alignment. In *Advances in Neural Information Processing Systems*, volume 36, pages 55006–55021. Curran Associates, Inc., 2023.

[199] Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large Language Models are Human-Level Prompt Engineers. In *The Eleventh International Conference on Learning Representations*, 2023.

A Prompting templates

```
### HUMAN:
What is the generic topic of this document?
Document: [DOCUMENT_GOES_HERE]

### RESPONSE:
The generic topic of this document is "[TOPIC_GOES_HERE]"
```

Listing 2: The TLLM prompting template.

The TLLM prompting template was also used when prompting the foundation model. A prompting template utilizing ICL could have been more performant, but having a one-to-one comparison for the foundation model and TLLM was a prerequisite for quantifying the impact of fine-tuning.

```
[INST]
<<SYS>>
You are a topic modeling assistant. Given a document, you infer its
    general topic. Topics are always between 1-3 words long.
A topic never contains a brand, product, location, person, animal,
    event or vehicle. This is because the topic is something more
    generic.
<</SYS>>
What is the topic of this document?
Document: [DOCUMENT_GOES_HERE]
[/INST]
```

Listing 3: Prompt for generating topics using Llama-2-70b-chat.

```
root ::= (
    "The general topic of this document " ("is"|"appears to be") " \""
        + [a-z]+ " "? [a-z]+? " "? [a-z]+? " "? [a-z]+? " "? [a-z]+?
       "\"."
)
```

Listing 4: Response grammar for generating topics with Llama-2-70b-chat.

When generating topics with Llama-2-70B-chat, giving the LLM the option to use "appears to be" instead of "is" seemed to improve topic generation quality. The model would use the form of "appears to be" for documents where it was less confident, and "is" for the ones where the topic was more obvious. The response grammar allows the model to output up to five words. This modification was made in an effort to produce comprehensible topics even when the model was unable to do so with only three words. Without this modification, the model would opt to write its desired response

without whitespaces, combining the last words into one.

```
[INST]
<<SYS>>
You are a topic modeling assistant. Given a set of words, you infer
    their shared topic.
The topic is something that relates strongly to all of the given words
    .
Topics are always between 1-3 words long.
<</SYS>>
Example 1:
What is the shared topic of words "referee", "referee bias", "
    refereeing controversy"?
The shared topic of these words is "refereeing".

Example 2:
What is the shared topic of the words "infant sleep training", "
    insomnia", "sleeping"?
The shared topic of these words is "sleep".

Example 3:
What is the shared topic of the words "drone", "military drone", "
    drone security"?
The shared topic of these words is "drones".

Example 4:
What is the shared topic of the words "camp", "summer camp"?
The shared topic of these words is "camping".

Example 5:
What is the shared topic of the words "cannabidiol bill", "drug
    legalization", "medical cannabis", "quitting marijuana"?
The shared topic of these words is "drugs".

Example 6:
What is the shared topic of the words "drum and bass", "drums", "
    electric bass"?
The shared topic of these words is "music".

What is the shared topic of the words [CLUSTER_OF_TOPICS_GOES_HERE]?
[/INST]
```

Listing 5: Prompt for combining topics with llama-2-70b-chat. The example topics were chosen from known clusters of topics.

```
root ::= (
    "The topic of these words is \"" [a-z]+ " "? [a-z]+? " "? [a-z]+?
       "\"."
)
```

Listing 6: Response grammar for combining clusters of topics with Llama-2-70b-chat.

```
You are a topic modeling assistant. Given a document, you infer its
    general topic. Topics are always between 1-3 words long. A topic
    never contains a brand, product, location, person, animal, event
    or vehicle. This is because the topic is something more generic.
    Answer with "The generic topic of this document is ", followed by
    the topic.
```

Listing 7: GPT-3.5-turbo system prompt without ICL.

```
You are a topic modeling assistant. Given a document, you infer its
    general topic. Topics are always between 1-3 words long. A topic
    never contains a brand, product, location, person, animal, event
    or vehicle. This is because the topic is something more generic.
    Answer with "The generic topic of this document is ", followed by
    the topic.

Here are some examples.

Document: On December 13th last year, Parisians had to squint their
    eyes to be able to make out the iron girders of the normally
    imposing Eiffel Tower amid an unusually thick blanket of grey fog.
Topic: pollution

Document: The piña colada is a cocktail made with rum, cream of
    coconut or coconut milk, and pineapple juice, usually served
    either blended or shaken with ice
Topic: alcohol
```

Listing 8: GPT-3.5-turbo system prompt with ICL. The example documents and their topics are from the TLLM training dataset. The documents, originally from the OpenWebText [46] and Wikipedia datasets [182], have been truncated for brevity in the prompt.

```
What is the generic topic of this document?
Document: [Document]
```

Listing 9: GPT-3.5-turbo user prompt.

After prompting GPT-3.5-turbo, the results were lowercased, duplicate whitespaces were removed and any special characters, such as quotations and colons, were sanitized.