|  |  |  |
| --- | --- | --- |
|  |  | Research notes  NAZMUS SAMMO-103512692 |

Date: 3 September 2023

Why Data Pre-Processing:

The main component of a large language model is data. How good a LLM model will perform after the fine tuning of the model totally depends on the data you feed on the model. On the pre training process of a large language model, we feed the model billions of unstructured data, (when referring to data pre-processing for LLM pre training is where we take the data and transform into a format that algorithms can understand) to train the model about how to build a sentence, grammars, basically all the feature how human language work. So, after the machine got all the unstructured data, it uses the clustering unsupervised method to make a cluster of data using vector database. And when we try to interact with the model the model can’t really perform well. For example, for our project we have chosen the BERT base uncased model, which is 110M parameter model, when we tried to benchmark the pre trained BERT base model it performs awfully with our benchmarking data. So, when it comes to picture of fine tuning the BERT model on a certain domain, we need to feed the model with a really good structured dataset. Why dataset is so important? When we look at the industry leading LLMs we can see chatGPT and BARD ai, these models feed with well, labelled dataset. And then with the help of Supervised or RLHF method it got trained. Also if we look at some open sourced fine-tuned LLM like ALPACA or Vicuna where they took LLAMA model and fine tune with GPT conversation, it performs 90% similar to gpt. Now we can understand the importance of well structured dataset.

Data Pre-Preprocessing Techniques:

Data Cleaning:

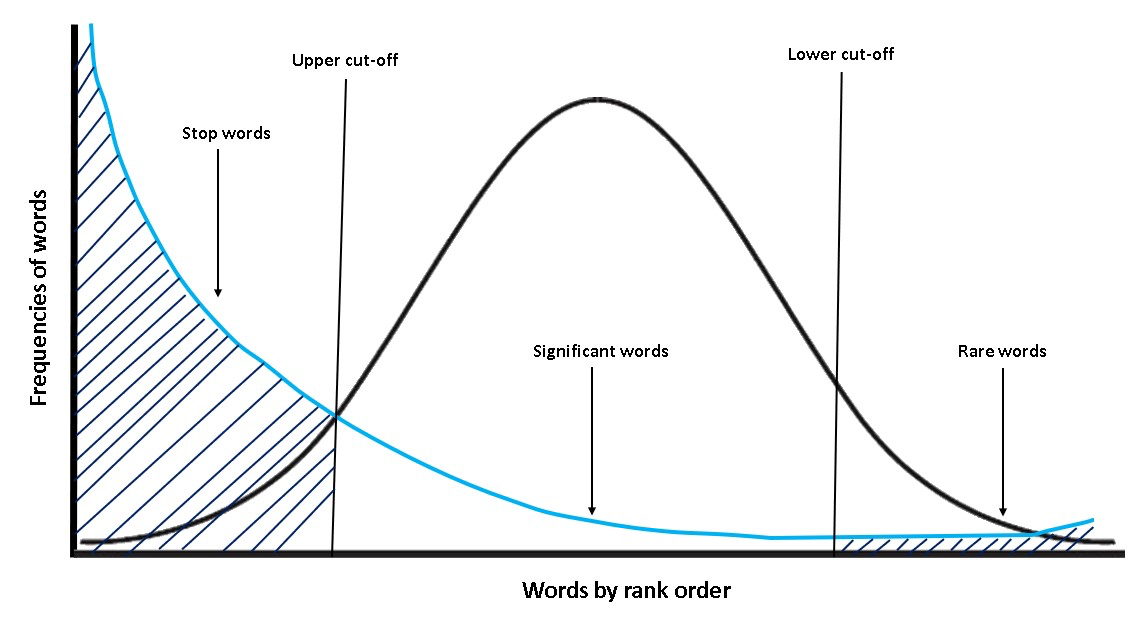
Data Cleaning is the first step we need to do after getting the data from the source. When we receive the data from the source most of the time, the data is really filled with empty rows or full of noise. So if we use the dataset without cleaning it, the model’s cost or loss function will be very high because the model will be trying to fit those noise in the training state. So to have increased overall highest quality information in the dataset we need to clean the dataset to train our model with the finest possible data. But for the project of this unit, we will be using cleaned data from renown source.

Tokenization:

If we feed a LLM constant stream of natural language data is not an optimal way for the model to learn the corpus during training. In fact, it probably would result in horrible performance with the model displaying signs of very low semantic understanding. Tokens are the basic unit of text that a LLM uses to process and generate new text. It can be characters, words, sub words or other segment of text depending on the type and the size of the model. Tokens are input into or produced from the model and given numerical values or IDs, as well as being organized into sequences or vectors. The foundational units of language for the model are tokens. Tokenization is the process of splitting the input and output texts into smaller units that can be processed by our model. By affecting the meaning and context of the tokens, tokenization can also have an impact on the quality and variety of the created texts. Depending on the complexity and diversity of the texts, tokenization can be done using a variety of techniques, including rule-based, statistical, or neural. The amount of data and the number of computations that the model must do are impacted by tokenization. The model uses more memory and processing resources as the number of tokens it must handle increases. The creation of a vocabulary that associates tokens with number representations is a crucial step in the tokenization process. Since LLMs operate on numerical inputs, each token in the lexicon is given a special identification number or index. This mapping enables fast computing and modelling by allowing LLMs to interpret and manage text data as numerical sequences.

Stop Words Removal:

There are mainly 3 types of words in a dataset and these are Stop words, Significant words and rarely occurring words. Stop words are the highly occurring words such as: a, an, the, is, was etc. Significant words are those words that have moderate frequency in the docs and add actual meaning to the text. Rarely occurring words used very rarely and lesser important than significant words. Now the question is why we should remove the stop words? Basically because they don’t provide any meaningful information. Also in a dataset the frequency of the stop words are way too much so if we remove the stop words from the dataset then the dataset becomes smaller than before also this provides much reduced size which results in faster computation on text data and the text classification model deal with a lesser number of features resulting in a robust model.



Lemmatization:

Lemmatization (or lemmatization) in linguistics is the process of grouping together the inflected forms of a word so they can be analyzed as a single item, identified by the word's lemma, or dictionary form. We use lemmatization to convert the input data set into a form that chops off the grammatical syntax to simplify the data set. This implies that multiple different words in the raw data set could map onto the same word after lemmatization. Consequently, this reduces noise and simplifies the training process alongside the obvious increase in the speed of analysis.

This approach works aptly under settings where the task is more closely associated with a contextual analysis of raw language. Popular applications are sentiment analysis and search querying, where the goal is to derive essence and return mappings from the root word. In other words, a good search algorithm would not discriminate and push down a result that mentions "cared"as opposed to a search query containing "care."

Stemming:

Stemming is a technique in Natural Language Processing used to extract the base form of words by removing affixes. It's a text normalization process commonly employed in information retrieval systems, including search engines, to ensure that different morphological variants of a word are treated as a single item. For instance, stemming would reduce the words “likes”, “liked”, “likely”, and “liking” to their common root, "like". While there are several algorithms for stemming, such as the Porter, Snowball, and Lancaster stemmers, each has its unique approach and efficiency. It's essential to note that stemming might not always produce valid words, and the outcome may differ from the word's lemma, its base form as listed in the dictionary. Stemming can sometimes introduce errors due to over-stemming, where words are reduced too much, or under-stemming, where words aren't reduced enough. Regardless, it remains a crucial component in many NLP tasks, helping improve system performance and data consistency.

Stemming vs Lemmatization:

Stemming and lemmatization are both text normalization techniques used in Natural Language Processing, but they serve different purposes and operate differently. Stemming involves chopping off prefixes and suffixes from a word to extract a base form, which might not always be a valid word in the language. For instance, "running" might be stemmed to "runn". On the other hand, lemmatization reduces words to their base or dictionary form, known as a lemma, taking into account the word's meaning and context. In lemmatization, "running" would typically be reduced to "run", a valid word. While stemming is generally faster due to its heuristic approach, lemmatization is more accurate and sophisticated as it considers the word's part of speech and sometimes even its surrounding context. Consequently, the choice between stemming and lemmatization often hinges on the specific needs of a project, weighing factors like processing speed against linguistic accuracy.

Advance Data Processing Techniques:

Advanced data preprocessing techniques are important in optimizing LLM models, there are many types of method researchers are working on. Among those:

Subword tokenization, with methods like Byte-Pair Encoding (BPE) and WordPiece, breaks text into meaningful subunits.

Data augmentation techniques, including back translation and noise injection, enhance NLP models by diversifying training data.

Pre-trained embeddings, such as Word2Vec and GloVe, streamline model initialization, while specialized embeddings cater to categorical data.

Dynamic Time Warping (DTW) address time-series variations, and feature engineering tools, from feature crosses to polynomial features, enrich data representation.

SMOTE, refining text data through TF-IDF and n-grams, and employing normalization strategies are all part of the preprocessing arsenal.

Moreover, dimensionality challenges are met with techniques like PCA and t-SNE, and missing data is addressed via sophisticated imputation methods.

Feature Engineering:

Fine-tuning large language models (LLMs) necessitates a deep understanding of data representation and preprocessing. LLMs, unlike traditional models, digest vast amounts of textual data and don't rely on manually crafted features. However, the way you present this data is paramount. For instance, structuring prompts effectively can steer the model's response in desired directions. Equally vital is the tokenization process, ensuring that the data's breakdown aligns with the model's inherent structure. Introducing task descriptors or prefixes can offer clearer context, guiding the model more precisely. Furthermore, managing the length of the input sequences is crucial, as long sequences might need to be truncated or shorter ones padded to fit the model's specifications. Additionally, data augmentation techniques, from back-translation to synonym replacements, can diversify training samples, aiding in robust fine-tuning. Thus, while LLMs eliminate the need for traditional feature engineering, they introduce a unique set of considerations to optimize their performance.

Reference:

1. Webz.io. (n.d.). Optimize LLM Data Preprocessing with Structured Historical Web Data. Available at: <https://webz.io/blog/machine-learning/optimize-llm-data-preprocessing-with-structured-historical-web-data/>
2. Microsoft. (n.d.). Prompt Engineering Tokens. Available at: <https://learn.microsoft.com/en-us/semantic-kernel/prompt-engineering/tokens>
3. Selin, L. (n.d.). Demystifying Tokens in LLMs: Understanding the Building Blocks. LinkedIn. Available at: <https://www.linkedin.com/pulse/demystifying-tokens-llms-understanding-building-blocks-lukas-selin/>
4. WisdomML. (n.d.). What are Stopwords in NLP and why we should remove them? Available at: <https://wisdomml.in/what-are-stopwords-in-nlp-and-why-we-should-remove-them/>
5. GeeksforGeeks. (n.d.). Introduction to Stemming. Available at: <https://www.geeksforgeeks.org/introduction-to-stemming/>
6. Analytics Vidhya. (2022). Stemming vs Lemmatization in NLP: Must know differences. Available at: <https://www.analyticsvidhya.com/blog/2022/06/stemming-vs-lemmatization-in-nlp-must-know-differences/>
7. Manning, C., Raghavan, P., & Schütze, H. (n.d.). Stemming and Lemmatization. Stanford NLP Group. Available at: <https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html>
8. KDnuggets. (2019). 4 Tips for Advanced Feature Engineering & Preprocessing. Available at: <https://www.kdnuggets.com/2019/08/4-tips-advanced-feature-engineering-preprocessing.html>
9. Labellerr. (n.d.). Data Collection and Preprocessing for Large Language Models. Available at: <https://www.labellerr.com/blog/data-collection-and-preprocessing-for-large-language-models/>
10. Appy Pie. (n.d.). Datasets and Data Preprocessing for LLM Training. Available at: <https://www.appypie.com/blog/datasets-and-data-preprocessing-for-llm-training>