* Word Embedding
* Word Classification
* Language Modeling
* Transfer learning
* Attention
* LLMs
  + Berts
  + GPTs
  + Chat gpt and instruct gpt

## Word Embeddings

The method of representation of words in vector space. There are 4 ways of representing a word in vector space, 1. Discrete, distributional statistics, Statically learned like Glove, Skip Gram and contextual embeddings like one learned from bert, elmo and gpts.

### Discrete representation

One hot encoder : have ith vector as 1, not an ideal representation for the words as the vectors have no meaning, no way to capture context.

### Distributional Static

What is meaning or context

1. Words which frequently appear in similar contexts have similar meaning
2. How do we add context
3. Count based - where we count the occurance in a context matrix
4. Positive Pointwise Mutual Information - colorance but using distributional-similarity
5. Latent Semantic Analysis - context using tf-idf for a document

### Static learned representation

Static learned representations for embeddings refer to fixed, pre-trained word representations that do not take into account the contextual information of the surrounding words. Unlike contextual embeddings, which capture context-dependent meanings, static embeddings assign a single vector representation to each word in isolation.

Word2Vec and GloVe are examples of popular methods used to learn static embeddings. These methods generate word vectors by analyzing large corpora of text data. Word2Vec uses a shallow neural network to predict the probability of a word given its context or vice versa. GloVe, on the other hand, constructs a co-occurrence matrix based on word co-occurrence statistics and factorizes it to obtain word vectors.

The resulting word vectors from these methods represent the learned semantic and syntactic properties of the words. They capture similarities and relationships between words based on their usage patterns in the training data. These static embeddings can be useful in various NLP tasks, such as word similarity, text classification, and sentiment analysis.

One advantage of static embeddings is their efficiency and simplicity. Once the word vectors are trained, they can be directly used in downstream tasks without the need for additional computations. Additionally, they can be easily applied to out-of-vocabulary (OOV) words by assigning them a special vector or by using simple heuristics.

However, static embeddings have limitations when it comes to capturing the fine-grained nuances and context-specific meanings of words. Since they treat each word as an independent entity, they may not adequately capture the variability in word semantics across different contexts. This can lead to suboptimal performance in tasks that require a deeper understanding of the context.

Overall, static learned representations for embeddings provide a useful baseline for various NLP tasks but may not capture the full contextual information necessary for more advanced language understanding. Contextual embeddings, such as those generated by models like BERT or GPT, have emerged as a powerful approach to overcome these limitations by capturing contextual nuances and providing more accurate word representations.

Glove or Skipgram

Skpigram works better with subword

### Contextual Embeddings

Contextual embeddings are a type of word representation used in natural language processing (NLP) that capture the meaning and contextual information of words or phrases within a given text. Traditional word embeddings, such as word2vec or GloVe, assign a fixed vector representation to each word in isolation, without considering the surrounding context.

In contrast, contextual embeddings take into account the context in which a word appears, enabling them to capture the nuances and variations in meaning that can arise from different contexts. These embeddings are typically generated by pre-trained models, such as BERT (Bidirectional Encoder Representations from Transformers) or GPT (Generative Pre-trained Transformer), which are trained on large corpora of text data.

Contextual embedding models utilize a deep neural network architecture that employs a transformer-based architecture. These models process text in a bidirectional manner, meaning they consider both the preceding and following words when generating an embedding for a particular word. By considering the wider context, these models can better capture the semantic relationships and dependencies between words, leading to more accurate representations.

The training process for contextual embedding models involves exposing the model to a vast amount of text data and training it to predict missing words within sentences or to perform other language-related tasks. This process helps the model learn to understand the contextual meaning of words and encode that information into the embeddings it produces.

Once trained, contextual embedding models can be used for a variety of NLP tasks, such as text classification, named entity recognition, sentiment analysis, machine translation, and question answering. By incorporating contextual embeddings into these tasks, models can leverage the rich contextual information encoded within the embeddings, resulting in improved performance and a better understanding of the underlying text.

In summary, contextual embeddings provide a representation of words or phrases that considers the surrounding context, allowing NLP models to capture the subtle variations in meaning that arise from different contexts. These embeddings are generated by pre-trained models, such as BERT or GPT, and can be used to enhance a wide range of NLP tasks.

<https://chat.openai.com/share/647af471-e79c-4076-a3be-4ebebc519cda>

Isotropic

Elmo, Bert, GPT-2

Anistropropic

### References

1. <https://lena-voita.github.io/nlp_course/word_embeddings.htm>
2. <https://explosion.ai/blog/applied-nlp-thinking>
3. [How Contextual are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings](https://aclanthology.org/D19-1006.pdf)
4. <https://fasttext.cc/docs/en/unsupervised-tutorial.html>
5. <https://jaketae.github.io/study/word2vec/>
6. <https://web.stanford.edu/~jurafsky/slp3/slides/6_Vector_Apr18_2021.pdf>
7. https://web.stanford.edu/~jurafsky/slp3/6.pdf

## Language Models

## Attention/ Self Attention / Multihead attention

## 

## 

## LLMs

### Motivation

Language modeling is usually framed as unsupervised distribution estimation from a set of examples (x1, x2, ..., xn) each composed of variable length sequences of symbols

(s1, s2, ..., sn). Learning to perform a single task can be expressed in a probabilistic framework as estimating a conditional distribution p(output|input). Since a general system should be able to perform many different tasks, even for the same input, it should condition not only on the input but also on the task to be performed. That is, it should model p(output|input, task).

### Training Data

Web can be generally unintelligible, hence how can that we used for training. Reddit hyperlink is used for this purpose. Any hyperlink with 3 outward + sign is used as proxy for good link.

Input Representation

1. UTF-8 byte level representation doesn’t come to picture as they are not competitive with word level
2. include pre-processing steps such as lowercasing, tokenization, and out-of-vocabulary tokens which restrict the space
3. Byte pair encoding : frequent words as a single tokens, while less frequent words are represented by multiple tokens, each of them representing a word part.[7] For example, the word "transformers" would be represented by two tokens, one encoding a frequent word "transform" as its first subword and the other encoding the "ers" as another frequent subword. BPE brings the perfect balance between character- and word-level hybrid representations which makes it capable of managing large corpora. This behavior also enables the encoding of any rare words in the vocabulary with appropriate subword tokens without introducing any “unknown” tokens. This especially applies to foreign languages like German where the presence of many compound words can make it hard to learn a rich vocabulary otherwise. With this tokenization algorithm, every word can now overcome their fear of being forgotten (athazagoraphobia)
4. Unigram tokenization:

### Architecture

Transformer long sentensteces

In brief, the five common LLM challenges include:

1. Understanding model limitations
   1. Model Bias
   2. Build
2. Choosing your model’s endpoint
3. Finetuning the model to your task
4. Choosing the right set of parameters
5. Designing prompt for the model
6. Training and deployment

<https://txt.cohere.com/best-practices-for-deploying-language-models/>

<https://txt.cohere.com/5-challenges-of-working-with-large-language-models-and-how-to-cope-with-them/>

<https://txt.cohere.com/how-to-train-your-pet-llm-prompt-engineering/>