WORD EMBEDDING SPACES

BENJAMIN MATTHIAS RUPPIK

Abstract. Word embedding spaces are a powerful tool for natural language processing (NLP), as they allow us to represent words as real-valued vectors that capture their semantic and syntactic properties. These ideas fit into the more general framework of *Representation Learning*, a way to automatically learn features from data. In this seminar, we will explore different methods for creating and using word embedding spaces, from static models like word2vec and fastText, to contextual models like ELMo, BERT, RoBERTa, and GPT. We will also analyse the geometry and topology of word embedding spaces, and how they relate to linguistic phenomena such as analogy, similarity and compositionality. Finally, we will discuss some applications of word embeddings in task-oriented dialogue systems, such as intent detection, slot filling and response generation. We will also introduce some current aligned language models such as InstructGPT and ChatGPT that leverage word embeddings for generating natural and engaging conversations.

This document contains extended notes for the Seminar "Word Embedding Spaces", Heinrich-Heine-University Düsseldorf, Summer Term 2023.

Organization

- **Summer Term 2023 at HHU:** "Master's Seminar on Word Embedding Spaces"
- Wednesday; 12:30 – 14:00; from 05.04.2023 to 12.07.2023; Location: Gebäude 25.22 - 2522.U1.72
- Schedule on Google Sheets
- Lecture course repository: `https://github.com/ben300694/word-embeddings`
- Live stream on WebEx will probably be available for registered participants.
- **Requirements for receiving credit points:**
  - Active participation in the seminar sessions.
    - Active participation in the seminar sessions means that students are expected to attend all the sessions, read the assigned materials in advance, and engage in constructive discussions with their peers and the instructor.
    - Active participation also means that students should ask relevant questions, share their opinions and insights, and provide constructive feedback to their classmates' presentations.
  - Delivery of one scientific presentation.
    - The topics will be assigned by me at the beginning of the semester. Students should contact me as soon as possible to choose their preferred topic from the available list.
    - Before giving their presentation in class, students must schedule a meeting with me to show a draft of their slides and receive feedback and suggestions for improvement.
    - The presentations should be clear, concise and well-structured. They should include an introduction, a literature review, a main argument, and a conclusion. They should also cite relevant sources and use appropriate visual aids.
  - Submission of a LaTeX-file and compiled PDF of an extended abstract:
    - No more than 2 pages.
    - Using the ACL Overleaf template.
    - Including relevant references to scientific literature.
- **Formal Prerequisites:** None

Key words and phrases. Natural Language Processing, Word embeddings, Dialog Systems, Topological Data Analysis.
Date: 2023-03-09.

- **Recommended Prerequisites:** Background in deep learning, linear algebra.

Policy on AI writing assistants. AI writing assistants, such as chatbots or language models like ChatGPT [Ope23] and GitHub Copilot, may be used as a tool to enhance writing skills and productivity. However, it is important to note that the final product submitted must be original work and properly cited if using any direct quotes or paraphrased material. Plagiarism, including use of AI writing assistance without proper citation, will not be tolerated and is a violation of academic honesty.

Please have a look thoughts collected on the website Educator considerations for ChatGPT.

1. Introduction

In a *word embedding* we encode language by associating to a word w a point or vector $v_w \in \mathbb{R}^n$ in some ambient space, such that words which have similar meaning are close together in the space. When words are modelled as vectors or points in an ambient high-dimensional space, this is called an *embedding*.

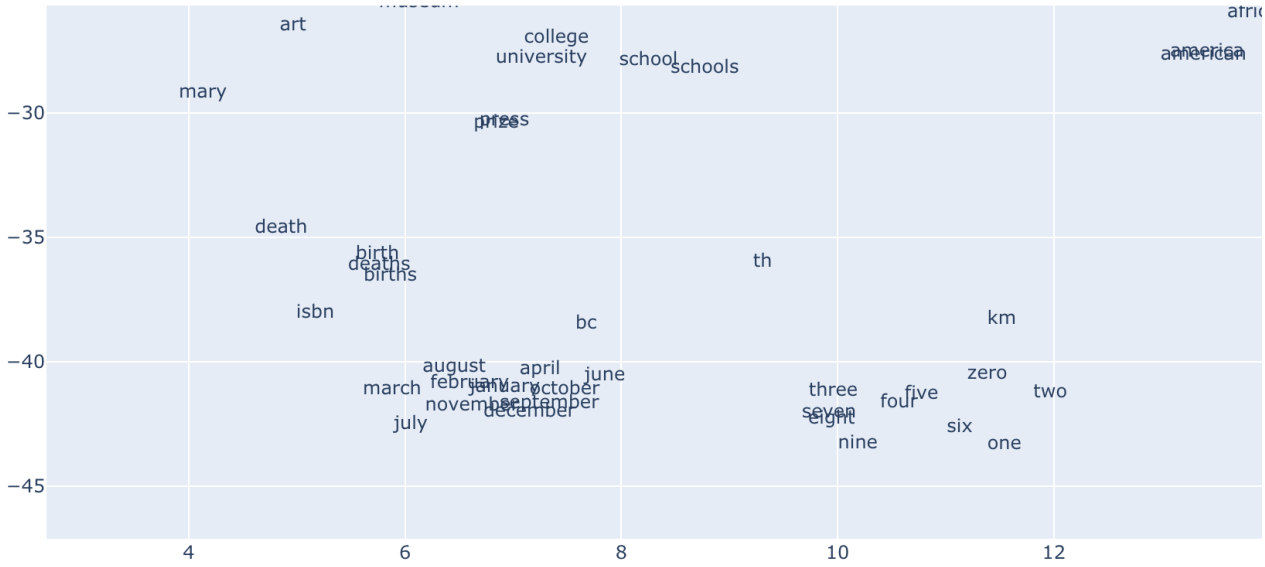


Figure 1. Some of the word vectors in a 100 dimensional fastText embedding trained on a Wikipedia corpus; projected to 2 dimensions using t-SNE.

Words which are used in similar contexts usually have similar meanings.[1] The *distributional hypothesis* states that words which frequently appear in similar contexts have similar meanings. There is *first-order co-occurence*, also called *syntagmatic association*, if the words are typically nearby each other; for example: 'wrote' and 'book', 'wrote' and 'poem'. We also observe *second order co-occurence*, called *paradigmatic association*, if the words have similar neighbors; for example: 'wrote' and 'said', 'wrote' and 'remarked'.

This makes it possible to acquire meaningful representations from unlabeled data – and there is lots of unlabeled data, e.g., in the form of text from Wikipedia, books, or news headlines. Thus, we can make both *distributed* and *distributional representations*: Distributional means that it is based on counts of words in context; distributed means that the representation is vector based.

The task of finding good word embeddings becomes hard because there are many subtleties in the usage of words in natural language. Words (or *lemmas*) like 'mole' can have multiple meanings, which are its *word senses*; a word which has multiple meanings is *polysemous*. The embedding should also be able to capture concepts such as *synonyms* (words have the same meaning in many or all contexts) and *antonyms* (the word senses are opposites with respect to one feature of the meaning), *similarity*, *relatedness*, *connotation* or *sentiment*. There is usually a *super-subordinate* relation (also called *hyperonymy*, *hyponymy* or "is-a" relation). On the other hand, *stop words* like 'a', 'the', 'is', 'are', 'and' are so commonly used that they carry very little useful information.

There is a big difference between *static* and *contextualized* embeddings: In a *static embedding* (Section 2) there is one fixed embedding for each word in the vocabulary. In a *contextual embedding* (Section 3) the vector we associate to a word is different if it is used in a different context.

[1]This is not always true, as the words 'good' and 'bad' demonstrate, more on this below.

We will also talk about how one can systematically evaluate the quality of a word embedding. In *extrinsic evaluation*, we use a word embedding for a specific downstream task and measure whether it increases performance. *Intrinsic evaluation* is a harder challenge.

1.1. General References and Links.

- For a general introduction see the book [JM09], but take a look at the draft of the 3rd edition, in particular Chapter 6 on "Vector Semantics and Embeddings". There are corresponding slides as well.
- [Eis19, Chp. 14] on "Distributional and Distributed Semantics"; draft available here.
- Michael Heck's slides on the Space of Word Embeddings.
- Lena Voita's NLP Course For You, in particular the lesson on Word Embeddings and the lesson on Sequence to Sequence (seq2seq) and Attention.
- Rasa Algorithm Whiteboard YouTube playlist.
- AI Coffee Break with Letitia YouTube channel.

1.1.1. *Acknowledgments.* The content of this document is based on the seminar "Word Embedding Spaces" run at Heinrich-Heine-University Düsseldorf in the Summer Term 2022 and 2023. I am grateful for the suggestions and feedback collected during the sessions. Thank you to Milica Gašić, Chris Geishauser, Michael Heck for very helpful additional input.
Parts of the current notes are based on the final reports written by the following students:

- Nick Rucks: fastText and GloVe
- Lukas Rücker: Sequence to Sequence and Recurrent Methods
- Benedikt Prusas, Benedikt Jung: Transformers
- Thanh Nam Le, Michail Angelos Severino Theoktistou: Masked Language Models and BERT

A small portion of these learning materials was drafted with the help of AI writing assistants such as ChatGPT, GitHub Copilot and Microsoft Bing Search Chat Mode. The majority of the content was written by human experts and all content has been manually edited and fact checked for accuracy.

2. Background and Static Word Embeddings

2.1. Frequency based methods. Frequency based word embedding methods are one of the simplest and oldest approaches to word embedding. They are based on determining the frequencies of words in documents or sentences, and using them to create vector representations for words. Frequency based methods can be divided into three main types: count vectors, TF-IDF, and co-occurrence matrices. These methods have some advantages, such as being easy to understand and implement, but also some disadvantages, such as being sparse, high-dimensional, and ignoring the semantic context of words.

Here are a few key-words and terminology that you should know about:

- Co-occurence counts;
- Term-document matrix: each document is represented by a vector of word counts;
- **T**erm **F**requency – **I**nverse **D**ocument **F**requency (TF-IDF): sparse vectors, words are represented as a simple function of the counts of neighbors;
- **P**ointwise **P**ositive **M**utual **I**nformation (PPMI): $\mathrm{PMI}(w_1, w_2) = \log \frac{p(w_1,w_2)}{p(w_1)\cdot p(w_2)}$;
- Dense versus sparse embeddings;
- Latent semantic analysis: singular value decomposition applied to term-document matrix (weighted by log frequency and normalized by entropy);
- Cosine similarity.

References:

- General references in Section 1.1;
- Using "tricks" from word2vec to improve count-based models [LGD15];
- Skip-gram with negative-sampling (SGNS, see word2vec below) implicitly factorizes the shifted pointwise mutual information (PMI) matrix [LG14].

2.2. **word2vec.** Word2vec is a more recent and advanced technique for word embedding that uses a shallow neural network to learn word associations from a large corpus of text. Unlike frequency based methods, word2vec can capture the semantic context of words by placing them in a lower-dimensional vector space. Word2vec can also detect synonymous words or suggest additional words for a partial sentence. Word2vec consists of two main models: skip-gram and CBOW, which use different ways of predicting words based on their contexts. Word2vec also uses two optimization techniques: hierarchical softmax and negative sampling, which speed up the training process and improve the quality of the embeddings.

References:

- The original sources are [Mik+13a], [Mik+13b].
- The survey [Ron14] has a detailed explanation of the parameter update equations in word2vec.
- [AH19] try to find a theoretical explanation for these linear relations.
- Similarities across languages can be detected in the embeddings [MLS13], which allows constructions of mappings between semantic spaces even without having parallel data [Con+17].
- Such an analysis can lead to ideas for improving the embeddings, as in [MBV17] (they subtract the mean from the word vectors and eliminate top PCA components).
- General Language Understanding Evaluation (GLUE) benchmark, SuperGLUE

static embedding; **s**kip-**g**ram with **n**egative **s**ampling (SGNS); Idea: train a classifier to a prediction task 'the word w is likely to show up near the word c'; self-supervision: can use the next word in a corpus of running text as the supervision signal

- get positive examples from taking a target word w and its neighboring context words (have to choose a context window size L, these notes [Gol15] discuss the effect of the window size).
- randomly sample other words to get negative examples (based on the empirical distribution of words, but usually modified to sample less frequent words more often).
- train a classifier (for example by logistic regression, i.e. the probability score is given as the sigmoid of the dot product) to distinguish the two cases; skip-gram makes simplifying assumption that all context words are independent.
- the learned weights of the classifier give the embedding of the word w: target embedding in matrix W and context embedding in matrix C.

Variations of word2vec training methods:

- Skip-Gram: predict context words given the central word.
- **C**ontinuous **B**ag-**o**f-**W**ords (CBOW): predict the central word from the sum of context words (this sum is called the "bag of words").
- Simple, but powerful approach: Represent a sentence with the average of its word embeddings. Hybrid: word embeddings are computed distributionally, and the sentence embedding computed by composition.

Evaluating word embeddings: *Intrinsic evaluation* measures how well the word vectors capture meaning, for example by evaluating on word similarity and word analogy tasks; *extrinsic evaluation* is performed by looking at the performance of a downstream task. Linear structure in the embedding space can be observed from word analogies, such as that the vector of

$$v_{\text{king}} - v_{\text{man}} + v_{\text{woman}} \text{ is close to } v_{\text{queen}}.$$

2.3. **GloVe.** GloVe [PSM14] stands for "global vectors", which refers to the fact that the model captures global corpus statistics directly. This is similar to most matrix factorization methods but GloVe additionally provides dense vector representations and has a linear substructure in the vector space. Therefore, it combines the advantages of global factorization methods and local context window methods. On the other hand, it cannot handle OOV words. However, by efficiently training on huge corpora, it can at least decrease the occurrence probability for OOV words.

Starting point for learning the GloVe embeddings are the ratios of co-occurrence probabilities

$$\frac{P_{ik}}{P_{jk}}, \tag{2.1}$$

where P_{ik} is the probability that word k appears in the context of word i. Further the probability is defined by

$$P_{ij} = \frac{X_{ij}}{\sum_k X_{ik}}, \tag{2.2}$$

where X_{ij} is the frequency that word j occurs in the context of word i while $\sum_k X_{ik}$ is the sum of co-occurrence frequencies of all words k in the context of word i. The ratios from Equation (2) are able to discriminate relevant from irrelevant words in the corpus. Hence from these ratios, the authors derive a weighted least squares objective to learn the GloVe embeddings, i.e., they optimize

$$\mathcal{J} = \sum_{i,j=1}^{V} f(X_{ij})(w_i^\top \widetilde{w}_j + b_i + \widetilde{b}_j - \log X_{ij})^2 \tag{2.3}$$

where w_i and $\widetilde{w}_j$ are the randomly initialized vector representations for the target word i and context word j while b_i and $\widetilde{b}_j$ are bias terms that are learned together with the embeddings. The loss imposes that the inner product of the word vectors $w_i^\top \widetilde{w}_j$ should be approximately equal to their log co-occurrence counts $\log X_{ij}$. Hence, GloVe can be viewed as a matrix factorization method. The weighting function f penalizes irrelevant word pairs, either if they are too rare or too frequent. During optimization of $\mathcal{J}$, the authors sample non-zero entries X_{ij} from the co-occurrence matrix X. This matrix tabulates how often words co-occur with each other and can be obtained by a single pass through the whole corpus.

The GloVe embeddings are successfully evaluated on the word analogy task, validating that the vector space contains linear sub-structures. Moreover, it is demonstrated that GloVe embeddings achieve better results on the word similarity task than embeddings from Word2Vec. In general, GloVe is able to consistently outperform Word2Vec in a similar training setting. Moreover, since it can leverage bigger amounts of data it further provides a better representation for rare words than Word2Vec. Finally, during the extrinsic evaluation the performance on a Named entity recognition (NER) task could be improved using GloVe embeddings as features compared to a CBOW embeddings from Word2Vec.

static embedding; captures global corpus statistics (so it considers global context), based on ratios of probabilities from word-word co-occurrence matrix X_{ij}; these counts are used to define the loss function, which leads to minimization of a sum of squares

$$J = \sum_{i,j} \frac{X_{ij}}{X_{\max}} (w_i \widetilde{w_j}^T - \log X_{ij})^2$$

w_i is the word vector and $\widetilde{w_j}$ the context vector; can add in a weighting function to penalize rare events and not over-weight frequent events, and add in bias terms for each weight vector; does not handle out of vocabulary (OOV) words;

References:

- Original source [PSM14], GloVe project page.

2.4. **fastText.** FastText [Boj+16] is an extension of the Skip-gram model introduced by [Mik+13b] and therefore belongs to the family of local context window methods. In contrast to Skip-gram and Continuous Bag-of-Words (CBOW) [Mik+13b] it is able to embed out-of-vocabulary (OOV) words. Furthermore, by considering character level information it improves the representation for rare and compound words which leads to an increased performance, especially for morphologically rich languages like German or Finnish.

FastText computes a bag of constituent n-grams $\mathcal{G}_w$ for each word w in the corpus, containing the word itself plus all n-grams for a predefined range of n. Then, the word representation $\mathbf{u}_w$ is the sum of n-gram embeddings $\mathbf{z}_g$ obtained by a pre-trained Skip-gram model, i.e.,

$$\mathbf{u}_w = \sum_{g \in \mathcal{G}_w} \mathbf{z}_g. \tag{2.4}$$

Besides the different representation $\mathbf{u}_w$, the training of the objective is adopted from the Word2Vec framework.

The authors show in an intrinsic evaluation that the inclusion of character level information enables an improved performance on the word similarity task compared to Word2Vec. This also holds for the syntactic questions of the word analogy task. Moreover, the representations that FastText computes for OOV words always improves the performance on the word similarity task which validates that the representations are reasonable. During further evaluation, the authors showed that the most important n-grams in the representation actually correspond to valid morphemes, e.g., prefixes and suffixes. For that, the authors omitted a single n-gram from the representation $\mathbf{u}_w$ to get a restricted representation $\mathbf{u}_{w \setminus g}$. Then, they ranked all n-grams by their cosine distance between $\mathbf{u}_w$ and $\mathbf{u}_{w \setminus g}$, to determine the importance of each n-gram for $\mathbf{u}_w$. Furthermore, the better usage of subword information enables FastText to obtain better embeddings from smaller datasets than Word2Vec. Additionally, the word representation trained with subword information outperforms plain Skip-gram embeddings on a language modeling task.

static embedding; deals better with unknown words and word sparsity; add special boundary symbols < and > to the word, example for $n = 3$ and the word 'where': <where> and <wh, whe, her, ere, re>; learn skip-gram embedding for each constituent n-gram, word is represented by the sum of all the embeddings of the constituent n-grams;

References:

- Original source [Boj+16]

3. Contextual Word Embeddings

3.1. Sequence to sequence tasks and Recurrent networks.

3.1.1. *Motivation.* Modeling language using static embeddings like word2vec or Glove, there are a few problems. Different meanings of the same word cannot be accurately captured, as each word has only a single vector embedding. Thus, depending on the training corpus, the embedding will either only capture a single meaning, or the embedding will be the average of different meanings. Furthermore, these embeddings allow for simple arithmetic operation that capture some meaning, but these lack the flexibility to capture the interaction between words that are so essential to language. To tackle more complex tasks like translation we need to rethink language modelling to integrate a source input. One way to do this is by using an Encoder-Decoder architecture [Voi].

3.1.2. *Encoder and Decoder.* The idea of the Encoder-Decoder architecture, is that the Encoder takes an input and produces a latent representation. The Decoder uses this representation to (re-)create information. More specifically, using translation as an example (throughout this abstract) it looks like this: The Decoder uses the given representation of a source sentence to create a translation. For translations recurrent neural networks (RNNs) have proven to be effective implementation for this.

3.1.3. *Recurrent Neural Networks.* A RNN is a neural network that is getting repeated throughout time, and at each time step t there can be a new input and output. The crux because RNNs are eligible for NLP task, is that the hidden layer at time step $t-1$ influences the hidden layer at time step t. Thus, the problem of lacking the ability to capture language dependencies between words is being tackled. A RNN can then be used both as the Encoder and as the Decoder. RNNs in their basic form, can run into problems during model training because of vanishing or exploding gradients [SVL14]. Furthermore, the

information propagation is not ideal for longer sequences and performance suffers. To address this a commonly used modification is using a Long Short-Term Memory (LSTM) RNN.

3.1.4. *Long Short-Term Memory.* A LSTM is a different kind of RNN, where the internal structure of each reoccurring unit is different. Besides the hidden state, there is an additional cell state. This cell state gets passed through each unit, with little interference. It helps to keep the gradient in appropriate ranges as well as transfer information through many units [Kar]. With these LSTMs Peters et al. designed a model named Embeddings from Language Models (ELMo), to tackle the problem of finding embeddings for polysemic words.

3.1.5. *Embeddings from Language Models (ELMo).* ELMo produces a different embedding for each word dependent on the sentence it is used in. This is implemented as a multi-layered bidirectional LSTM. Using an LSTM for each direction ensures to capture information of the entire input sequence. The model then combines the hidden state representation of both LSTMs by concatenating them, and them adds up the concatenated states of each layer up with a task-specific weighting [Pet+18a]. The embeddings created by ELMo have the potential to improve the language understanding of many NLP models. Still, using a single vector representation for an entire source sentence is bound to fail at capturing all information. To reduce this dependency Bahdanau et al. added the concept of *attention*.

3.1.6. *Attention.* The idea behind attention is that the model learns to take in more specific information from the single source inputs. This can be implemented with a context vector. A context vector is the result of calculating attentions scores between a single hidden state of the decoder and all encoder states. Using a softmax function, these attention scores can be turned into probabilities or weights. Summing up each encoder state weighted by its softmax value, we calculate the context vector [BCB15]. The decoder at each time step uses this context and is not solely dependent on the single representation from the Encoder. If implemented with an appropriate attention score, the whole process is differentiable and can be learned task-specific by the model.

Embeddings from **L**anguage **Mo**dels (ELMo): The ELMo architecture is based on earlier ideas using Recurrent Neural Nets (RNNs) to model sequences, such as sentences in natural language. ELMo produces deep contextualized word representations, which are learned functions of internal states of a deep bidirectional language model with Long short-term memory (LSTM) units. ELMo embeddings are context-sensitive, i.e. each token representation is a function of the entire input sentence. Here lower layers appear to capture syntactics, while higher layers capture semantics. ELMo is unsupervised, can handle OOV, but is not truly bidirectional.

References:

- Original ELMo research article: 'Deep Contextualized Word Representations' [Pet+18b].
- The concept of attention, which lets a model focus on different parts of the input while processing a sequence, was introduced in [Gar+19].
- Lecture notes: Sequence to Sequence (seq2seq) and Attention

3.2. **Transformers.** The authors in [Vas+17] introduce a neural network architecture for the task of machine translation without any recurrence. This resolves the problems of recurrent neural networks such as weak long range dependencies, training instability and scale-ability.

We can view a transformer as a "sequence to sequence" model and differentiate between stacked encoder and decoder blocks.

Embeddings get contextualized throughout multiple encoder blocks. The information flows through the attention mechanism and is processed by basic feed forward layers.

Ultimately, the blocks can be broken down to matrix operations and the attention formulation is derived from a database analogy [SIS21].

References:

- Original research article introducing the transformer architecture: 'Attention is all you need' [Vas+17].

- Blog post: 'Visualizing A Neural Machine Translation Model (Mechanics of Seq2seq Models With Attention)'
- Blog post: 'The Illustrated Transformer'
- Blog post: 'Transformers from scratch'
- PyTorch implementation 'The Annotated Transformer'
- Vision transformers [Dos+21]
- Visualizations of attention maps: BertViz

Encoder and decoder block; self-attention mechanism; query, key and value matrices; multi-head attention (combining multiple self-attention mechanisms); positional embeddings; autoregressive models: predict the next token in a sequence, for this we have to mask the self-attention in the decoder block so that the model can only attend to tokens in the preceding sequence. The original transformer model was trained on machine translation, which is a sequence to sequence task.

3.3. **Masked Language Models and BERT.**

3.3.1. *Introduction.* BERT (**B**i-directional **E**ncoder **R**epresentations from **T**ransformers) follows the Transformer architecture [Vas+17]; more concretely the encoder part of it. BERT allows tokens to attend both previous and posterior tokens. This is the main advantage setting BERT apart from previous language representation models. Two BERT versions are trained, BERT-Base to be compared with the Open AI's GPT model and BERT-Large with more blocks, attention heads, and a larger hidden dimension.

3.3.2. *Model Input.* As input, BERT accepts either single or a pair of sentences, which are simply concatenated with a special token [SEP] in between. To enable down-stream tasks, there is also a classification token [CLS] at the beginning. To further distinguish the tokens belonging to each sentence, a third segment embedding is used, besides the usual token and positional embeddings known from the standard transformer. Tokenization is generated via WordPiece [**wu2016google**], which uses a byte-pair encoding scheme. WordPiece is composed of a 30k token vocabulary, each word is searched in this corpus and if it does not exist it will be decomposed into constituent pieces. This input embedding is static at first but will become contextualized after going through the model.

3.3.3. *BERT Architecture.* BERT uses a multi-layer bidirectional transformer encoder. This encoder is based on the original transformer encoder. Each BERT layer contains two normalization layers, a feed forward layer and a multi-head attention layer. The major difference is that BERT uses bidirectional self-attention, words can see them selves, there is no constrain, while the GPT transformer uses constrained self-attention where every token can only attend to the context to its left. BERT comes in two sizes: BERT Base has 12 layers, a hidden size of 768, 12 self-attention heads and 110M parameters to train. BERT Base has the same model size as GPT so their performances are comparable. BERT Large has 24 layers, a hidden size of 1024, 16 self-attention heads and a total of 340M to train. BERT Large is to achieve better results on benchmarks stated in the paper.

3.3.4. *Pre-training and Fine-tuning.* BERT is pre-trained using BooksCorpus and the English Wikipedia, via two unsupervised tasks as follows:

(1) Masked Language Model (MLM). To avoid mappings between input-output due to a bidirectional context where words can see themselves, 15 percent of the tokens are randomly masked. The masked tokens are then predicted, as the final hidden vectors corresponding to them are fed into a softmax over the vocabulary. To mitigate a mismatch between pre-training and fine-tuning as there is no [MASK] token during fine-tuning, only 80 percent of the chosen tokens are replaced by [MASK]; 10 percent are replaced by a random token, and the remaining 10 percent are left unchanged, as illustrated below:
   - my cat is hungry $\rightsquigarrow$ my cat is [MASK]
   - my cat is hungry $\rightsquigarrow$ my cat is president
   - my cat is hungry $\rightsquigarrow$ my cat is hungry

(2) Next-Sentence-Prediction (NSP). We want to predict if sentence B is the actual sentence that proceeds sentence A or not. This task is beneficial for downstream tasks that require understanding the relationship between sentences, such as question-answering. To learn this relationship between sentences, a binary classification task is used where sentence B is 50 percent of the time the one that follows sentence A (label = *IsNext*), and 50 percent of the time is chosen randomly (label = *NotIsNext*). Here are two examples.
CLS The man went to [MASK] store [SEP] He bought a gallon [MASK] milk [SEP] $\rightsquigarrow$ *IsNext*
CLS The man went to [MASK] store [SEP] Penguins [MASK] flightless birds [SEP] $\rightsquigarrow$ *NotIsNext*

Fine-tuning is done by initializing the model with the pre-trained weights to adapt for specific tasks. During this process the pre-trained weights change slightly and the tasks specific weights are learned. In the paper, BERT is fine-tuned for 12 different NLP tasks. The BERT Large model is shown to be the best-performing system across the multi-task GLUE benchmark, the SQuAD benchmark (both version 1.1 and 2.0) as well as the SWAG benchmark.

3.3.5. *Conclusion.* The authors show that Transformers continue to be state-of-the-art in many NLP tasks. They exhibit the benefits of using a true bidirectional structure based on a MLM, which allows to solve more downstream tasks. They also show how BERT can be trained relative faster and in a more efficient way by using the self-attention mechanism from transformers that use parallel training instead of traditional LSTMs that use sequential training.

References:

- Original research article: [Dev+18]
- (Introduction to) Transfer Learning in the NLP for you course.
- [Cla+19] analyze the role of the different attention heads in the trained BERT model.
- A **Ro**bustly **O**ptimized **BERT** Pretraining **A**pproach (RoBERTa): The authors observe that BERT was significantly under-trained, and update the model [Liu+19].
- GPT for language generation.

The objective of the BERT model is a *Masked language modeling task*, in which we randomly mask words in the training text and try to predict the masked words from their context. Additionally, the model is trained on a *next sentence prediction* task. For the input embedding: WordPiece tokenization; special `[CLS]` token for the representation of the entire input sequence.

3.4. **Sentence Embeddings.** *Contrastive learning* is a technique that aims to learn representations of data that are invariant to different views of the same data. In other words, contrastive learning tries to learn features or patterns from the data that do not change much when the data is transformed in some way. For example, if you have an image of a cat and you rotate it or crop it or change its color, the image still shows a cat. Contrastive learning wants to learn embeddings that capture this fact and make the embeddings of different views of the same cat image similar to each other. It does so by maximizing the similarity between embeddings of different views of the same data (positive pairs) and minimizing the similarity between embeddings of different data (negative pairs). Contrastive learning can also be applied for constructing *sentence embeddings*, which are vector representations of natural language sentences that capture their semantic meaning.

Constructing sentence embeddings is hard because sentences are complex linguistic units that convey various aspects of meaning, such as syntax, semantics, pragmatics and discourse. A good sentence embedding should be able to capture all these aspects and reflect the similarity or difference between sentences in a consistent and interpretable way.

As a baseline, you could think of a bag of words approach with a static word embedding as a sentence encoder. This does not work well in practice, because it ignores the order and structure of words in a sentence. It also treats each word as an independent unit and does not account for the context or the polysemy of words. For example, consider the following two sentences:

- He saw her duck.
- She saw his duck.

A bag of words approach with a static word embedding would assign similar embeddings to these two sentences because they have very similar words. However, these sentences have different meanings depending on how we interpret the word "duck". It could be a noun (an animal) or a verb (an action). A good sentence embedding should be able to distinguish between these meanings and assign different embeddings to these sentences.

References:

- Sentence-BERT: [RG19] and SentenceTransformers find an embedding on the sentence level so that semantically similar sentences are close in the embeddings space; adds a pooling operation to the output of BERT or RoBERTa to derive a fixed sized sentence embedding; siamese and triplet network structures.
- TransEncoder: For pairwise comparison of sentences, [Liu+21] alternate between a bi-encoder (which produces fixed-dimensional sentence embeddings) and a cross-encoder (which has attention heads that span the complete sequence of both sentences), so that they learn from each other to train an unsupervised sentence representation model.
- SimCSE [GYC21]: Uses contrastive learning to improve sentence embeddings, where the positive pairs are created by adding dropout in the encoder as the noise.
- Dialog utterance embeddings via contrastive learning [Zho+22]
- Video: Transformer Neural Networks - an Overview!

3.5. **Multilingual word embeddings.** Most word embeddings we have discussed are *monolingual*, meaning that they only represent words from one language. This limits their applicability for cross-lingual tasks such as machine translation, cross-lingual information retrieval, or multilingual text analysis.

Multilingual word embedding spaces aim to overcome this limitation by learning embeddings for words from multiple languages in a common vector space. This way, words that have similar meanings across languages are close together in the vector space, regardless of their linguistic differences. Multilingual word embedding spaces enable us to leverage linguistic resources from different languages and transfer knowledge across them. The hope is that this can improve performance on various downstream tasks because the model can acquire and transfer knowledge from other languages.

References:

- MUSE is an implementation of both supervised and unsupervised multilingual word embeddings with fastText
- [CC18] train unsupervised MWEs;
- [Fen+20] build language-agnostic BERT Sentence Embedding
- Video: Language Agnostic BERT
- Please look for further references on multilingual transformer architectures.
- [Con+17] aligned embeddings without having parallel data.
- [Jaw+19] jointly learn multiple languages in a common latent space.

3.6. **Multimodal Embeddings.** *Multimodal machine learning* deals with the combination and analysis of different types of data, such as text, speech, images, videos, etc. It aims to create systems that can understand and generate multimodal content, and perform tasks that require cross-modal reasoning and communication.

A *multimodal embedding space* is a high-dimensional space where different types of data, such as text, images, audio, etc., can be represented as points or vectors. The goal of multimodal embedding is to map data from different modalities to a common space where they can be compared and related based on their semantic similarity. For example, a multimodal embedding space can be used to find images that match a given text description or vice versa. These models leverage the power of natural language to provide a flexible and expressive way to interact with visual data.

Text-to-image models are a type of generative models that can create realistic images from natural language descriptions. One example of such models is DALL-E, which was developed by OpenAI. DALL-E is a simple decoder-only transformer that receives both the text and the image as a single stream of tokens and models all of them autoregressively. It can generate diverse and creative images for

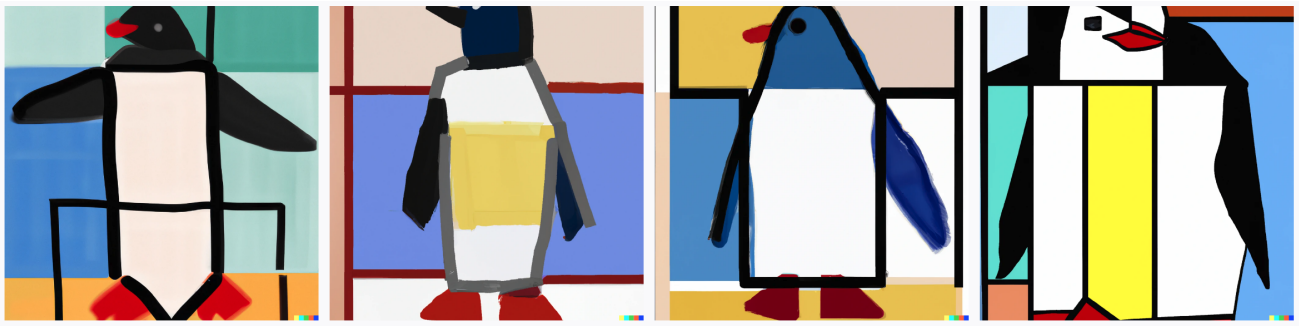


FIGURE 2. DALL-E 2 generations for the prompt *Painting of a penguin in the style of Piet Mondrian.* Piet Mondrian (1872-1944) was a Dutch artist pioneering abstract art, his later paintings were reduced to simple geometric objects.

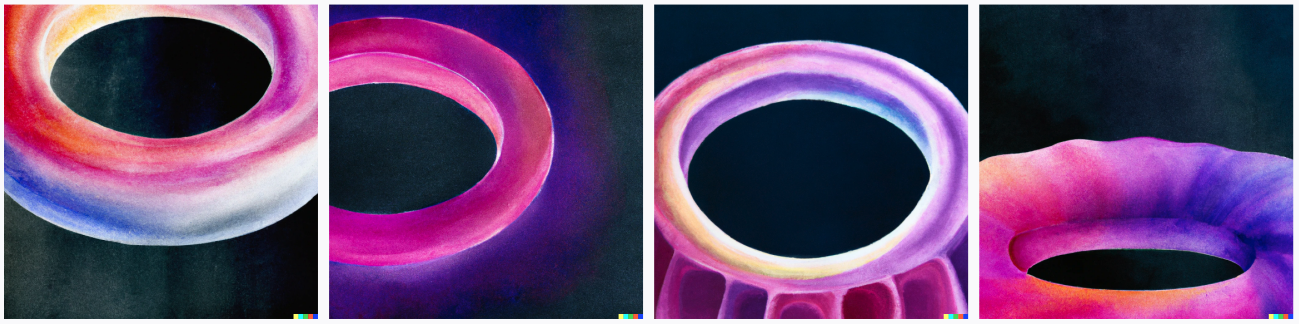


FIGURE 3. DALL-E 2 generations for the prompt *A watercolour drawing of a torus surface in 3-dimensional space on a dark background.*

a wide range of text prompts, such as "an armchair in the shape of an avocado" or "a store front that has the word openai written on it". For examples of image generations with its sucessor, the diffusion model DALL-E 2, see Figures 2 and 3.

References:

- Joint text and image space: [Zha+21]
- CLIP: Connecting Text and Images; [Rad+21]; CLIP (Contrastive Language-Image Pre-training) is a multimodal machine learning model that learns a joint space for words and images by training on a large-scale dataset of image-text pairs. CLIP can perform zero-shot image classification by comparing natural language descriptions with image features.
- Text-to-image model DALL-E 2 [Ram+22]

4. Structure in Word Embeddings

4.1. Sentiment specific word embeddings.

TODO Add abstract on sentiment in word embeddings

References:

- Using sentiment aware word embeddings for emotion classification [Mao+19];
- TweetEval dataset, which includes emotion recognition and sentiment analysis tasks [Bar+20], the authors released their pre-trained RoBERTa models;
- German sentiment classification with using fastText and BERT embeddings [Guh+20];
- EmoWOZ [Fen+21] is a task-oriented dialogue dataset annotated with emotions.

Sentiment is a positive or negative evaluation, and we can enhance our *semantic* specific word embeddings with *sentiment* [Yu+17]. Words with similar vector representations can have an opposite sentiment polarity (e.g., 'good' vs 'bad' or 'happy' vs 'sad'). To capture this in the embedding one can add a post-processing step to adapt the pre-trained vectors to sentiment applications, where we move

the word closer to a set of both semantically and sentimentally similar nearest neighbors (i.e., those with the same polarity) and further away from sentimentally dissimilar neighbors (i.e., those with an opposite polarity).

4.2. **Bias in Word Embeddings.** TODO Add abstract for bias in word embedding spaces

A desirable association could be "a man to a woman is as a king to a queen", whereas an undesirable association is "a man to a woman is as a physician to a nurse". Unfortunately, it appears that word embeddings trained on a corpus of text extracted from books and websites tend to incorporate and amplify the biases contained in our language. When such biased embeddings are used for downstream tasks, this can lead to bias amplification. To investigate gender bias systematically, [Bol+16] take a seed pair $(v_{\text{he}}, v_{\text{she}})$ and find other pairs of word vectors which have a difference similar to the seed vector. They also provide methods for debiasing the embedding, but they argue that bias ultimately appears to come from the training data.

References:

- Iterative nullspace projection to debias word embeddings [Rav+20];
- Visualizations of bias mitigation techniques [Rat+21], their code for creating interactive demos is available online;
- Gender bias in the contextualized embeddings from ElMo [Zha+19];
- Bias in BERT embeddings [Kur+19];
- Debiasing contextualized embeddings [KB21].

5. Word Embeddings in Dialogue Systems

5.1. **Applications of Word Embeddings in Task-oriented Dialogue Systems.** A *task-oriented dialogue system* is a computer system that can communicate with a human via natural language to help them complete a specific task, such as booking a ticket, ordering food or scheduling a call. Task-oriented dialogue systems are different from chatbots that aim to have open-ended conversations with humans for entertainment or social purposes. Task-oriented dialogue systems are more focused on achieving a clear goal and providing relevant information to the user.

A task-oriented dialogue system typically consists of four main components:

(1) *Natural Language Understanding (NLU)*: This component is responsible for analyzing the user's input and extracting relevant information, such as intents (what the user wants to do) and entities (what the user refers to). For example, if the user says "I want to book a flight to Paris for next week", the NLU component should identify the intent as booking a flight and the entities as "Paris" and "next week". Word embeddings improve the natural language understanding component by enabling it to handle synonyms, paraphrases, out-of-vocabulary words and cross-lingual inputs.

(2) *Dialogue State Tracking (DST):* This component is responsible for keeping track of the current state of the dialogue, such as what information has been provided by the user and what information is still missing. For example, if the user has specified their destination but not their departure date, the DST component should store this information and update it accordingly. Word embeddings enhance the dialogue state tracking component by allowing it to compare user inputs and system outputs based on their semantic similarity

(3) *Policy:* This component is responsible for deciding what action to take next based on the current state of the dialogue. For example, if some information is missing from the user's request, the policy component should decide to ask a clarifying question. If all the information is available, the policy component should decide to confirm or execute the request. Word embeddings facilitate the policy component by enabling it to learn from diverse data sources and generalize to unseen scenarios.

(4) *Natural Language Generation (NLG):* This component is responsible for generating an appropriate response in natural language based on the action decided by the policy component. For example, if the policy component decides to ask for more information, then NLG should generate a

question like “When do you want to depart?”. If the policy component decides to confirm or execute the request, then NLG should generate a response like “OK, I have booked your flight to Paris for next week.”. Word embeddings boost the natural language generation component by allowing it to produce fluent and diverse responses that match the user’s style and preferences.

Nowadays, word embeddings are essential for building effective and robust task-oriented dialogue systems that can handle complex and dynamic user requests across different languages and domains.

TODO Add references

5.2. **Aligning Language Models.** The *GPT model family* is a family of language models that use deep learning techniques to generate natural language text. They are built using several decoder blocks of the transformer architecture, which enables them to learn from large amounts of text data and produce coherent and diverse texts in an autoregressive manner.

Aligning language models means training them to follow instructions given by humans, such as generating summaries, answering questions, or writing stories. This can be done by using reinforcement learning from human feedback (RLHF), a method that rewards the model for producing outputs that match the user’s preferences Alternatively, it can be done by self-instructing, a framework that generates instruction, input, and output samples from a language model itself and uses them to finetune the original model.

InstructGPT and ChatGPT are two examples of aligned language models that can follow instructions given by users. InstructGPT can generate texts for various tasks such as summarization, translation, or sentiment analysis. ChatGPT can generate engaging and natural conversations for different domains such as gaming, sports, or movies.

- InstructGPT [Ouy+22]
- Sparrow
- ChatGPT

TODO Add references

- *Zero-shot learning* is a problem setup in machine learning where, at test time, a learner observes samples from classes which were not observed during training, and needs to predict the class that they belong to. For example, a zero-shot model can classify an image of a zebra without having seen any images of zebras during training.
- *Few-shot learning* is a problem setup in machine learning where a learner needs to learn a new task from a very small number of labeled examples (typically less than 10 per class). For example, a few-shot learning model can recognize handwritten characters from different alphabets after seeing only one or few examples of each character.
- *In-context learning (ICL)* is a problem setup in natural language understanding where a large pre-trained language model observes a test instance and a few training examples as its input, and directly decodes the output without any update to its parameters. For example, an in-context learning model can perform sentiment analysis by conditioning on some input-output pairs that demonstrate the task.

TODO Add references and examples

6. Topological Data Analysis and Geometric Deep Learning

6.1. **Geometry of the word embedding space.** [MT17] find that in embeddings trained via skip-gram with negative sampling, the word vectors all point in roughly the same direction, i.e. they lie in a narrow cone. This distribution is sometimes called *anisotropy*. The opposite of anisotropy is *isotropy* where the vectors are directionally uniform. The context vectors of the skip-gram model point in the other direction (which means that they have negative dot product with the mean of the word vectors). In contrast to this, for GloVe the word and context vectors point in the same direction.

Method for finding words which are used differently in two text corpora: train embedding on each corpus, find a mapping between the embedding spaces that produces an alignment (e.g. linearly via

orthogonal Procrustes). Then words which do not match well under this mapping appear to have different meanings. Using historical texts, one can see how the meaning changes through time [HLJ16].

Different method: again train embeddings on two corpora separately, then for each word find the closest vectors in each embedding space, if these differ a lot the embeddings are different [Gon+20].

References:

- Anisotropy can also be studied in contextualized word embeddings, see [Eth19], where Ethayarajh defines measures of isotropy and compares those between the embedding models ELMo, BERT and GPT-2.
- [Cai+21] argue while contextual embeddings might appear anisotropic on first sight, they contain clusters and manifolds that reflect isotropy.
- [RP21] present clustering based approaches for mitigating the anisotropy in contextual word embedding spaces.
- For isotropy in sentence embeeddings see [Li+20]
- [NF18] observe that mappings between semantic spaces usually are only locally linear, and not globally linear;

6.2. **Embeddings in Curved Geometries.** Many real-world data sets have complex structures that cannot be adequately captured by Euclidean vector spaces. For example, images of faces can be seen as points on a high-dimensional manifold that is curved and nonlinear. To learn meaningful representations of such data, we need to take into account the *intrinsic geometry* of the underlying *manifold. Riemannian manifold learning (RML)* is a framework that aims to do this by using concepts and tools from Riemannian geometry. RML assumes that the input high-dimensional data lie on an intrinsically low-dimensional Riemannian manifold, and seeks to construct coordinate charts that preserve the geodesic distances and local neighborhoods of the data points. By doing so, RML can reveal the latent structure of the data and enable various applications such as dimensionality reduction, clustering, classification, visualization and more.

Important keyword in this context include:

- *Manifolds*: A manifold is a topological space that is locally Euclidean, meaning that around every point, there is a neighborhood that looks like a flat space of some dimension. For example, a sphere is a two-dimensional manifold because any small patch on its surface can be flattened into a plane. Manifolds can be used to model complex data sets that have some underlying structure or smoothness.
- *Geodesics*: A geodesic is the shortest path between two points on a curved surface or manifold. For example, on a sphere, the geodesics are the great circles. Geodesics can be used to measure distances and angles on manifolds, and to define notions of curvature and parallel transport.
- *Riemannian optimization*: Riemannian optimization is a branch of optimization that deals with finding optimal solutions on manifolds. It generalizes classical optimization methods such as gradient descent and Newton's method by taking into account the geometry of the manifold. Riemannian optimization can be used to solve problems such as matrix factorization, low-rank approximation, principal component analysis and more on manifolds.

TODO Add references for Riemannian manifold learning

6.2.1. *Positively Curved Spaces: Spherical Embeddings.* Euclidean embeddings might impose some limitations such as linearity and orthogonality. Spherical Word Embeddings [Men+19] are an alternative approach that learns word vectors on a unit sphere, where the angle between two vectors reflects their similarity. Spherical word embeddings can overcome some of the drawbacks of Euclidean word embeddings, such as being more flexible and expressive, being able to model antonyms and polysemous words better, and being more robust to noise and outliers. Spherical word embeddings can also be extended to learn paragraph or document embeddings on a sphere, which can improve various downstream tasks such as document clustering and classification.

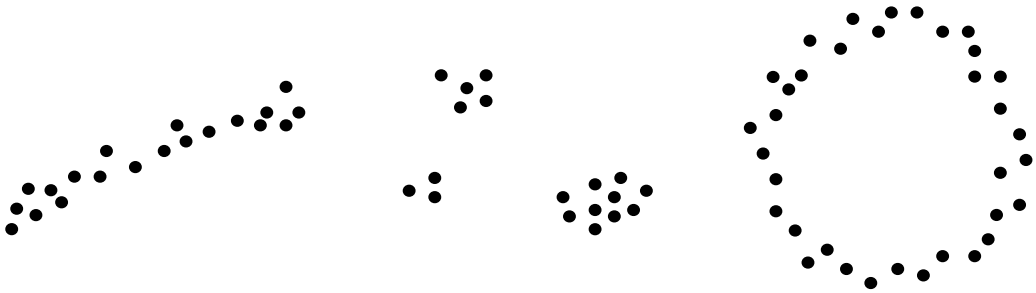


FIGURE 4. Collection of various shapes of point clouds.

6.2.2. *Negatively Curved Spaces: Hyperbolic Embeddings.* We can embed in so-called *negatively curved spaces*, which are also known as *hyperbolic embeddings*. The geometry of these spaces allows a more efficient representation of tree-like structures and hierarchies, as was observed in [NK17]. See also this blog post discussing the paper, and this blog post with implementation details. In our case, where we associate words to points in a hyperbolic space, these are called *hyperbolic word embeddings* (which can be constructed e.g. with Poincaré GloVe [TBG18]). Further references include an earlier paper on hyperbolic text embeddings [Dhi+18], these in turn were inspired by hyperbolic image embeddings [Khr+19]. There is also a proposed hyperbolic version of fastText [Zhu+20]. One can probe the contextualized embeddings of a BERT model by mapping into a hyperbolic space [Che+21], and analyzing the structure of the projected data.

6.3. **Topological Data Analysis.** TODO Edit

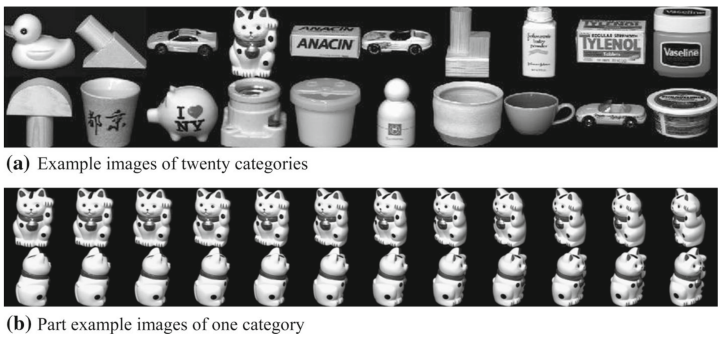
Representation learning: Find a compact and dense representation of the data that captures its underlying structure, patterns, and relationships $\rightsquigarrow$ **Point cloud** $X \subset \mathbb{R}^N$.

Topological Data Analysis [CV22]: Mathematical framework for understanding and simplifying complex data structures by capturing their essential *shape*. Figure 4

Regression: Identify the relationship between a dependent variable and one or multiple independent variables. Linear *dimensionality reduction techniques* such as PCA.

Clustering: Identify groups of similar data points. For instance, *Single Linkage Clustering* groups data points based on the minimum distance between two points in a cluster.

Periodic data: Cyclic patterns can appear in time series (daily patterns of human behavior, seasonal fluctuations in the stock market, annual changes in weather patterns, ...). Expressing the data in terms of a circular coordinate, it becomes easier to see patterns in the data that may not be immediately apparent otherwise. See Figures 5 and 6



(a) Example images of twenty categories

(b) Part example images of one category

FIGURE 5. COIL-20 dataset (128 × 128 grayscale images, twenty objects with 72 poses each)

FIGURE 6. 2-dimensional *t*-SNE projection

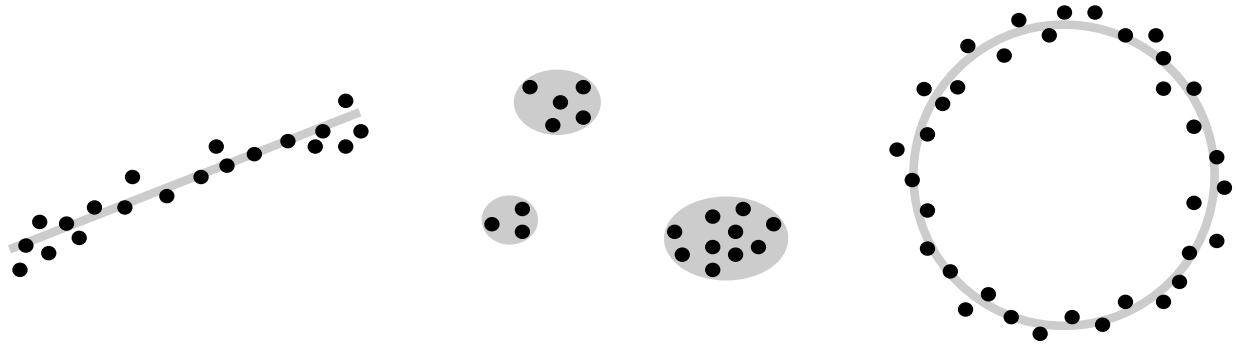


FIGURE 7. Point clouds with approximating shapes.

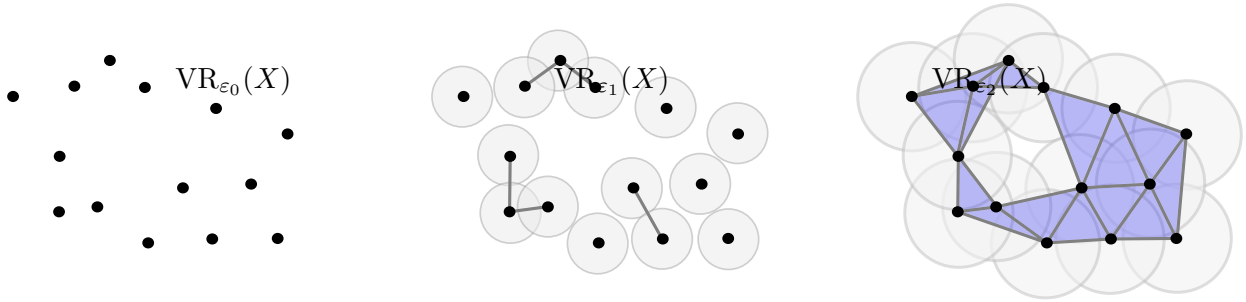


FIGURE 8. Vietoris-Rips complex with various size scales TODO Fix labels

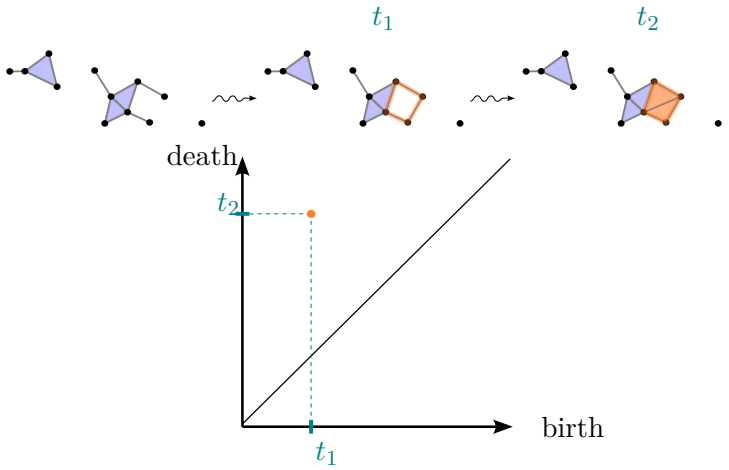


FIGURE 9. Filtered complex with persistence diagram. TODO Fix labels

Topological Data Analysis is particularly useful in the analysis of high-dimensional data, where traditional regression methods may struggle because *the best approximating shape might not be known a priory.* Figure 7

6.3.1. *Persistent Homology. Vietoris-Rips complex:* Figure 8 Combinatorial description of the space via a *simplicial complex*

Idea of Persistent Homology: Capture the evolution of topological features over *different size scales* ε_i, to obtain insight into the intrinsic geometry of the data.

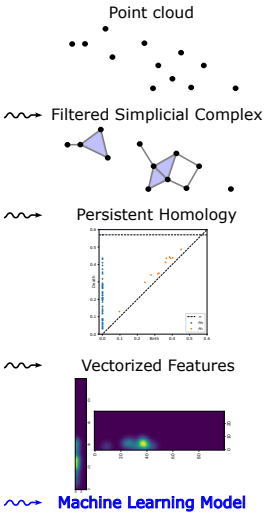


FIGURE 10. Topological Deep Learning Pipeline.

Persistence diagrams: Figure 9 Visual representations of the topological features and their *lifespan* in a filtered simplicial complex.

Topological Deep Learning: Figure 10 Incorporate topological structures and insights into deep learning algorithms to improve their ability to capture and analyze complex relationships between data points.

- Understanding of geometric and topological properties of data
- Insights into inner working of deep learning models

6.3.2. *References for TDA.* There have been two main developments in Topological Data Analysis (TDA), which might lead to further interesting application of TDA to word embeddings in the future:

- *Persistent homology* is a description of the topological features of a space at various scales. Usually the data comes with a *filtration*, and we compute the homology of the pieces which assemble into a *persistence module*. These can be encoded into a *bar-code*, which then is used in further machine learning pipelines. For example [RCB21] develop a Transformer model which takes persistence diagrams as input.
- The MAPPER algorithm is a projection technique which also uses clustering to capture more of the global features of the dataset. An interesting projection choice is t-Stochastic Neighborhood Embedding (t-SNE, [MH08]).

Here are some general references for background reading on Topological Data Analysis:

- [MR19]: Survey paper with an introduction to "Topological Data Analysis for Physicists".
- [DW21]: Book (draft) on Topological Data Analysis.
- Vidit Nanda's Computational Algebraic Topology Notes.

6.4. **Manifold Hypothesis and Singularities.** The *Data Manifold Hypothesis* states that high-dimensional data lies on or near a low-dimensional structure called a manifold.[2] Figure 11

- **Consequence:** Most variation in the data can be captured by a small number of dimensions
- Important in dimensionality reduction techniques such as PCA, t-SNE, UMAP
  $\rightsquigarrow$ Helps explain why these techniques often produce meaningful results

Situations where the data **does not** have a low-dimensional structure and cannot be effectively mapped onto a lower-dimensional space:

- Noise

[2]Manifold: Locally Euclidean topological space (plus some extra conditions)

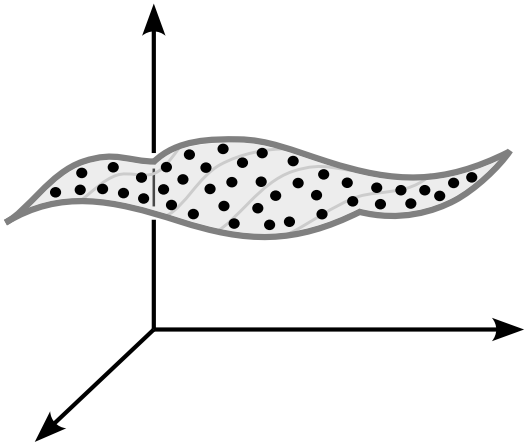


FIGURE 11. Illustration of the Manifold Hypothesis.

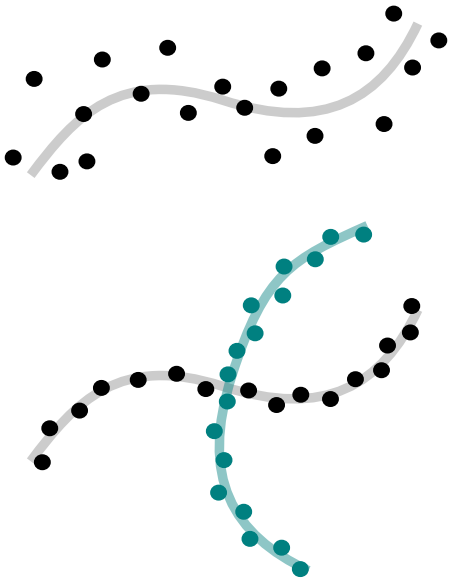


FIGURE 12. Potential problems with the manifold hypothesis.

- Few samples: it may not be possible to accurately estimate the underlying structure of the data due to the limited number of samples
- Multiple manifolds: If the data has multiple underlying structures, it may not lie on a single low-dimensional manifold, making it difficult to map onto a single reduced space.
- **Singularities**

Figure 12
TODO

6.4.1. *Topological Polysemy and singularities in word embeddings.* The paper [JGZ20] studies singular points in a fixed word embedding trained via fastText, see Figure 13 for an illustration. They find that the dimension of the zeroth local homology group of a punctured neighborhood of a word correlates with the number of different meanings of a polysemous word.
TODO

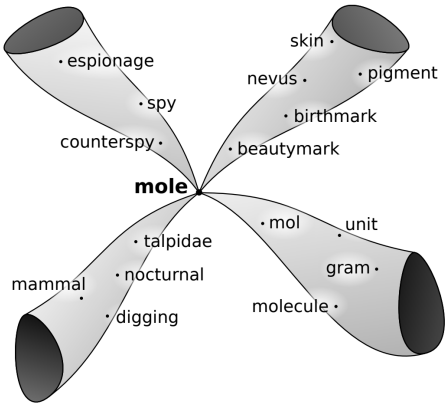


FIGURE 13. Conceptual illustration of a singular point in a static word embedding in the neighborhood of the polysemous word 'mole'.

REFERENCES

[AH19] C. ALLEN and T. M. HOSPEDALES. Analogies Explained: Towards Understanding Word Embeddings. In: *CoRR* abs/1901.09813 (2019). arXiv: 1901.09813 (↑ 5).

[BCB15] D. BAHDANAU, K. CHO, and Y. BENGIO. Neural Machine Translation by Jointly Learning to Align and Translate. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Y. BENGIO and Y. LECUN. 2015 (↑ 8).

[Bar+20] F. BARBIERI, J. CAMACHO-COLLADOS, L. ESPINOSA ANKE, and L. NEVES. TweetEval: Unified Benchmark and Comparative Evaluation for Tweet Classification. In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics, Nov. 2020, pp. 1644–1650. DOI: 10.18653/v1/2020.findings-emnlp.148 (↑ 12).

[Boj+16] P. BOJANOWSKI, E. GRAVE, A. JOULIN, and T. MIKOLOV. Enriching Word Vectors with Subword Information. In: *CoRR* abs/1607.04606 (2016). arXiv: 1607.04606 (↑ 6, 7).

[Bol+16] T. BOLUKBASI, K. CHANG, J. Y. ZOU, V. SALIGRAMA, and A. KALAI. Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings. In: *CoRR* abs/1607.06520 (2016). arXiv: 1607.06520 (↑ 13).

[Cai+21] X. CAI, J. HUANG, Y. BIAN, and K. CHURCH. Isotropy in the Contextual Embedding Space: Clusters and Manifolds. In: *International Conference on Learning Representations*. 2021 (↑ 15).

[CV22] G. CARLSSON and M. VEJDEMO-JOHANSSON. *Topological data analysis with applications*. Cambridge University Press, Cambridge, 2022, pp. xi+220. ISBN: 978-1-108-83865-8. DOI: 10.1017/9781108975704 (↑ 16).

[Che+21] B. CHEN, Y. FU, G. XU, P. XIE, C. TAN, M. CHEN, and L. JING. Probing BERT in Hyperbolic Spaces. In: *CoRR* abs/2104.03869 (2021). arXiv: 2104.03869 (↑ 16).

[CC18] X. CHEN and C. CARDIE. Unsupervised Multilingual Word Embeddings. In: *CoRR* abs/1808.08933 (2018). arXiv: 1808.08933 (↑ 11).

[Cla+19] K. CLARK, U. KHANDELWAL, O. LEVY, and C. D. MANNING. What Does BERT Look at? An Analysis of BERT's Attention. In: *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 276–286. DOI: 10.18653/v1/W19-4828 (↑ 10).

[Con+17] A. CONNEAU, G. LAMPLE, M. RANZATO, L. DENOYER, and H. JÉGOU. Word Translation Without Parallel Data. In: *CoRR* abs/1710.04087 (2017). arXiv: 1710.04087 (↑ 5, 11).

[Dev+18] J. DEVLIN, M. CHANG, K. LEE, and K. TOUTANOVA. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: *CoRR* abs/1810.04805 (2018). arXiv: 1810.04805 (↑ 10).

[DW21] T. K. DEY and Y. WANG. *Computational Topology for Data Analysis*. 2021 (↑ 18).

[Dhi+18] B. DHINGRA, C. J. SHALLUE, M. NOROUZI, A. M. DAI, and G. E. DAHL. Embedding Text in Hyperbolic Spaces. In: *CoRR* abs/1806.04313 (2018). arXiv: 1806.04313 (↑ 16).

[Dos+21] A. Dosovitskiy et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In: *International Conference on Learning Representations*. 2021 (↑ 9).

[Eis19] J. Eisenstein. *Introduction to Natural Language Processing*. Adaptive Computation and Machine Learning series. MIT Press, 2019. isbn: 9780262042840 (↑ 4).

[Eth19] K. Ethayarajh. How Contextual are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings. 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)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 55–65. doi: 10.18653/v1/D19-1006 (↑ 15).

[Fen+20] F. Feng, Y. Yang, D. Cer, N. Arivazhagan, and W. Wang. Language-agnostic BERT Sentence Embedding. In: *CoRR* abs/2007.01852 (2020). arXiv: 2007.01852 (↑ 11).

[Fen+21] S. Feng, N. Lubis, C. Geishauser, H. Lin, M. Heck, C. van Niekerk, and M. Gasic. EmoWOZ: A Large-Scale Corpus and Labelling Scheme for Emotion in Task-Oriented Dialogue Systems. In: *CoRR* abs/2109.04919 (2021). arXiv: 2109.04919 (↑ 12).

[GYC21] T. Gao, X. Yao, and D. Chen. SimCSE: Simple Contrastive Learning of Sentence Embeddings. In: *CoRR* abs/2104.08821 (2021). arXiv: 2104.08821 (↑ 11).

[Gar+19] S. Garg, S. Peitz, U. Nallasamy, and M. Paulik. Jointly Learning to Align and Translate with Transformer Models. 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)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 4453–4462. doi: 10.18653/v1/D19-1453 (↑ 8).

[Gol15] Y. Goldberg. A Primer on Neural Network Models for Natural Language Processing. In: *CoRR* abs/1510.00726 (2015). arXiv: 1510.00726 (↑ 5).

[Gon+20] H. Gonen, G. Jawahar, D. Seddah, and Y. Goldberg. Simple, Interpretable and Stable Method for Detecting Words with Usage Change across Corpora. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 538–555. doi: 10.18653/v1/2020.acl-main.51 (↑ 15).

[Guh+20] O. Guhr, A.-K. Schumann, F. Bahrmann, and H. J. Böhme. Training a Broad-Coverage German Sentiment Classification Model for Dialog Systems. English. In: *Proceedings of the 12th Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, May 2020, pp. 1627–1632. isbn: 979-10-95546-34-4 (↑ 12).

[HLJ16] W. L. Hamilton, J. Leskovec, and D. Jurafsky. Diachronic Word Embeddings Reveal Statistical Laws of Semantic Change. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1489–1501. doi: 10.18653/v1/P16-1141 (↑ 15).

[JGZ20] A. Jakubowski, M. Gasic, and M. Zibrowius. Topology of word embeddings: Singularities reflect polysemy. In: *Proceedings of the Ninth Joint Conference on Lexical and Computational Semantics*. 2020, pp. 103–113. arXiv: 2011.09413 [cs.CL] (↑ 19).

[Jaw+19] P. Jawanpuria, A. Balgovind, A. Kunchukuttan, and B. Mishra. Learning Multilingual Word Embeddings in Latent Metric Space: A Geometric Approach. In: *Transactions of the Association for Computational Linguistics* 7 (Apr. 2019), pp. 107–120. issn: 2307-387X. doi: 10.1162/tacl_a_00257. eprint: https://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00257/1923013/tacl_a_00257.pdf (↑ 11).

[JM09] D. Jurafsky and J. H. Martin. *Speech and Language Processing*. MIT Press, 2009. isbn: 978-0-13-187321-6 (↑ 4).

[KB21] M. Kaneko and D. Bollegala. Debiasing Pre-trained Contextualised Embeddings. In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Online: Association for Computational Linguistics, Apr. 2021, pp. 1256–1266. doi: 10.18653/v1/2021.eacl-main.107 (↑ 13).

[Kar] Karpathy. *The Unreasonable Effectiveness of Recurrent Neural Networks*. http://karpathy.github.io/2015/05/21/rnn-effectiveness/. Accessed: 2022-05-05 (↑ 8).

[Khr+19] V. Khrulkov, L. Mirvakhabova, E. Ustinova, I. V. Oseledets, and V. S. Lempitsky. Hyperbolic Image Embeddings. In: *CoRR* abs/1904.02239 (2019). arXiv: 1904.02239 (↑ 16).

[Kur+19] K. Kurita, N. Vyas, A. Pareek, A. W. Black, and Y. Tsvetkov. Measuring Bias in Contextualized Word Representations. In: *Proceedings of the First Workshop on Gender Bias in Natural*

Language Processing. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 166–172. DOI: 10.18653/v1/W19-3823 (↑ 13).

[LG14] O. Levy and Y. Goldberg. Neural Word Embedding as Implicit Matrix Factorization. In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger. Vol. 27. Curran Associates, Inc., 2014 (↑ 4).

[LGD15] O. Levy, Y. Goldberg, and I. Dagan. Improving Distributional Similarity with Lessons Learned from Word Embeddings. In: *Transactions of the Association for Computational Linguistics* 3 (2015), pp. 211–225. DOI: 10.1162/tacl_a_00134 (↑ 4).

[Li+20] B. Li, H. Zhou, J. He, M. Wang, Y. Yang, and L. Li. On the Sentence Embeddings from Pre-trained Language Models. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 9119–9130. DOI: 10.18653/v1/2020.emnlp-main.733 (↑ 15).

[Liu+21] F. Liu, Y. Jiao, J. Massiah, E. Yilmaz, and S. Havrylov. Trans-Encoder: Unsupervised sentence-pair modelling through self- and mutual-distillations. In: *CoRR* abs/2109.13059 (2021). arXiv: 2109.13059 (↑ 11).

[Liu+19] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. In: *CoRR* abs/1907.11692 (2019). arXiv: 1907.11692 (↑ 10).

[MH08] L. van der Maaten and G. Hinton. Visualizing Data using t-SNE. In: *Journal of Machine Learning Research* 9.86 (2008), pp. 2579–2605 (↑ 18).

[Mao+19] X. Mao, S. Chang, J. Shi, F. Li, and R. Shi. Sentiment-Aware Word Embedding for Emotion Classification. In: *Applied Sciences* 9.7 (2019), p. 1334. ISSN: 2076-3417. DOI: 10.3390/app9071334 (↑ 12).

[Men+19] Y. Meng, J. Huang, G. Wang, C. Zhang, H. Zhuang, L. M. Kaplan, and J. Han. Spherical Text Embedding. In: *NeurIPS*. 2019. arXiv: 1911.01196 [cs.CL] (↑ 15).

[Mik+13a] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient Estimation of Word Representations in Vector Space. In: *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*. Ed. by Y. Bengio and Y. LeCun. 2013 (↑ 5).

[MLS13] T. Mikolov, Q. V. Le, and I. Sutskever. Exploiting Similarities among Languages for Machine Translation. In: *CoRR* abs/1309.4168 (2013). arXiv: 1309.4168 (↑ 5).

[Mik+13b] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed Representations of Words and Phrases and their Compositionality. In: *CoRR* abs/1310.4546 (2013). arXiv: 1310.4546 (↑ 5, 6).

[MT17] D. Mimno and L. Thompson. The strange geometry of skip-gram with negative sampling. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, Sept. 2017, pp. 2873–2878. DOI: 10.18653/v1/D17-1308 (↑ 14).

[MBV17] J. Mu, S. Bhat, and P. Viswanath. All-but-the-Top: Simple and Effective Postprocessing for Word Representations. In: *CoRR* abs/1702.01417 (2017). arXiv: 1702.01417 (↑ 5).

[MR19] J. Murugan and D. Robertson. *An Introduction to Topological Data Analysis for Physicists: From LGM to FRBs*. 2019. arXiv: 1904.11044 [astro-ph.IM] (↑ 18).

[NF18] N. Nakashole and R. Flauger. Characterizing Departures from Linearity in Word Translation. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 221–227. DOI: 10.18653/v1/P18-2036 (↑ 15).

[NK17] M. Nickel and D. Kiela. Poincaré Embeddings for Learning Hierarchical Representations. In: *CoRR* abs/1705.08039 (2017). arXiv: 1705.08039 (↑ 16).

[Ope23] OpenAI. *ChatGPT*. https://chat.openai.com/. Feb. 2023 (↑ 2).

[Ouy+22] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. *Training language models to follow instructions with human feedback*. 2022. arXiv: 2203.02155 [cs.CL] (↑ 14).

[PSM14] J. Pennington, R. Socher, and C. Manning. GloVe: Global Vectors for Word Representation. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*

(EMNLP). Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162 (↑ 5, 6).

[Pet+18a] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. 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, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*. Ed. by M. A. Walker, H. Ji, and A. Stent. Association for Computational Linguistics, 2018, pp. 2227–2237. DOI: 10.18653/v1/n18-1202 (↑ 8).

[Pet+18b] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. In: *CoRR* abs/1802.05365 (2018). arXiv: 1802.05365 (↑ 8).

[Rad+21] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In: *International conference on machine learning*. PMLR. 2021, pp. 8748–8763. arXiv: 2103.00020 [cs.CV] (↑ 12).

[RP21] S. Rajaee and M. T. Pilehvar. A Cluster-based Approach for Improving Isotropy in Contextual Embedding Space. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Online: Association for Computational Linguistics, Aug. 2021, pp. 575–584. DOI: 10.18653/v1/2021.acl-short.73 (↑ 15).

[Ram+22] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen. *Hierarchical Text-Conditional Image Generation with CLIP Latents*. 2022. arXiv: 2204.06125 [cs.CV] (↑ 12).

[Rat+21] A. Rathore, S. Dev, J. M. Phillips, V. Srikumar, Y. Zheng, C. M. Yeh, J. Wang, W. Zhang, and B. Wang. VERB: Visualizing and Interpreting Bias Mitigation Techniques for Word Representations. In: *CoRR* abs/2104.02797 (2021). arXiv: 2104.02797 (↑ 13).

[Rav+20] S. Ravfogel, Y. Elazar, H. Gonen, M. Twiton, and Y. Goldberg. Null It Out: Guarding Protected Attributes by Iterative Nullspace Projection. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 7237–7256. DOI: 10.18653/v1/2020.acl-main.647 (↑ 13).

[RG19] N. Reimers and I. Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In: *CoRR* abs/1908.10084 (2019). arXiv: 1908.10084 (↑ 11).

[RCB21] R. Reinauer, M. Caorsi, and N. Berkouk. *Persformer: A Transformer Architecture for Topological Machine Learning*. 2021. arXiv: 2112.15210 [cs.LG] (↑ 18).

[Ron14] X. Rong. word2vec Parameter Learning Explained. In: *CoRR* abs/1411.2738 (2014). arXiv: 1411.2738 (↑ 5).

[SIS21] I. Schlag, K. Irie, and J. Schmidhuber. Linear Transformers Are Secretly Fast Weight Programmers. In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by M. Meila and T. Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 18–24 Jul 2021, pp. 9355–9366 (↑ 8).

[SVL14] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In: *Advances in neural information processing systems* 27 (2014) (↑ 7).

[TBG18] A. Tifrea, G. Bécigneul, and O. Ganea. Poincaré GloVe: Hyperbolic Word Embeddings. In: *CoRR* abs/1810.06546 (2018). arXiv: 1810.06546 (↑ 16).

[Vas+17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention Is All You Need. In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762 (↑ 8, 9).

[Voi] Voita. *Sequence to Sequence (seq2seq) and Attention*. https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html. Accessed: 2022-05-05 (↑ 7).

[Yu+17] L.-C. Yu, J. Wang, K. R. Lai, and X. Zhang. Refining Word Embeddings for Sentiment Analysis. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, Sept. 2017, pp. 534–539. DOI: 10.18653/v1/D17-1056 (↑ 12).

[Zha+21] Y. Zhang, M. Choi, K. Han, and Z. Liu. Explainable Semantic Space by Grounding Language to Vision with Cross-Modal Contrastive Learning. In: *CoRR* abs/2111.07180 (2021). arXiv: 2111.07180 (↑ 12).

[Zha+19] J. Zhao, T. Wang, M. Yatskar, R. Cotterell, V. Ordonez, and K.-W. Chang. Gender Bias in Contextualized Word Embeddings. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 629–634. doi: 10.18653/v1/N19-1064 (↑ 13).

[Zho+22] Z. Zhou, D. Zhang, W. Xiao, N. Dingwall, X. Ma, A. O. Arnold, and B. Xiang. *Learning Dialogue Representations from Consecutive Utterances*. 2022. doi: 10.48550/ARXIV.2205.13568 (↑ 11).

[Zhu+20] Y. Zhu, D. Zhou, J. Xiao, X. Jiang, X. Chen, and Q. Liu. HyperText: Endowing FastText with Hyperbolic Geometry. In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics, Nov. 2020, pp. 1166–1171. doi: 10.18653/v1/2020.findings-emnlp.104 (↑ 16).

Dialog Systems and Machine Learning Group, Heinrich-Heine-University Düsseldorf, Germany
Email address: benjamin.ruppik@hhu.de
URL: http://bruppik.de