Understanding and exploiting subspace organization in contextual word embeddings

David Yenicelik
ETH Zürich

A dissertation submitted to ETH Zürich
in partial fulfilment of the requirements for the degree of
Master of Science ETH in Computer Science

Under the supervision of
Florian Schmidt
Yannic Kilcher
Prof. Dr. Thomas Hofmann

Eidgenössische Technische Hochschule Zürich
Data Analytics Lab
Institute of Machine Learning
CAB F 42.1, Universitätsstrasse 6, 8006, Zürich
Zürich 8006
Switzerland

Email: yedavid@ethz.ch

May 13, 2020

Declaration

I David Yenicelik of ETH Zürich, being a candidate for the M.Sc. ETH in Computer Science, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose.

Total word count: 32,612

Signed:

Date:

Abstract

One of the goals of Natural Language Understanding *NLU* is to formalize written text in a way, such that linguistic features can be captured through computational models, which can then again be used for downstream machine learning tasks. The most popular methodology in NLU is the use of vectors that represent written text, including word-vectors, sentence-vectors, etc. These vectors are often referred to as *embedding* vectors, as they embed more complex concepts into a vector representation.

Because embeddings - due to their strong usage in downstream tasks - can be considered as the fundamental unit of machine learning in the domain of natural language, any shortcomings will propagate to the performance of the target task. Thus, improving the way these embeddings have strong implications for any subtasks in NLU.

Given that language models such as BERT only capture a function black-box function f between input and output, our aim is to understand how the the subspace organization relates to linguistic features. Due to the strong importance of *semantics*, which captures meaning in text, we will focus our analysis on *semantics*.

Our main contribution are two-fold. We first test to what extent the semantic subspace is linearly separable between semantic classes, and then test for multi-modality in these structures through a clusterability test. We then analyse the relationship between the linguistic features of part-of-speech and semantics, as these show strong correlation in languages. Finally, we introduce new embedding-vectors for tokens with high variance, and try to understand what downstream tasks are affected the most, using this as a proxy to understand what the strong word-based variance in BERT context vectors most corresponds to.

Our findings show that although BERT does not capture an interpretable concept of a semantic subspace, but that it rather focuses on the more broad linguistic features which are imminent to context. This can often introduce undesirable features such as strong sentiment, position in sentence, and thus lead to considerable bias in downstream applications.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Although a lot of progress has been made through programs such as *ELIZA*, and more modern examples such as Siri and Google Duplex which show that a machine can have conversations with humans, it is not yet apparent that machines can think, or in other words, operate with semantic concepts. In most of these cases, the underlying language model uses some heuristics, or black-box language function f which simply maps one input to another. Although the relation between language and thought is a much more fundamental and philosophical one, for this work we assume that semantic concepts can be captured as some latent variable, i.e. language is just an expression of thought.

One of the goals of Natural Language Understanding NLU is to address this shortcoming, by formalizing written text in a way, such that linguistic features can be captured through computational models, which can then again be used for downstream machine learning tasks. The most popular methodology in NLU is the use of vectors that represent written text, including word-vectors, sentence-vectors, and more. These vectors are often referred to as *embedding* vectors, as they embed more complex concepts into a vector representation. Optimally, the algebra of the space that we operate in, including the various operations such as multiplication and addition should have meaningful relations to the concepts captured by these embeddings.

The result of this work includes language models and embedding vectors.

Language models *LM* are a generalization of NLU models that can both generate word-embeddings, sentence embeddings, but also intermediate representations that can be used for downstream tasks. Most commonly, language models take into consideration the context that a word token appears and, and includes this in the calculation when creating the embedding.

Because embeddings - due to their strong usage in downstream tasks - can be considered as the fundamental unit of machine learning in the domain of natural language, any shortcomings will propagate to the performance of the target task. Thus, improving the way these embeddings have strong implications for any subtasks in NLU, including but not limited to named entity recognition *NER*, sentiment analysis, and translation between languages. However, most of the modern LMs are so complex that these are considered black-box models.

Although this work does not try to imminently push the performance of these language models, we believe that providing a better understanding of the underlying principles will pave the way for future research to better track these issues. For the sake of focus, and because we believe that meaning is the most important linguistic feature for communication, we will focus on investigating how modern LMs capture semantics.

1.0.1 Main Contributions

Our main contribution test to what extent the semantic subspace is linearly separable between semantic classes, and then test for multi-modality in these structures through a clusterability test. We then analyze the relationship between the linguistic features of part-of-speech and semantics, as these show a strong correlation in languages. Finally, we introduce new embedding-vectors

for tokens with high variance and try to understand what downstream tasks are affected the most, using this as a proxy to understand what the strong word-based variance in BERT context vectors most corresponds to. In all of our work, we focus on BERT, as it provides a good balance between popularity, close-to-state-of-the-art performance, and generalizability on other modern language models.

Our findings show that although BERT does not capture an interpretable concept of a semantic subspace, it rather focuses on the more broad linguistic features which are imminent to context. This can often introduce undesirable features such as strong sentiment, position in sentence, and thus lead to considerable bias in downstream applications. We also show that linguistic features such as sentiment are much more strongly captured in modern language models, allowing for language stereotypes to easily manifest itself, leading to stronger cases of bias in language models.

We will first start with summarizing the required background work 2, show existing analysis on the inner workings of BERT 3, conduct tests that do not modify the BERT model 4 and make modifications in BERT to understand what effect this has on downstream tasks 5. We will finally draw conclusion and show potential for future wor 6

Chapter 2

Background

[49] was one of the early works that argue that there is an inherent structure in language and that this structure can be formalized. They also mention that the relation between the linguistic representation in terms of sounds and tokens is related to the meaning that the representation entail.s This representation is often captured in the form of a word token w, such as `bank` or `cat`. However, despite the obvious relationship, the distinction between the distributional structure of the language and semantics is not a formal one. For the above case, the token `bank` may have different meanings $m_{\text{bank-financial instution}}$, $m_{\text{bank-sea bank}}$, whereas the token `cat` will have a more straight-forward semantic interpretation of $m_{\text{cat-animal}}$. [49] argues that there is a parallel semantic structure, and argues that this is not a one-to-one relation between vocabulary and different semantic classes. Generally, one of the main views of this paper is that the meaning of a word is defined by the context it carries. The formalization of language is not a trivial problem to consider, as this also touches base on a philosophical level [50], [138], posing the question on the nature of the relation between thought and (language)-representation.

2.1 Linguistic Features

There is a vast number of linguistic features, going from phonological features, morphological and syntactic features to semantic features. This is not an exhaustive summary, but focusing on the properties of languages relevant to this work.

We will first introduce some linguistic features relevant to this work. These make a formal analysis of our topic easier.

Polysemy describes the phenomenon that a word token w may have multiple meanings. As a simple example, the number of meanings that `cat` can entail is $|M_{\text{cat}}| = |\{m_{\text{cat-animal}}\}| = 1$, whereas the number of meanings that `bank` can entail is $|M_{\text{bank}}| = |\{m_{\text{bank-financial instution}}, m_{\text{bank-sea bank}}\}| = 2$. These are just examples, but the true numbers vary depending on the granularity chosen by humans (which again is a non-trivial question of when one concept stops and another distinct concept starts) and the language viewed. This definition and investigation are often left to linguists to decide.

Part of Speech *PoS* implies the category of the syntactic function of a word. Classes including adjectives *ADJ*, adposition *ADP*, adverbs *ADV*, auxiliary *AUX*, conjunction *CONJ*, coordinating conjunction *CCONJ*, determiner *DET*, interjection noun *INTJ*, noun *NOU*, numeral *NUM*, particle *PART*, pronoun*PRON*, proper noun *PROPN*, punctuation *PUNCT*, subordinating conjunction *SCONJ*, symbol *SYM*, verb *VERB*, as is defined by [3]. In this work, we will be using the spaCy python package [54] whenever we need to apply computation on word-tokens to identify the underlying part of speech class, and limit ourselves to high-level part-of-speech tags include nouns, verbs, and adjectives. For a more extensive list of linguistic features, please refer to A.1.

2.1.1 Tokens and n-grams

A token is often considered the most basic unit of language. In the above sections, we have implicitly assumed that this token is always a word w . However, there is a variety of ways that sentences can be encoded into a sequence of tokens. In a sentence such as

`The man travelled down the road.`

the set of word-tokens would constitute would be {down, man, road, the, traveled}. The way a sentence is split up into an ordered array of individual tokens is referred to as *tokenization*. Tokenizing the above sentence (assuming we only allow lower-case tokens) would result in

$$[the, man, traveled, down, the, road]$$

It is important to note that there are also other ways to interpret language tokens. I will provide insights on different formats, as the language models that we will be using a different tokenization method.

Characters can be considered as another base unit of language. For the sentence (**??**), the set of basic units would correspond to the Latin alphabet plus whitespace and numbers $a - z""0 - 9$ (" " denotes whitespace), and the above sentence would be tokenized into the sequence

$$[t, he, "", m, a, n, "", t, r, a, v, e, l, e, d, "", d, o, w, n, "", t, h, e, "", r, o, a, d]$$

Choosing characters as the basic token levels has shown considerable success and popularity in language modeling [119] and translation [67].

Byte pair encodings [42] presents another popular mechanism to tokenize sentences, and also shows success in the case of language modeling and translation [108]. Here, the most frequent pair of bytes in a sequence

is replaced with the most frequent pair of bytes in a sequence with a single, unused byte. This can also be used as a compression scheme.

Depending on what corpus this tokenizer is trained on, the above sentence could be tokenized into the following sequence

$$[the., "", ma., n., "", t., rav., e., led., "", do., w., n., "", the., "", r., oa., d.]$$

The tokenization of the sentence above implies that the corpus that the tokenizer was trained on includes as a majority of word-occurrences the sequence of characters *the*, as well as *lled* and *do*. The . (dot) at the end of each character denotes the end of the byte-pair. The main idea behind this tokenization is to construct a vocabulary of frequent subwords.

WordPiece tokenizer [140] follows a similar idea as the byte-pair encoding and constructs a vocabulary given a corpus by maximizing entropy. The main difference to the byte-pair encoding is that the vocabulary naturally includes a set of the most commonly appearing words in the respective language, and is then aggregated by introducing additional subword units to cover unseen character sequences.

The tokenization of **??** would then look as follows (depending on which implementation and version of the WordPiece tokenizer is used)

$$[the, man, travel, \#\#led, down., the, ro, \#\#ad.]$$

where we can see that words such as `the` and `down` are not further divided into tokens and that suffixes such as `##led` are introduced, because these are unregular patterns which allow for better generalization.

2.2 Word Embeddings

We will now present ways to capture relations between tokens by representing these through vectors.

Word-Embeddings In general, we want to find a mapping

$$w \mapsto (x_w, b_w) \in \mathcal{X}^{d+1} \tag{2.1}$$

where w is a token representation from a vocabulary $w \in V$ and where this token representation is transformed into a $d+1$-dimensional vector representation x_w, and a bias-term b_w that is specific to the token w. Whenever we talk about *word-vectors*, *(word)-embeddings*, or *feature-vectors*, we will refer to the image of the above map. For convenience and unless stated otherwise, we will assume that x_w absorbs the bias term b_w as an additional vector-element. Also, for simplicity and unless otherwise stated, we will use the euclidean real-valued vector-space $\mathbb{R}^{d+1}$ to described the resulting embedding vectors. Please note, however, that the choice of the embedding space $\mathcal{X}$ is not fixed in general, and as such, work in other spaces such as hyperbolic spaces [43] have also been conducted.

Distance: Our goal is to build an embedding space where the distance between word-embeddings correspond to the relation between words (i.e. the semantics entailed by their tokens). Formally, we introduce the concept of *distance* $d : \mathcal{X} \times \mathcal{X} \mapsto [0, \inf)$, which should capture the relation between different elements. For a set of elements $x, y, z \in \mathcal{X}$, following properties must hold for a mapping to be a valid distance metric:

1. $d(x, y) \geq 0$ (non-negativity)
2. $d(x, y) = 0 \iff x = y$ (identity, i.e. if two embedding vectors are identical, they must capture the same underlying word instance)
3. $d(x, y) \leq d(x, z) + d(z, y)$ (triangle inequality)

One consequence of the above rules is that for words a, b, c, each having a word embedding x, y, z respectively, $d(x, y) < d(x, z) \iff$ word instance b is conceptually closer to word a than word c. For convenience, whenever I input a, b, c into the distance function d, the word-embeddings for the corresponding word-tokens shall be used. Also, please notice that I left out the notion of symmetry for distance measures.

Learning a distance A popular paradigm in machine learning is to maximize a posterior probability distribution given some data $\mathbf{X}$. Following the idea of the distributional structure of language [49]. In the context of word vectors, we want to maximize the probability that w occurs in the context window of $w\prime$ through some parameters θ. Implicitly, this corresponds to minimizing the distance between x_w and x'_w, while keeping the distance between x_w and all other word-vectors constant. We call $w\prime$ a *context word* for w.

$$p\left(w|w'\right) \tag{2.2}$$

and in the work we focus at, the probabilistic maximization problem corresponds to minimizing the distance in the embedding space

$$\forall w, w' : \qquad \max p\left(w|w'\right) \iff \min d(w|w') \tag{2.3}$$

Please note that we do not generalize to formalizing a dual-relationship between the two problem-statements.

Exploiting the distributional structure of language: Our goal is to learn distance between words by exploiting the distributional structure of a set of sentences, which make up a *corpus*. One of the early formalizations of representing the distributional structure of words through was expressed in [13], which argues that a sequential statistical model can be constructed to estimate this true posterior. In it's most naive form, this would imply that

we can estimate the probability of a sequence of words $\mathbf{w}$ as

$$p(\mathbf{w}) = \prod_{t=1}^{T} p\left(w^{(t)} | w^{(t-1)}, \ldots, w^{(1)}\right) \tag{2.4}$$

where $\mathbf{w} = w^{(1)}, \ldots, w^{(T)}$ is the sequence of words, $p(\mathbf{w})$ the probability of this sentence occurring. This corresponds to an autoregressive model. One could for example use (hidden) markov models to model this relation, as we could fix the *context size* n (in this case, only looking at previously occurring words) and exploit the n-step markovian assumption of

$$p\left(w^{(t)} | w^{(t-1)}, \ldots, w^{(1)}\right) = p\left(w^{(t)} | w^{(t-1)}, \ldots, w^{(t-n+1)}\right) \tag{2.5}$$

Fixing the context window to n words for each computation (i.e. convolution), introduces the concept of *n-grams*. *n-grams* can also occur with characters, where n characters are fed in to some model at a certain timestep t.

Next to a context which only consists of the *previous* n words, one can also regard a context which includes both the previous *and subsequent* R words. Combining this with a probability model which assumes conditional independence between all but one context word, this leads us to

$$p(\mathbf{w}) = \prod_{t=1}^{T} \prod_{\triangle \in \mathcal{I}} p\left(w^{(t)} | w^{(t+\Delta)}\right) \tag{2.6}$$

whose probability we wish to estimate (and maximize if this is in the given training dataset), $\mathcal{I} = \{-R, \ldots, -1, 1, \ldots, R\}$ is the context window.

Loss Objective The above function is a probability estimate of the true underlying distribution. However, given that computational power is limited,

we are usually interested in learning a parametrized model which captures this underlying distribution. One way to learn this parametrized model is to **maximize** a loss by backpropagating the gradients of the parameters. In that case, the loss function would look as follows.

$$\mathcal{L}(\theta; \mathbf{w}) = \prod_{t=1}^{T} \prod_{\triangle \in \mathcal{I}} p_\theta \left(w^{(t)} | w^{(t+\Delta)} \right) \tag{2.7}$$

By maximizing the mean loss over all sentences in a big-enough corpus $\mathcal{C}$, we can find a reliable estimator model with parameters $\hat{\theta} = \text{argmin}_\theta \mathcal{L}(\theta; \mathbf{w})$ using a maximum likelihood estimation approach. One would usually work with the log-loss as this allows for thee gradients to be more easily computed and avoids numerical issues.

Intuitively the above models flow the idea expressed by [49] very well, speaking that the meaning of a word is captured by its neighborhood. However, the above equations follow a *continuous bag of words CBOW* approach, which aims at predicting a target word w^t given some context words w'. In contrast, it is also possible to follow a *skip-gram SG* approach, where the aim is to predict context words w' given a target word w^t. The set of optimal models usually does not change $\theta \in \Theta$, whereas the chosen model $\hat{\theta}$ is different. The skip-gram model is found to work better with rare words, whereas the CBOW model seems to have slightly better accuracy on words that are occurring more frequently.

Using a skip-gram approach, (2.2) for example would be reformulated into

$$\mathcal{L}(\theta; \mathbf{w}) = \prod_{t=1}^{T} \prod_{\triangle \in \mathcal{I}} p_\theta \left(w^{(t+\Delta)} | w^{(t)} \right) \tag{2.8}$$

Although we have mentioned that the probability density distribution p is often modelled through a parametrized function (due to the sheer amount of data within corpora), we have not presented methods on how this parametriza-

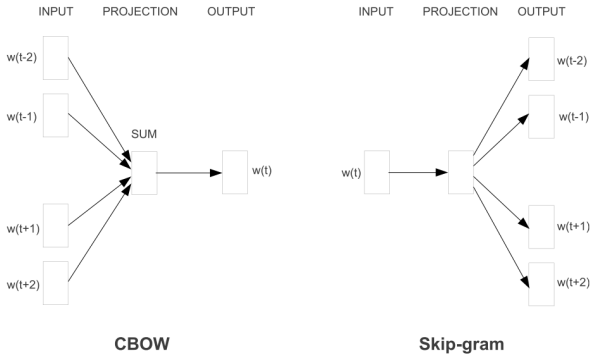


Figure 2.1: Figure taken from [81]. The CBOW architecture predicts the current word based on the context. The Skip-gram predicts surrounding words given the current word.

tion can be achieved. In the following section, we will show how vectors for x_w and b_w can be found, and other ways to parametrize the probability density function p. We will not go into too much detail as to how these models are trained, as all of them can be trained by applying a gradient-based optimization on the loss term.

2.2.1 Static Word Embeddings

Here we will talk about word-embeddings where each word-token only has a single embedding x_w, i.e. there is a bijection between word-tokens and word-embeddings. Specifically, the mapping (2.1) is not a probabilistic function with a latent random factor. We will start with simple static word embeddings and proceed to more complex models in subsequent sections.

Basic Model

The first model we are going to look at is a most basic model which fulfills the properties of the distance metrics shown in (2.2) and minimizes words within the same context.

Here, we can introduce a *log-bilinear* [1] model where the log-probability is define as

$$\log p_\theta(w|w') = \langle \mathbf{x}_w, \mathbf{x}_{w'} \rangle + b_w + \text{ const.} \tag{2.9}$$

The actual probability density function can be calculated by exponentiating the log-probability

$$p_\theta\left(w|w'\right) = \frac{\exp\left[\langle \mathbf{x}_w, \mathbf{x}_{w'} \rangle + b_w\right]}{Z_\theta\left(w'\right)}$$

where $Z_\theta\left(w'\right) := \sum_{v \in \mathcal{V}} \exp\left[\langle \mathbf{x}_v, \mathbf{x}_{w'} \rangle + b_v\right]$ is a normalization constant such that the probability mass sums to 1, and the model parameters entail the word-embeddings $\theta = (x_w, b_w) \in \mathbb{R}^{d+1}$

Because we have $\theta = \forall w \in \mathcal{V} : \{x_w, b_w\}$, we can use a gradient-based loss-minimizing method which minimizes the cost function for θ, for a word w and all their possible context words $w' = w^{(t+\Delta)}$ within a context size R.

$$\mathcal{L}(\theta; \mathbf{w}) = \sum_{t=1}^{T} \sum_{\Delta \in \mathcal{I}} \log p_\theta\left(w^{(t+\Delta)}|w^{(t)}\right)$$

.

This basic model comes with drawbacks. Amongst others the distance would be minimized if all word-embeddings would collapse onto a single point. No term that forces unrelated words to move away from each other, a property that we are interested in as isotropy for example has shown practical performance [39].

Word2Vec

One of the most prominent of word vector models is proposed by [81, 82]. Here, a neural network with a single embedding layer can be trained to transform one-hot-vectors $\in \{0,1\}^{|\mathcal{V}|}$ which represents a word w in vocabulary $\mathcal{V}$ into a latent vector representation $x_w \in \mathbf{R}^{d+1}$ using a neural network - which boils down to a linear embedding matrix W. Both a continuous bag of word and also a continuous skip-gram approach can be used. The skip-gram approach is preferred in practice. Specifically, the loss-function that is optimized looks as follows

$$\mathcal{L}(\theta; \mathbf{w}) = \sum_{t=1}^{T} \sum_{\Delta \in \mathcal{I}} [\tag{2.10}$$

$$b_{w^{t+\Delta}} + \langle \mathbf{x}_{w^{(t+\Delta)}}, \mathbf{x}_{w^{(t)}} \rangle \tag{2.11}$$

$$- \log \sum_{v \in \mathcal{V}} \exp \left[\langle \mathbf{x}_v, \mathbf{x}_{w^{(t)}} \rangle + b_v \right] \tag{2.12}$$

As one can see, the loss function takes as input the bilinear loss from the basic model, and complements this by adding a regularizing term, such that random samples are not put next to each other. This idea is referred to as *negative sampling*.

Backpropagation is used to optimize over the networks weights. Also, highly frequent words are optionally subsampled and their frequency is re-weighted

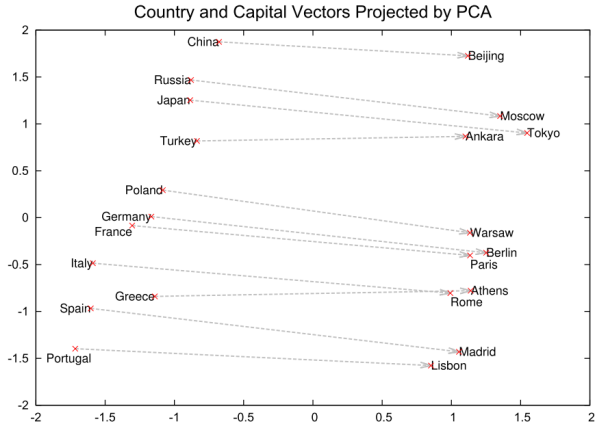


Figure 2.2: Figure taken from [82]. A 2-dimensional PCA projection of the 1000-dimensional skip-gram vectors of countries and their capital cities. The proposed model is able to automatically organize concepts and learn implicit relationships between them. No supervised information was provided about what a capital city means.

using the formula

$$P\left(w_i\right) = 1 - \sqrt{\frac{t}{f\left(w_i\right)}}$$

where $f(\cdot)$ is the function that returns the frequency of a word w_i in a given corpus.

GloVe

For the global vectors for word representation *GloVe* [92], the authors follow a matrix factorization approach. First, a global co-occurrence matrix is created $\mathbf{N} = (n_{ij}) \in \mathbb{N}^{|\mathcal{V}||\mathcal{C}|}$ where each entry n_{ij} is determined by the number of occurrences of word $w_i \in \mathcal{V}$ in context $w_j \in \mathcal{C}$. Given that the vocabulary size can exceed multiple thousand items, this practically results in a sparse matrix.

$$\mathcal{H}(\theta; \mathbf{N}) = \sum_{i,j} f\left(n_{ij}\right) \left(\underbrace{\log n_{ij}}_{\text{target}} - \underbrace{\log \tilde{p}_\theta\left(w_i | w_j\right)}_{\text{model}}\right)^2 \tag{2.13}$$

$$= \sum_{i,j} f\left(n_{ij}\right) \left(\log n_{ij} - \langle x_i, y_j \rangle\right)^2 \tag{2.14}$$

$$\tag{2.15}$$

where $f(n_{ij})$ is the weighting function which assigns a weight for each entry in the co-occurence matrix based on the co-occurence of words w_i and w_j. In the second line, we also again use a bilinear model $\tilde{p}_\theta\left(w_i | w_j\right) \propto \exp\left[\langle \mathbf{x}_i, \mathbf{y}_j \rangle + b_i + c_j\right]$. The constants b_i, c_j are left out and are assumed to be absorbed in the embedding vectors. A popular choice for the weighting function is $f(n) = \min\left\{1, \left(\frac{n}{n_{\max}}\right)^\alpha\right\}$ with $\alpha \in (0, 1]$. The motivation behind this is that frequent words do not receive a word-frequency which is considered too high (there is a cutoff at some point) and small counts are considered noise and slowly cancelled out. Further extensions which include other basic units of include for example the fastText embeddings [18].

Further extensions have been recorded to model word embeddings as probability densities using a Bayesian skip-gram approach [21], where the resulting model architecture is similar to the word2vec neural network, with a probabilistic latent module instead of a deterministic matrix. Yet another line of work [141] aims to exploit the formal properties of Wasserstein distances to learn word embedding. Many concepts from the above ideas have been translated into analogous probabilistic concepts. We present Gaussian embeddings as one of the possible probabilistic extensions to word embeddings in the Appendix under Section A.2.

2.2.2 Context Embeddings

In contrast to static word embeddings, context embeddings rely not only on the target word w as input, but also require a subseet of the context around $[c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_T]$ it as input. Specifically, the mapping (2.1) is modified in such a way to take as input the previous context words $c_{\text{previous}} = w^{(t)}, \ldots, w^{(t-d+1)}$, and optionally the subsequent context word $c_{\text{subsequent}} = w^{(t-d-1)}, \ldots, w^{(1)}$ as well.

$$(c_{\text{previous}}, w, c_{\text{subsequent}}) \mapsto (x_w) \in \mathcal{X}^{d+1} \tag{2.16}$$

As an example, while static word embeddings would produce the same vector x_0 for all occurrences of the word *bank*, the context embedding would produce different embeddings x_1, x_2 and x_3 respectively for the three sentences:

```
I withdrew some cash at the bank's ATM.
The bank was closed.
I walked down the sea bank.
```

Any change in the context words will generally result in a different embedding x. Capturing this logic through a probability density function, we now calculate the probability

$$p(w^{(t-d)}|c_{\text{previous}}, w^{t-d}, c_{\text{subsequent}}) = p(w^{(t-d)}|w^{(t)}, \ldots, w^{(t-d+1)}, w^{(t-d)}, w^{(t-d-1)}, \ldots, w^{(1)}) \tag{2.17}$$

instead of independently querying the probability $p(w^{(t-d)})$ without any context words.

There are different ways to capture this probability, from simple markov models, to LSTMs, to transformers. We will go over the most prominent models of the past few years, discussing their underlying mechanism.

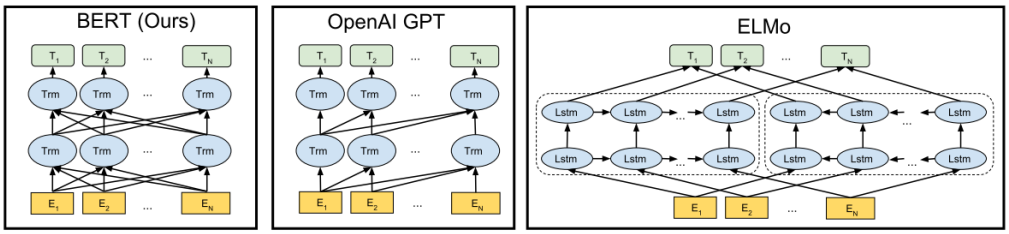


Figure 2.3: Figure taken from [36]. BERT uses a bidirectional transformer, which is not limited to reading in all the input from only left to right or right to left. OpenAI GPT (next section) uses a left-to-right Transformer, while ELMo is using a bidirectional LSTM which naturally captures a single direction per LSTM.

ELMo

The simplest idea of context embeddings, which take into account more than just a single word, but all tokens from the context $\mathbf{c} = (c_{\text{previous}}, c_{\text{subsequent}})$is the *Embeddings from Language Models* (ELMo) model proposed by [93].

ELMo contains the Long Short-Term Memory *LSTM* memory cell as it's basic unit. To get a short overview of the architecture of the LSTM cell, and the resulting recurrent neural network, we refer the reader to Section A.3 in the Appendix. When we make use of two LSTM networks, one that reads the sequence in the positive direction (i.e. $[\ldots, x_{t-1}, x_t, x_{t+1}, \ldots]$ in this particular direction) and another one that reads the sequence in a negative direction (i.e. $[\ldots, x_{t+1}, x_t, x_{t-1}, \ldots]$ in this particular direction), the resulting architecture is called a *bidirectional LSTM*, also called *biLSTM*. ELMo consists of two layers of such bidirectional LSTMs, where the output of the first layer is fed in as the input to the second layer. Specifically, the loss function that the language model optimizes during unsupervised training results in

$$\sum_{k=1}^{T} \Big(\log p \left(w_k | w_{k-1}, \ldots, w_1; \Theta_x, \vec{\Theta}_{LSTM}, \Theta_s \right) \tag{2.18}$$

$$+ \log p \left(w_k | w_{k+1}, \ldots, w_T; \Theta_x, \overleftarrow{\Theta}_{LSTM}, \Theta_s \right) \Big) \tag{2.19}$$

where p is the output of an LSTM model parametrized by it's weights θ, including a softmax head to arrive at a valid probability distribution amongst candidate words in the vocabulary $\mathcal{V}$.

Given that the LSTM is a sequential model, it outputs as many hidden representations as there are timesteps T that are given as input to the model. Each LSTM layer produces one set of such hidden representations. ELMo concatenates the hidden representations for both of the bidirectional LSTM layers to arrive at a single context embedding $x_{w_i} = [h_{w_i}^{\text{backward}}, h_{w_i}^{\text{forward}}]$.

Formally, one would pre-train this model on a huge corpus in an supervised way through masked pre-training where the input sequence $[w_1, \ldots, w_i, \ldots, w_T]$ is also expected as the output sequence offset by one timestep $[w_2, \ldots, w_{i+1}, \ldots, [PAD]]$. For further details on pre-training ELMo we refer the reader to [62]. This pre-trained language model would then be extended by another linear layer, which would be adapted for a downstream task. This downstream task could be named entity recognition, part of speech tagging etc. The gradients of the ELMo biLSTM model are all updated during fine-tuning.

The Transformer Architecture

Although ELMo creates context embeddings, the performance is limited to LSTMs which can only capture timestep-directed sequences. Specifically, the LSTM used within ELMo is a sequential timestep model which covers a storage mechanism within the weights it uses, theoretically resulting in a 1-step markov model

$$\begin{aligned} p(w_1, \ldots, w_T) = & \, p(w_T | w_{T-1}; h_{T-1}, \theta) \\ & \ldots p(w_t | w_{t-1}; h_{t-1}, \theta) p(w_{t-2} | w_{t-2}; h_{t-2}, \theta) \\ & \ldots p(w_0; \mathbf{0}, \theta) \end{aligned} \tag{2.20}$$

where h_i is the forward-propagation vector outputted by the LSTM cell at timestep i, which are supposed to "store" inforrmation. Although intuitively

h_{T-1} is supposed to capture information from many timesteps back, there is no explicit information flow between the LSTM cell at timestep $t+1$ and $t-1$ (i.e. any two cells which are more than 2 steps away).

The transformer architecture [125] improves on this shortcoming. The transformer architecture introduces the concept of attention for modern language models, which can model relations between any two input tokens w_i, w_j with $i \neq j$. For a more detail treatment of the attention mechanism, we refer the reader to [11]. The attention mechanism takes all the latent token representations for the full sample sentence (not only the previous or subsequent context) and calculates a correlation between the hidden representations of each timestep. The transformer outputs a latent representation which can be converted to softmax probabilities through a linear head.

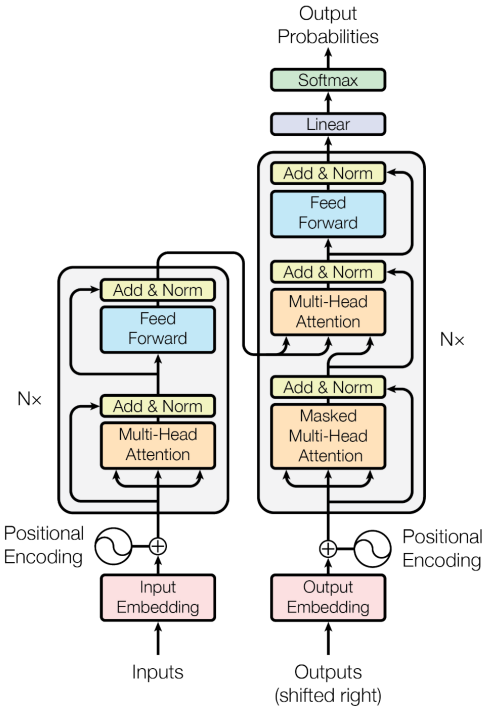


Figure 2.4: Figure taken from [125]. The architecture of the transformer module which encapsulates multiple attention modules.

For each word, a vector representation is computed using an embedding layer.

This results in a set of vectors M which represents the values, and K which represents the keys. One can then take set of query vectors Q and calculate the inner product $\langle K, Q \rangle$. One then applies a softmax this set of dot-products to arrive at a notion of which two vector are similar to each other. In the case of self-attention, the query vectors Q are equivalent to K. This allows for the model to be more attentive between two elements, and allows the model to focus exploit relationships between different tokens within a token-sequence better.

This has implications for the complexity of the model. Instead of calculating the raw probability of (**??**), and even using a markovian assumption because computation power is expensive, the transformer architecture implicitly calculates the joint probability of an entire sequence of words, thus decreasing the step of computations through which information flows.

We will not go into further detail of how and why the transformer architecture works better than the LSTM to keep focus. All of the below presented models use the transformer architecture as a building block instead of the LSTM used in ELMo. The following architectures are all based on the transformer module.

BERT

The Bidirectional Encoder Representations from Transformers **BERT** [36] improves uses the transformer architecture instead of recurrent neural networks. Although there are different version of BERT published, the base model of BERT consists of 12 attention layers, which output a 768-dimensional hidden representation of the input sequence. This hidden representation - which is produced for each item in the input sequence - can be used for downstream tasks. The main advantage of BERT is the more direct information flow through the attention mechanism. Unlike LSTMs, comparisons between any two input tokens can be made, without relying on intermediate latent variables which may block the flow of information. Formally, the attention mechanism allows for a direct comparison of hidden states h_t and h_k with

($|t-k| > 1$). These hidden states h_t and h_k can be interpreted as context embeddings for the workd w_t and w_k which are passed as the sequence $[w_1, \ldots, w_t, \ldots, w_k, \ldots, w_T]$ into BERT. in contrast an LSTM would contain a single hidden state h_{t-1} passed forward at every timestep, which would have to contain the information for all previous words h_k where $k < t-1$. by using multiple attention layers of on top each other, BERT is a deep model, allowing for more complex patterns to be trained.

BERT is **pre-trained in two phases**. In the first phase BERT is trained using a **masked language model** approach. Sentences with a maximum length are sampled from a corpus. About 15% of words in the sentence are replaced with the `[MASK]` token, and the weights are optimized in such a way to predict the word which was replaced by the `[MASK]` token.

```
[CLS] The man went to [MASK] store. [SEP]
```

Figure 2.5: Example from [36]. An sentence where 15% of the tokens are replaced with the [MASK] token. During the firrst phase of pre-training, the weights of the BERT model are optimized in such a way to predict the true underlying words. In this case, the word to be predicted is `the`

The second pre-training phase is a **Next Sentence Prediction** task, where BERT is supposed to predict a sentence s_i given its predecessor sentence s_{i-1} in a fulltext corpus. This is a binarized task, which means that given inputs s_{i-1}, s_r, BERT is supposed to predict whether or not sentence $r = i$ (i.e. s_r is the subsequent sentence to s_{i-1}, or whether s_r is some randomly sampled sentence). 50 % of the training set here consists of randomly sampled sentences, and the other 50 % of the training set consists of the actual next sentence for a given corpus.

```
[CLS] the man went to [MASK] store [SEP]
he bought a gallon [MASK] milk [SEP]
```

Figure 2.6: Example from [36]. Two input sequences which where 15% of the tokens are replaced with the [MASK] token. During pre-training, the weights of the BERT model are optimized in such a way to predict the true underlying words. In this case, the second sentence is a continuation of the first one, and thus the label would be *isNext*.

In addition to the expensive pre-training which is applied on a 3.3 billion word corpus, BERT is a language model which is supposed to be fine-tuned to specialized downstream tasks. For this, the learning rate is kept lower, while the gradient updates all model weights. One can either apply an additional linear layer on top of the T hidden representations produced, or use specialized outputs in BERT to apply prediction tasks.

Multiple versions of BERT are provided, including $BERT_{LARGE}$ and $BERT_{BASE}$, each taking as input a sequence of maximal length 512, as the learned attention weights inside the transformer architecture only allow a maximal sequence length. Because we will make modifications to the inner workings of BERT, mostly to the transformer modules itself, we now present the full pipeline.

The pipeline of BERT starts with a sentence s which is tokenized into tokens $[t_1, \ldots, t_T]$ using the WordPiece tokenizer discussed in section 2.1.1. Now these tokens, which are in the vocabulary of the BERT tokenizer, are converted to indicies, which correspond to index of each individual embedding inside BERT. These embeddings are passed through multiple transformer layers inside of BERT, resulting in a sequence of hidden representations which can be interpreted as the context embeddings $[h_1, \ldots, h_T] = [x_1, \ldots, x_T]$.

Other language models

Although we will only be working with the BERT language model in the subsequent sections, there is a wide variety of other popular language models.

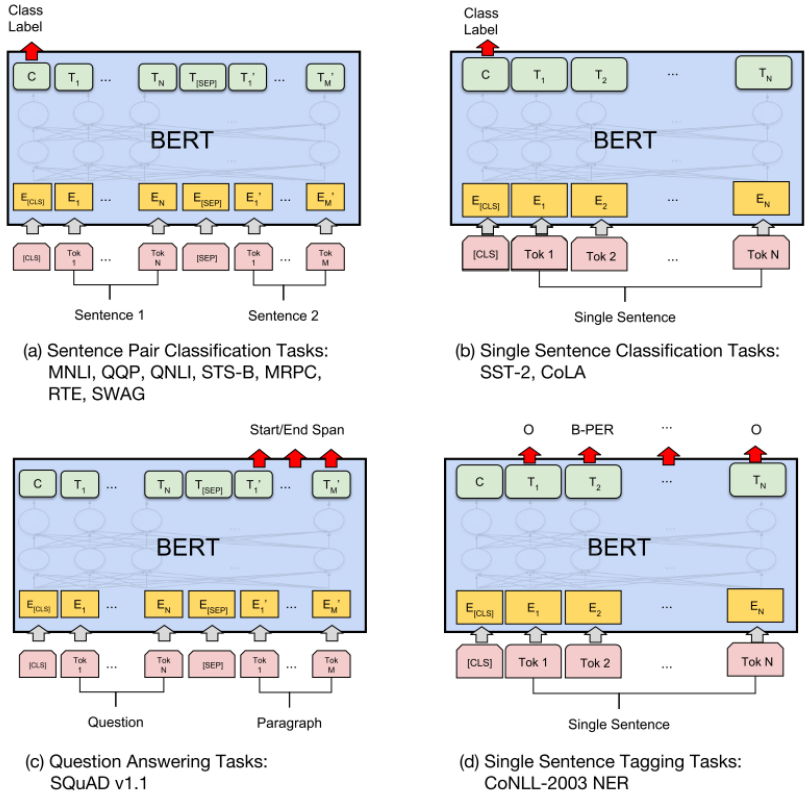


Figure 2.7: Figure taken from [36]. Ways to fine-tune BERT on different GLUE tasks.

These include GPT [98] and GPT-2 [99] which are also based on transformers, however only allows the sequence modeling to be forward looking (in contrast to BERT, which also allows the future context to take into the calculation when creating the context embedding). These also include more fundamental modifications to the transformer, including the introduction of layer normalization [10] layer to the input of each sub-block and an additional layer normalization after the final self-attention block. Other language models which extent BERT models include ALBERT [66] - which is widely considered one of the state-of-the-art transformer models, DisilBERT [107], HUBERT [86], ERNIE [118], RoBERTa [71], SpanBERT [60] and StructBERT [131] and also compressing representations produces by BERT [109]. Other lines of work [127] also interpret BERT as a markov random field,

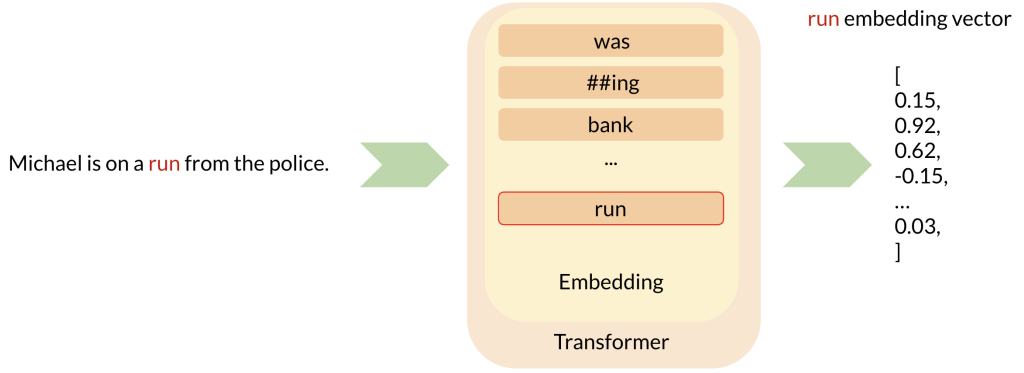


Figure 2.8: The BERT model takes as input a sentence s. The sentence s is converted to a sequence of BERT tokens $t_1, \ldots, t_m$ as defined in a given vocabulary V. Each item in the vocabulary V has a corresponding embedding vector inside the embedding layer of the transformer. This embedding vector is used by the intermediate layers of the transformer, and thus affects the downstream pipeline of the transformer for any subsequent layers of the transformer.

formalizing the process of sampling sentences.

2.3 GLUE benchmark dataset

The GLUE benchmark dataset was first introduced by [129]. We will be using the standard GLUE benchmarking, as BERT does not out-perform human-level scores on this dataset. Please note that most state-of-the-art models achieve human-level accuracies for standard GLUE benchmarks, which is the reason [128] introduced SuperGLUE with even more advanced language tasks.

We use an accuracy score for the majority of benchmarks. Because MRPC and QQP have unbalanced classes, the F1-score is used here. Finally, performance on STS-B is measured using pearson and spearman correlations, as the task output is a ranking. CoLa is assessed using the Matthews correlation. Because in later experiments, the Inference datasets are the most interesting tasks for us, we will give a short outline of these benchmarking tasks. For a description on all other GLUE tasks, please refer to section A.4

Corpus	|Train|	|Test|	Task	Domain
CoLA	8.5k	1k	acceptability	misc
SST-2	67k	1.8k	sentiment	movie reviews
MRPC	3.7k	1.7k	paraphrase	news
STS-B	7k	1.4k	sentence similarity	misc.
QQP	364k	391k	paraphrase	social QA questions
MNLI	393k	20k	NLI	misc
QNLI	105k	5.4k	QA/NLI	Wikipedia
RTE	2.5k	3k	NLI	news, Wikipedia
WNLI	634	146	coreference/NLI	fiction books

Figure 2.9: Table taken from [129]. Listing of all GLUE tasks including the linguistic phenomenon benchmarked, as well as the domain that the benchmarking data contains. Training and test set sizes are also provided.

in the Appendix.

Inference Tasks

The Multi-Genre Natural Language Inference Corpus **MNLI** [137] [20] is a crowd-sourced collection of sentence pairs with extual entailment annotations. For each premise and hypothesis sentence, the task is to predict whether the premise entails the hypothesis, contradicts the hypothesis, or neither. This is a 3-class classification problem. The matched MNLI version tests on data which is in the same domain as the training set, while the mis-matched MLNI tests on a training set which is cross-domain. The MNLI contains parse-trees, which is the reason we do not display these here. We will be working with the matched MNLI task only for reasons of convenience.

The Stanford Question Answering Dataset is a question-answering dataset [100]. The authors of GLUE augment this dataset and create the Question answering NLI **QNLI** benchmark, by relaxing the requirements that the model selects the exact answer, as well as the simplifying assumption that the answer is always present in the input and that lexical overlap is a reliable cue. The task is to determine whether a context sentence contains the answer to the initially posed question.

```
Who did the children work beside?
In many cases, men worked from home.
not_entailment
```

```
How many alumni does Olin Business School have worldwide?
Olin has a network of more than 16,000 alumni worldwide.
entailment
```

The Recognizing Textual Entailment **RTE** consists of a series of annual textual entailment challenges. Data is combined from RTE1, RTE2, RTE3 and RTE5 [34] [48] [15] [45]. This is a classification task where the inputs are two sentences, and the models task is to predict one of the possible outputs *neutral*, *contradiction* and *no entailment*.

```
Oil prices fall back as Yukos oil threat lifted
Oil prices rise.
not_entailment
```

```
Money raised from the sale will go into a trust for Hepburn's family.
Proceeds go to Hepburn's family.
entailment
```

The Winograd NLI **WNLI** dataset uses the original Winograd Schema Challenge dataset [68], which is a reading comprehension task where the reader must read a sentence with a pronoun and select the referent of that pronoun from a list of choices. Sentence pairs are constructed by replacing the ambiguous pronoun with each possible referent. The task is to predict if the sentence with the pronoun substituted is entailed by the original sentence. The dataset includes adversarial examples which test negatively when overfitted.

Some example sentences include

```
Bob was playing cards with Adam and was way ahead.
If Adam hadn't had a sudden run of good luck, he would have won.
Adam would have won.
label:0
```

```
Mark told Pete many lies about himself, which Pete included in his book.
He should have been more truthful.
Mark should have been more truthful.
label:1
```

2.4 WordNet

The online lexical database WordNet was original introduced in [83]. WordNet is a semantic reference system similar to a thesaurus whose design is inspired by current psycholinguistic theories of human lexical memory. English nouns, verbs and adjectives are organized into synonym sets, each representing one underlying lexical concept. Different relations link the synonym sets. WordNet 1 contains of 95,600 different word forms (51,500 simple words and 44,100 collocations), and include a total of 70,100 word meanings, or sets of synonyms. Compared to a standard dictionary, WordNet divides the lexicon into five categories, namely nouns, verbs, adjectives, adverbs and function words. The authors argue that the rest of language is probably stored separately as part of the syntactic component of language. The most ambitious feature of WordNet, however, is to attempt to organize lexical information in terms of semantics (word meaning).

The problem with alphabetical thesaurus is redundant entries. If Word w_a and word w_b are synonyms, the pair should be entered twice. The problem with a topical thesaurus is that two look-ups are required, first on an alphabetical list and again in the thesaurus proper. WordNet addresses this shortcoming through the concept of a *lexical matrix*.

Lexical matrix: The expression *Word* are often referred to both the utterance and to its associated concept. [83] specifies the difference between the two concepts as "word form", which refers to the physical utterance or inscription, and "word meaning", which refers to the lexicalized concept that a form can be used to express. We will refer to *semantics* whenever we mean *word meaning*.

The following table captures the concept of a lexical matrix, which encodes word-forms (columns), word-meanings (rows), and the existence of the possibility to express the word-meanings through the corresponding word-form (cell). If multiple entries exist for a single column, then the single word-form encode multiple meanings, implying that the word-form is polysemous (it encodes multiple word-forms). If multiple entries exist for a single row, the two words express the same underlying concept, and thus the two words-forms are synonyms.

Word	Word Forms				
Meanings	F_1	F_2	F_3	...	F_n
M_1	$E_{1,1}$	$E_{1,2}$			
M_2		$E_{2,2}$			
M_3			$E_{3,3}$		
$\vdots$				$\ddots$	
M_m					$E_{m,n}$

Figure 2.10: Table taken from [83]. Word-forms F_1, and F_2 are synonyms of each other, as they share one word meaning M_1. Word-form F_2, as it entails more than one meaning, namely M_1 and M_2.

WordNet also includes synonym sets referred to as *synsets*, which are a collection of word-forms that together determine a single meaning. WordNet represents a semantic unit through such synsets. WordNet is organized by semantic relations. Thus, WordNet includes a set of semantic relations. WordNet defines synonyms as a set of words, where one word is interchangable for another. Because words in a synsets are interchangable due to their synonymous nature, WordNet organizes words into nouns, verbs, adjectives and adverbs, as the syntactic rules of language must also stay conform.

Although the authors mention that synonyms should be best thought of a continuum along which similarity of meaning can be graded, the authors decide through the introduction of synsets to determine similarity in terms of a binary event, something which is either present, or not present. The two

Part of Speech	Definition
noun	(sloping land (especially the slope beside a body of water)) "they pulled the canoe up on the bank"; "he sat on the bank of the river and watched the currents"
noun	depository financial institution, bank, banking concern, banking company (a financial institution that accepts deposits and channels the money into lending activities) "he cashed a check at the bank"; "that bank holds the mortgage on my home"
noun	(an arrangement of similar objects in a row or in tiers) "he operated a bank of switches"
verb	(tip laterally) "the pilot had to bank the aircraft"
verb	(do business with a bank or keep an account at a bank) "Where do you bank in this town?"

Figure 2.11: Example output for WordNet 3.1 noun propositions for the word `bank`. In total, 18 different concepts are recorded.

tables 2.11 and **??** show examples of how WordNet 3.1 introduces different semantic classes for the words `bank` and `was`

We will make frequent use of WordNet, as it allows us to quantify semantics amongst word-forms.

2.4.1 SemCor dataset

The SemCor 3.0 corpus is an aggregation of the various Brown corpora [40], where each noun, adjective and verb is tagged with the WordNet 3.0 senses. It was part of the early WordNet project, initially introduced in [85]. We provide some examples to give an idea of what the SemCor dataset looks like. Looking at the example sentence

```
A Texas halfback who does n't even know the team 's plays,
 Eldon_Moritz, ranks fourth in Soutwest_conference scoring
 after three games.
```

The corresponding part-of-speech labels are

```
DT, NN, NN, WP, VBZ, RB, RB, VB, DT, NN,
POS, NN, NNP, VB, JJ, IN, NNP, VP, IN, JJ, NN
```

Part of Speech	Definition
noun	Washington, Evergreen State, WA, Wash. (a state in northwestern United States on the Pacific)
verb	(have the quality of being; (copula, used with an adjective or a predicate noun)) "John is rich"; "This is not a good answer" bank"; "that bank holds the mortgage on my home"
verb	(an arrangement of similar objects in a row or in tiers) "he operated a bank of switches"
verb	(form or compose) "This money is my only income"; "The stone wall was the backdrop for the performance"; "These constitute my entire belonging"; "The children made up the chorus"; "This sum represents my entire income for a year"; "These few men comprise his entire army"
verb	(work in a specific place, with a specific subject, or in a specific function) "He is a herpetologist"; "She is our resident philosopher"

Figure 2.12: Example output for WordNet 3.1 noun propositions for the word `was`. In total, 18 different concepts are recorded.

with the corresponding WordNet semantic class ids

```
None, Texas.1, halfback.1, None, does.0, not.1, even.7, know, None, team.1, i
None, ranks, fourth, in, 3, 1, 1, 1, scoring, None, three, 1, after.1, three.
```

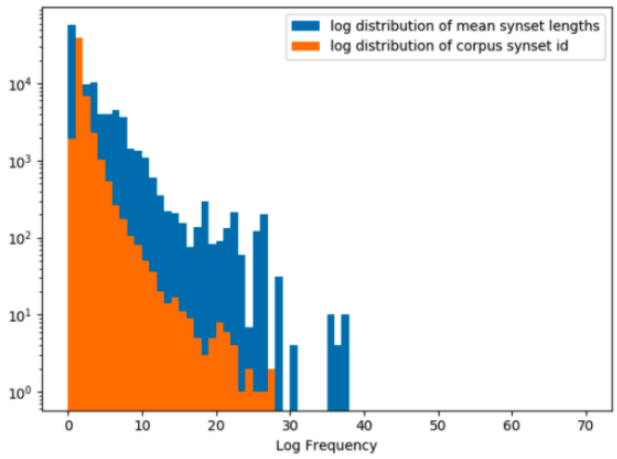


Figure 2.13: Shows that the SemCor data is biased towards words with low WordNet class IDs. Words with a low WordNet sense index (i.e. close to 0) occur much more often than words that have a high WordNet sense index (i.e. above 5). The x-axis shows the WordNet sense index for a chosen word, while the y-axis shows the log-frequency within SemCor. This is a cumulative plot over all words with WordNet senses within SemCor 3.0. The skew could be a natural effect of how word lower WordNet indecies are assigned to more commonly used words.

Chapter 3

Related Work

3.1 Clustering for Synsets in Static Word Embeddings

One of our goals is to identify semantic clusters within a given set of context embeddings. Implicitly, this requires the algorithm to solve a multi-modality detection problem, which in itself is hard. This implies that we cannot use algorithms such as k-means [72, 74], which requires the number of clusters to be known beforehand. Instead, we turn our attention to algorithms, which both find clustering assignments and also the best number of clusters that the data can be clustered by. Although we could also include more complex models such as autoencoders or other neural network based algorithms in this analysis, we decide to focus on common clustering algorithms outside the domain of neural networks. Please refer to section B.1 in the Appendix for an intuitive description of how of the algorithms we use work, as we assume that the reader knows of algorithms such as DBScan, Affinity propagation and MeanShit.

The Chinese whispers algorithm , is used to cluster for synsets is described in [91]. The authors create an EGO-network for each concept in the vector space whose correlation matrix $G(X)$ is the adjacency matrix for a

resulting subgraph $\tilde{G}$

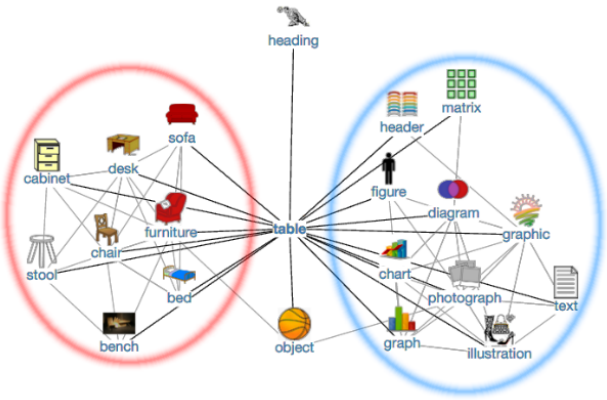


Figure 3.1: Figure taken from [91]. An EGO-network of the word `table` is created. Then, clustering is applied on the EGO network, to identify different semantic sets *synsets*.

The authors propose a multi-step approach, where they (1) learn word embeddings, (2) build a graph of nearest neighbours based on vector similarities, (3) apply induction of word senses using ego-network clustering and (4) aggregate the word-vectors with respect to the induced senses. The authors use the static word2vec word embeddings as the starting step. A graph is created by only keeping the N nodes with highest weight (deenoted by a similarity value). Clustering is done using the Chinese whispers algorithm. The result of this clustering conceptually corresponds to the synsets captured by WordNet sense classes. However, the underlying limitation lies in the choice of the word-vectors, as well as the graph-construction and graph-clustering.

Table 3.1: Table taken from [91]. Neighbors of the word `table` and its senses produced. The first row belongs to both senses, while the second and third row are distinct synsets.

Other similar work includes [9].

We will use a modified version of the Chinese Whispers algorithm. Because we are only given a set of context vectors organized in a matrix X, we must first generate a graph from the given set of vectors. Our modified version of the Chinese whispers looks as follows:

Vector	Nearest Neighbours
table	tray, bottom, diagram, bucket, brackets, stack, basket, list, parenthesis, cup, trays, pile, playfield, bracket, pot, drop-down, cue, plate
table# 0	leftmost# 0 , column# 1 , randomly# 0 , tableau #1 , top-left# 0, indent# 1, bracket# 3, pointer# 0 , footer# 1 , cursor# 1 , diagram# 0 , grid# 0
table# 1	pile# 1, stool# 1, tray# 0, basket# 0, bowl# 1, bucket# 0, box# 0, cage# 0, saucer# 3, mirror# 1, birdcage# 0, hole# 0, pan# 1, lid# 0

Algorithm 1: Checks sampled BERT vectors for clusters by meaning

Input: X: The set of context embeddings for a word w included in a set of n sentences. These are the BERT vectors we want to cluster;

Result: Cluster assignments for each of the samples within X.

corMatrix = cosineSimilarity(X, X)

closestNeighbors = argsort(corMatrix, axis=1)[:, :n]

corMatrix[not in closestNeighbors] = 0.

corMatrix[identifyHubs(corMatrix)] = 0.

clusters = runChineseWhispersOnGraph(corMatrix)

clusters = extrapolateHubsByNearestNeighbors(clusters)

return clusters

Intuitively we create a an adjacency matrix `corMatr` for a graph G from the set of vectors X using the following algorithm, which is then passed on for clustering to the `RunChineseWhispersOnGraph` algorithm using the NetworkX implementation [47].

To addresses the hubness problem mentioned in [33] and because we also see performance improvements in this, we identify hubs by the following cutoff formula.

$$\texttt{cutoff}(x) = \mu(X) + c\sigma(X) \tag{3.1}$$

where c is an adjustable hyperparameter. These hubs are removed if the connectivity value is too high.

Similar work has been done with the JoBimText framework [**?**], which provides a graph-based, sparse word similarity model. [**?**] average the word vectors of synsets to arrive at vectors which described semantics. Synsets are calculated using the same clustering approach which is applied on ego-graphs of the network as is done in [17].

3.2 Quantifying word sense disambiguation

Datasets have been proposed to benchmark word sense disambiguation performance for static word embeddings[**?**, **?**]. [95] address the problem of a missing benchmark dataset for *context embeddings* to test context embeddings for word sense disambiguation tasks. The authors introduce the **Word in Context WiC** dataset of labelled sentence pairs, where the language model is supposed to learn if the different semantic classes of a target word w.

An example of this dataset includes (the first one with the target word `bed`, the second one with a target word `window`:

```
There's a lot of trash on the bed of the river.
I keep a glass of water next to my bed when I sleep.
label:False
The expanded window will give us time to catch the thieves.
You have a two-hour window of clear weather to finish working on the lawn.
label:True
```

A more extensive list of datasets separated by knowledge-based approaches (i.e. making use of external resources like WordNet) and distributional-based approaches (i.e. using merely the distributional assumption of language) can be found in [88]. [35] is another survey which includes clustering of 3-grams, as well as graph-based approaches, where a graph is created in such a way that nouns are nodes, and edges between nodes exist whenever two words occur in the same paragraph.
Similar lines of work exist where clustering is applied on topic-dependent arguments [102] to improve data mining and document classification perfor-

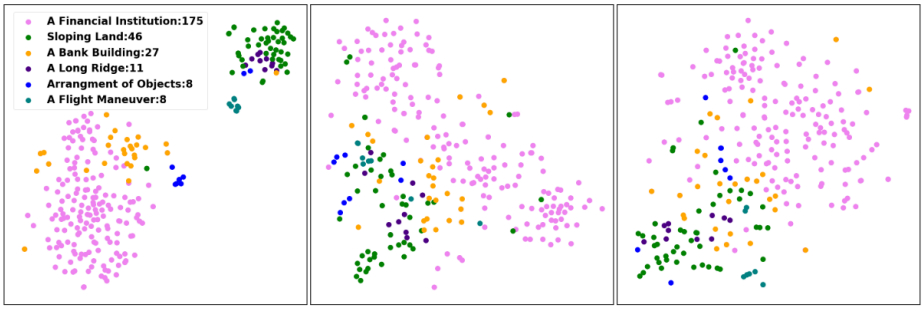
mance.

3.3 Semantic subspace inside BERT

Due to the different ways semantics can be quantified, different papers show mixed results in extracting a semantic subspace inside of BERT.
[79] analyses the structure inside BERT, and show that there is no strong correlation between the semantic feature of a word, and the location of the context embedding produced by BERT. They go so far as to say that the position i of the word w_i inside the sequence $s = [w_1, \ldots, w_i, \ldots, w_T]$ has a stronger implication on the position of the produced context embedding than the word token does. This is strongly manifested if one looks at context embeddings for words w_i where i is odd vs. even. It is also mentioned that most papers that do an active analysis on this topic do not go enough into depth on the topic of the theory of meaning, but rather are trying to identify common patterns in modern language models. The paper analysis these properties by conducting three experiments. The first one is about *word type cohesion*, which tests if similar words are projected into similar points in the context embedding space. They analyse if words with the same WordNet semantic class are projected to the same space inside BERT when sampled in different contexts. The authors make use of the silhouette score [**?**], and a clustering scheme of words - where a cluster is defined by words in the same semantic classes - and use the silhouette score as an argument to show that similar words are not projected to similar points in the context space. The authors use the Wiktionary dictionary [**?**] as an alternative to semantic classes defined by WordNet. The authors conclude that polysemous words result in degraded cohesion scores and decreased silhouette scores. Of course this does not imply that this global organization of word-vectors has a negative effect on downstream tasks, which is also visible in the strong success of transformer-based language models. Finally, the authors argue that the cosine similarity is a good distance measure for semantic similarity between embedding vectors. The authors continue with a sentence-pair similarity analysis, as well an analysis of the sentence-level

structure, which is not of strong interest for our investigation.

In slight contrast to the conclusion of the above work, [136] analyses to what extent context embeddings are better at word sense disambiguation tasks. The authors use a nearest neighbour classifier to measure the performacen on for two standard WSD benchmark datasets. The authors show that BERT is able to put polysemic words into distinct 'sense' regions of the embeddings space, while ELMo and other models do not seem to possess this ability. The authors argue that for static word vectors, different word senses are collapsed into the same string representation. The number of senses can be defined by a given parameter, or derived automatically through a non-parametric estimation. Prior work on using recurrent neural networks to classify sense labels is existent The authors argue that because for most NLP tasks, the fact that context emebddings perform better implies that they also capture polysemy much differently. The authors look at different contextual language models, including BERT and ELMo. The distributional hypothesis includes that if the same word regularly occurs in different, distinct contextes, we may assume polysemy of its meaning [84]. In their experiment, the authors use as the word embedding the concatenation of the averaged wordpiece vectors of the last four layers.



(a) BERT (b) Flair (c) ELMo

Figure 3.2: From [136]. T-SNE plots of different senses of `bank` and their context embeddings. The legend shows a short description of the different WordNet sensees and the frequency of occurrence in the training data.

The authors conclude that WSD can be surprisingly effective using solely

context embeddings. Different SensEval datasets are used for benchmarking, which also including SemCor.

[39] investigates the question of whether context embeddings for a given word w cover a closed area in the embedding space produced by BERT, or if there are infinitely many context-specific representations for each such word. The authors analyse ELMo, BERT and GPT-2. The authors argue that in all models, context embeddings are not isotropic, and thus not uniformly distributed with respect to direction, and are instead anisotropic, occupying a narrow cone in the vector space. In GPT-2 alone, two random words will almost have perfect cosine-similarity. For BERT, words in the same sentence grow more and more dissimilar in upper layers, but stay more similar to each other than randomly sampled vectors. The authors use the SemCor corpus to calculate the following measures as indication for properties of how a semantic subspace is organized within BERT:

- *self-similarity* of a word w in layer l is the average cosine similarity of word w across n different contexts as expressed by

$$\text{SelfSim}_\ell(w) = \frac{1}{n^2 - n} \sum_j \sum_{k \neq j} \cos\left(f_\ell\left(s_j, i_j\right), f_\ell\left(s_k, i_k\right)\right)$$

- *intra-sentence similarity* is the similarity between the word-vector and the sentence-embedding, which is the mean of the token-embeddings for that sentence. Formally

$$\text{Intrasim}_\ell(s) = \frac{1}{n} \sum_i \cos\left(\vec{s}_\ell, f_\ell(s, i)\right)$$
$$\text{where } \vec{s}_\ell = \frac{1}{n} \sum_i f_\ell(s, i)$$

- *maximum explainable variance* is proportional to the variance in ws contextualized representations for a given layer. It gives us an upper bound on how well a static embedding could replace word's contextu-

alized representations and is calculated by

$$\mathrm{MEV}_\ell(w) = \frac{\sigma_1^2}{\sum_i \sigma_i^2}$$

where $\sigma_1, \ldots, \sigma_m$ are the first m singular values of the occurrence matrix $[f_\ell(s_1, i_1) \ldots f_\ell(s_n, i_n)]$ where $f_l(s, i)$ maps word $w = s[i]$ in sentence s to a context embedding at layer l of the language model.

If both intra-similarity and self-similarity are low, then the model contextuaizes words in that layer by giving each one a context-specific representation that is still distinct from all other word representations. If the self-similarity is low, this suggests a less nuances contextualized. The authors conclude that all the language models considered are anisotropic in their sampled context embeddings. Due to this anisotropic property, the evaluations needs to be adjusted for anisotropy, which can be done by subtracting the mean value for each formula across all words and at layer l. For BERT, the average cosine similarity between uniformly randomly sampled words is between 0.2 and 0.6, increasing with the number of layers. This value would be close to 0. in a space where embeddings are isotropically / uniformly distributed across the full vectorspace. The authors argue that this is inherent to the process contextualization. The authors further show that stopwords, such as `the`, `of` and `to` have the lowest self-similarity, implying that these have the most context-specific representations. Although the authors argue that these words are not polysemous, it is apparent that these vectors are the ones with highest number of different contexts. The authors argue that the language models do not simply assign one of a finite number of word-sense representations to each word, as this would lead to less variation in the representation. Finally, the authors take the first principal components from a set of sampled word in different contexts as a static word embedding. They show that for layers 1 and 2 of BERT, these word-embeddings often outperform word sense disambiguation benchmarks.

[24] concludes a survey on vector representations of meanings. The authors present different sense representation exploiting monolingual corpora,

which apply clustering-based models to extract senses from text, or joint training methodologies, which perform induction and representation learning together. The paper also introduces learned topic embeddings and word-embeddings that are separately learned, jointly or subsequently. The different types of models include: (1) Dynamic polysemy, which is a direct solution to the varying problem of sense representation models for polysemous words, which would be to set the number of senses of a word as defined by an external sense inventory such as WordNet. Often, heuristics are used, such as that the frequency of a model is correlated with polysemy, as - according to the authors - more frequent words are probably more polysemous. (2) Pure sense-based models include which implicitly learn ambiguity of in context words. The authors re-iterate that about 80% of all words in WordNet 3.0 are monosemous, with less than 5% having more than 3 senses.

[70] create a thesaurus by through a co-occurrence-based approach to select a list of first-order candidates, and a list of second-order candidates by using a distributional-based approach. However, their approach makes strong use of part-of-speech taggers to disambiguate between words-senses by the presumed correlation to the PoS tag.
[110] create a relation extraction and semantic role labelling model using BERT as the underlying model. The authors extend the BERT Model with a simple linear layer on top of the last hidden layer. The authors use the CoNLL 2005 [26] and the 2012 dataset [96] for labelled training. The authors are able to supersede prior non-BERT models with this simple BERT extension for the tasks of semantic role labelling.

Similarly, [69] revise a modification to BERT resulting in the *SenseBERT* model which predicts not only the word-form of masked words, but also the WordNet supersenses. The authors use the Word in Context benchmark as the benchmark. A linear layer is appended above the final layer of BERT, making use of the final hidden representation. Training is done in a supervised fashion using the SemCor dataseet and the WiC tasks.

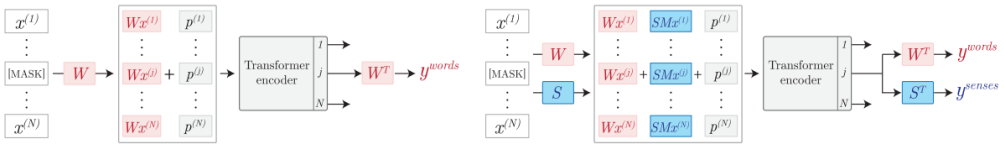


Figure 3.3: Figure from [69]. Standard BERT (left) and the SenseBERT adaptations (right), which include an embedding-encoder and an embedding-decoder specifically for WordNet senses.

The authors record an up to 12% gain in accuracy for in the SemEval-SS task (predicting SemCor WordNet labels), and up to 12% improvement for the Word in Context task. Even withouth fine-tuning, SenseBERT achieves competitive results with fine-tuned BERT. The authors show that globally, these vectors seem to be properly aligned by their senses, even though they do not formally quantify this statement.

A derivate to linguistic semantic concepts are semantic concepts specialized to a certain domain. [112] use context embeddings to better extract clinic concepts. This is a task similar to extract semantic meanings, with the differrence that meanings consists of medical concepts only, rather than the more non-domain-specific WordNet classes. Besides using SemCor, the authors also focus on medical datasets annotated with clinical concepts and amongst others, making use of BioBERT [**?**]. The authors demonstrate state-of-the-art results for clinical concept extraction.

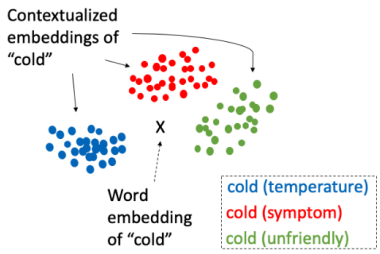


Figure 3.4: Figure taken from [112]. **Fictional** embedding vector points and clusters of `cold`. This is one of the results that we want to arrive at. Specifically, we desire distinct word-embeddings that capture the modes of the underlying probability distribution.

Figure 3.5: Figure taken from [112]. PCA visualizations using embedding vector of `cold` from BERT. The blue data points refer to `cold` as a temperature, whereas the red data points refer to `cold` as a symptom.

3.4 Syntactic subspace in BERT

Next to semantical features, BERT can also be analysed for syntactical properties in the output it produces.

[104] aims to devise a verb-clustering, and an argument-clustering approach which is supposed to devise a clustering that evokes frame-specific slots and thus semantic roles. The authors again make use of the Chinese whispers algorithm [17] to arrive at the semantic frame cluster. The authors use of ELMo and BERT word embeddings. For the chinese whispers algorithm again, a graph is devised from the set of sampled context embeddings X. Edges all have the same weight, thus a cutoff function is defined by which

edges of lower weight (and thus lower similarity) are removed The cutoff function for the verb-clustering task is

$$t_f = \frac{\mu + \sigma}{2} \tag{3.2}$$

where μ and σ are the mean and standard deviation of the pairwise distance distribution, respectively. For argument clustering, the cutoff function used is defined as

$$t_a = \mu - 1.5\sigma \tag{3.3}$$

as a higher neighboring threshold is needed as the authors empirically find out. Clustering performance is measured by the number of clusters found, and a purity metric which describes how homogenous the created clusters are. Labels from FrameNet [12] are used for the homogenuity test.

Next to clustering, [31] analyse how BERT and similar transformer models capture syntactic parse-trees. The authors argue that on a high level, semantic and syntactic features seem to be present in different subspaces. Although the authors mention that there is fine-grained geometric representations of word senses, which is encoded in a relatively low-dimensional subspace. This conclusion stems mostly from qualitative evaluation. The authors use the Penn Treebank [**?**] to infer a dependency parsing scheme using a linear classifier with output of the transformer context vectors as input. The authors achieve above 85.8% accuracy for two-class classification, and 71.5% accuracy for multi-class classification. In a prior work by [51] find that BERT encodes a parse tree where the tree distance seems to correspond to specifically to the square of the Euclidean distance. The authors argue using a theorem that shows that a tree with n nodes has a Pythagorean embedding in $\mathbb{R}^{n-1}$.

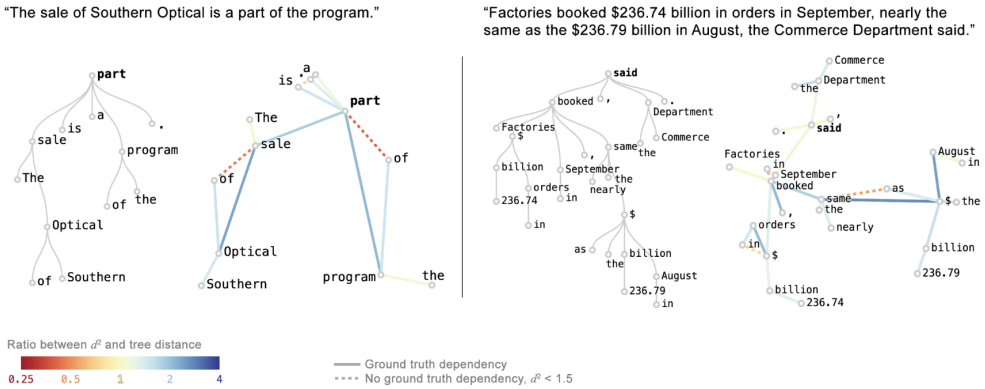


Figure 3.6: From [31]. Visualizing the embeddings of two sentences after applying the Hewitt Manning probe. Parse tree (left) in comparison to the PCA projection of the context embeddings (right). Small deviations are apparently, but an obvious resemblance exists. The color of an edge determines the squared Euclidean distance.

The authors also analyse semantic features qualitatively. However, it is important to realize that the qualitative evaluation is not extensive, and only considers the linguistic feature of *plurality*, and only on hand-picked examples. Also, it is important to notice that the semantic division between does not rule out a division between part-of-speech tags, as `die` have different meanings when interpreted as a verb and as a noun, which could also be the reason the clusters are so strongly visible.

Figure 3.7: From [31]. Embeddings for the word `die` in different contexts, visualized through UMAP. The blue text describes general annotations.

A nearest neighbour classifier is built using the BERT embeddings resulting from the labelled SemCor corpus, achieving state-of-the-art results on this particular word-sense-disambiguation tasks. The authors try to find a linear transformation matrix under which the word-sense-disambiguation task performs even better, however this results in only marginal benefit. The underlying assumption is that the semantic features are encoded in an underlying subspace of the BERT context vectors. The authors also note that this suggests that more semantic information may be available in lower layers.

[94] analyses the choice of different neural networks (i.e. LSTM, CNN or self-attention) and the effect it has on the underlying syntactical subspace produced.

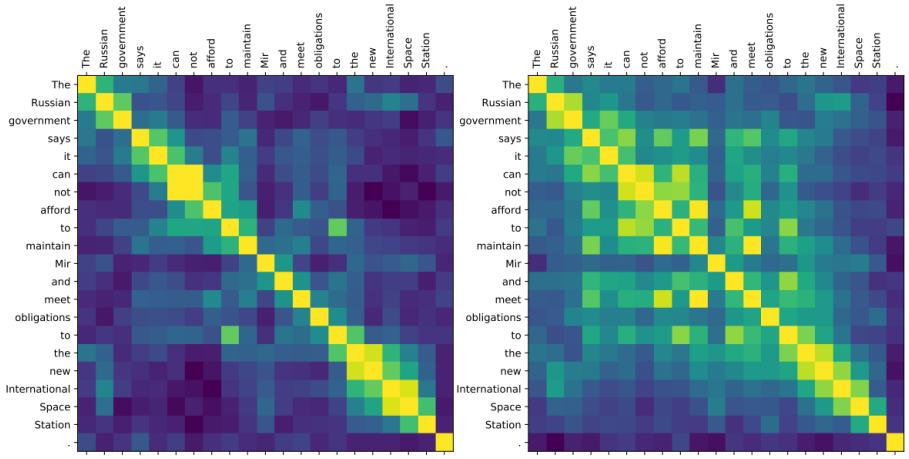


Figure 3.8: From [94]. The authors show different similarity intensities (cosine similarity) between token-pairs using the output of a 4-layer LSTM (left), and the output after the first layer (right).

The authors argue that in general, lower layers mostly capture local information, while top layers represent longer range relationships, as the attention matrix has local spikes. For biLMs, lower layers tend to place words from the same syntactic constituents in similar parts of the vector space. The authors argue that in biLMs, semantics is captured only to a small extent, whereas morphology alone is most often used to answer questions on word analogy tasks. The biLM architectures learn syntax, as PoS tags are captured at lower levels of the biLM layers. The authors further build a contstituency parser with almost 80% accuracy, by only adding a linear classifier and the span representation onto the lower layers of the biLMs. A parse-tree is built using a greedy decoding step similar to [61].
Other lines of work includes models such as *PAFIBERT* [120], which aims to transform sentences into a predicate-argument structure. Implicitly, this solves a frame identification, as well as semantic role assignment task.

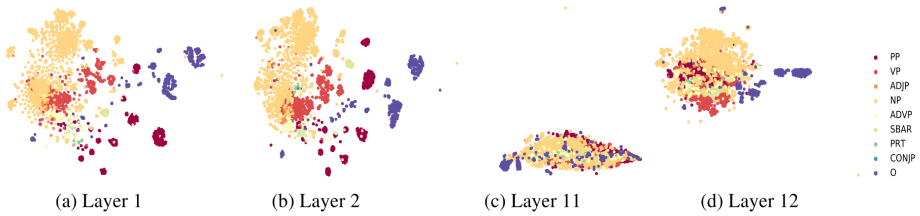
3.5 Static embeddings from context embeddings

The general idea behind distilling static embeddings from context embeddings is that one can arrive at a (1) compressed representation of the context embedding, and (2) capture polysemous concepts through multiple static embeddings rather than one embedding per token. We present a few ways static embeddings can be distilled from BERT.
This includes distilling knowledge or minimizing BERT to smaller neural network models [121], [122], and most famously [107]. Compressed sentence representations can also be learned i.e. through random projections or architectures similar to autoencoders [109], amongst others, making use of a max-pool operator over sentence-tokens to arrive at a sentence embedding.

3.6 Sentiment-subspace in BERT

[77] Measure social biases in BERT context embeddings. For this, they use static word embeddings by averaging over token-embeddings produced by BERT. Although the authors cannot prove general trends, they show that certain suspicious patterns of sensitivity suggest that bias is occurring in different contexts. Specifically, this is done by identifying relationships between concepts and attributes. The authors identify that in the embedding space, concepts such as `European American names` are much more closer to attributes such as `Pleasant` than concepts such as `African American names`, which is a strong indication of bias. Similar work is done by [23], which focus much more on the corpora which are used for training. The authors analyse the social bias contained in semantics derived from language corpora, which is the case also for modern language models like BERT. The authors hint that word-embeddings like word2vec also contain biases, where male names are more biased with career settings than female names. Again, the example with european-american and african-american names is shown.

[86] Include a final linear transformation which mimics the distance learning aspect to filter out the subspace of interesting for downstream tasks. This final linear transformation is task-specific and not learned during pre-training. When we need to arrive a singular word representation from multiple tokens most work such as [19, 7] do a simple mean, max or min pooling over all word-embeddings of all tokens.
[58] argue that BERT encodes surface features in bottom layers, syntactic features in middle layers, and semantic features in top layers. Using normalized mutual information, the authors cluster the representations at different layers l of the model, and argue that lower layers encode phrasal information better. The authors use ten probing tasks to evaluate language features extracted through BERT.



(a) Layer 1 (b) Layer 2 (c) Layer 11 (d) Layer 12

Figure 3.9: From [58]. 2D t-SNE plot of span embeddings computed from the first and last two layers of BERT.

The authors also use a maximum spanning-tree approach to arrive at a dependency parsing tree, using the attention weights.

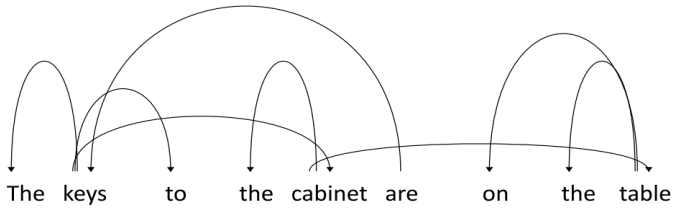


Figure 3.10: From [58]. Dependency parse tree induced from attention head in layer 11 in layer 2 using labelled root `are` as starting node with the maximum spanning tree algorithm

However, the authors do little work on semantics as defined by WordNet classes, but rather focus on part-of-speech, tenses and subject-verb agreement tasks.
[77] analyse the social biases that are part of the context embeddings and show that depending on the language model (ELMo, BERT), the amount of social bias deviates. Because context embeddings capture more than just semantics, bias is a natural implication of context vectors. A by-product of their research includes aggregating token-embeddings (i.e. aggregating the tokens *hav* and *###ing* to result at the word *having*). Although not very extensive on the topic of aggregation, the authors use techniques of mean-pooling, max-pooling, and last-pooling to determine arrive at a single context-vector, if the given token is intrinsically split-up by the language-model tokenizer (incl. BERT, ELMo, GPT).
Finally, evaluation on the corpora also yield inclusion of human-like biases [59] by names, races, male-vs-female. This bias can also be seen in how language changes over time.
[76] looks at how language models capture this changing semantics of language change over time. The authors train language models on a balanced set of genres, from the 1960s and the 1990s. The authors measure the variation coefficient of words. The authors cluster the word embedding spaces, and then calculate the drift measures through pearson and spearman correlations to arrive at a quantitative expression of how much language changes. Similarly, [55] analyse how meanings quantitatively using corpora from different historical epochs, including the 1890s, 1940s, 1960s, 1970s, 1990s and 2000s. Specifically, they measure the similarity scores for select context-vectors based on differently trained corpora. They demonstrate through examples of the word *gay* and *alien* that the similarity between certain context-vectors change through time.

3.7 Misc

[111] analyse properties of static word vectors by applying eigenvector analysis from Random Matrix Theory. The authors show that semantically

coherent groups not only form in the row space, but also in the column space, implying that individual word vector dimensions can be interpreted as human-interpretable semantic features. The authors make use of positive pointwise mutual information matrix, which is based on a log-ratio between the joint probability of a word w and its context words c. The authors apply a qualitative analysis by examining the top elements of the eigenvectors by sorting their absolute values in decreasing order. The authors analyse salient columns for words like airport. High absolute values imply that the word is relevant to the semantic group formed in the rowspace.

[135] follow a similar and more rigorous analysis with static word vectors. The authors argue that for standard word vectors, downstream tasks rely much more on local similarity rather than absolute positioning, and thus improving modern language models should focus explicitly on local geometric structures in sampling and evaluation methods.

Chapter 4

Understanding the semantic subspace organization in BERT

We begin our analysis by trying to understanding how BERT organizes its semantic subspace.
For this, we conduct three experiments. The experiments have varying difficulty for the algorithm to understand the subspace organization for a given word w. We start with a supervised algorithm which should build a discriminative model of the subspace, where labels are different semantic classes as defined by WordNet. The second experiment aims to identify a similar subspace organization through an unsupervised algorithm, implicitly identify a multimodal structure within the subspace at which we look at. Finally, we check how part-of-speech relates with semantics within the BERT model, hoping to simplify subsequent work by introducing the assumption that part of speech strongly correlates to semantics within the BERT model.

4.1 On the Linear Separability of meaning within sampled BERT vectors

The first experiment takes a word of interest w, samples a set of sentences S from a corpus (in this case, the above mentioned SemCor corpus and the news corpus [4] which does not have any WordNet annotated classes). To see

if there is any structure within BERT vectors w.r.t. the different meaning of one word, we ask ourselves whether or not different part of the meaning are at different locations of the embedding space produced by BERT. We test this hypothesis proactively by training a linear classifier which learns a separating hyperplane between the different WordNet classes for w.

4.1.1 Experiment setup

To produce a context embedding, we pass through a sequence of words $[w_1, \ldots, w_i, \ldots w_T]$ sampled from S, where $w_i = w$ (i.e. w_i coincides with the word of interest). The output of BERT is a set of hidden representations which are interpreted as the context embeddings $[x_{w_1}, \ldots, x_{w_i}, \ldots, x_{w_T}]$, where x_{w_i} corresponds to the context embedding of w_i. Repeating this n times, this results in a feature matrix X, where each row corresponds to the sampled sentence, and the columns correspond to the latent dimensions of the context embedding. In all of our experiments, we set $n = 500$, unless otherwise stated. However, there are cases (especially for the SemCor dataset), where we cannot find n number of sentence which include our word of interest w. In that case, we sample as many sentences as available in the corpus, making $n = min(S_{\text{available } w}, 500)$. We make sure however to conduct our investigation with words that occur often enough and take 30 occurrences per WordNet class as a lower cap. Because this experiment requires w to have multiple WordNet classes, we restrict the choice of w to be polysemous as defined by WordNet. Due to the restricted resources of SemCor, this limits our analysis to the following set of words. Although these words are not the most intuitive polysemous words (as would `bank` be, for example), the limited "obviousness" should allow this experiment to be stricter w.r.t. not rejecting the hypothesis.

Words used to test BERT-sampled vectors for linear separability of lexical features		
was	is	be
are	more	one
first	only	time

To arrive at uniform conditions for each sampled matrix of embedding vectors X, we apply a standard scalar such that the data is normalized around 0.0 with standard deviation 1.0. We also allow for dimensionality reduction using all sampled words. The reason for this is to have a stricter set of requirements for the separating hyperplane to be drawn. If the dimensionality of the data d is high, but the number of samples n is low, then the system of linear equations is underdetermined, as there is an infinite set of solutions possible to find a separating hyperplane. Thus, projecting X to a lower dimensionality works like a regularizer in that we restrict the set of hyperplanes to be drawn to be much more limited. On a separate note, this also tests if the semantic notion of the context vectors is kept at a lower dimension using a simple model such as PCA.

In summary, the experiment setup is captured by the following algorithm.

Algorithm 2: Checks sampled BERT vectors for linear interpretability by meaning

Input: A target word w_{target}; The latent dimensionality k for PCA;

Result: Accuracy of a logistic regression classifier

$\mathbf{D}, \mathbf{y} \leftarrow$ sample up to 500 sentences (as much as available) from the SemCor corpus which include the word w_{target}, along with the corresponding WordNet meaning-id;

$\mathbf{X} \leftarrow BERT(\mathbf{D})$ i.e. pass each sentence through BERT and retrieve the resulting word embedding x_w as defined in the above section;

$\mathbf{X}, \mathbf{y} \leftarrow oversample(\mathbf{X}, \mathbf{y})$ such that we don't have dominating classes;

$\mathbf{X}_{\text{train}}, \mathbf{X}_{\text{test}}, \mathbf{y}_{\text{train}}, \mathbf{y}_{\text{test}} \leftarrow trainTestSplit(\mathbf{X}, \mathbf{y}, testProportion = 0.4)$;

$\mathbf{X}_{\text{train}} \leftarrow StandardScaler(\mathbf{X}_{\text{train}})$ such that all the data is normalized;

$\mathbf{X}_{\text{train}} \leftarrow PCA(\mathbf{X}_{\text{train}}, k)$ such that all the data is projected to a lower latent dimensionality k;

$model \leftarrow LogisticRegression(\mathbf{X}_{\text{train}}, \mathbf{y}_{\text{train}})$;

$\hat{\mathbf{y}}_{\text{test}} \leftarrow model.predict(\mathbf{X}_{\text{test}})$;

$accuracy, confusionMatrix \leftarrow loss(\mathbf{y}_{\text{test}}, \hat{\mathbf{y}}_{\text{test}})$;

return $accuracy, confusionMatrix$;

We use standard sklearn [90] implementations for the Standard Scalar, PCA and Logistic Regression. We use the Hugginface transformers library [139] for

the this BERT model, and also all subsequent modification of BERT. The loss function we use is a binary simple $l1$ manhatten loss function, where the loss for a correct sample is 0, and for an incorrect class assignment is 1. We apply 5-fold cross validation, and measure the mean accuracy as well as the standard deviation of the accuracy. We also note the variance kept after projecting PCA on the lower dimensionality k to understand how a simple dimensionality reduction technique can affect the separation between semantic classes.

4.1.2 Results

We run the above experiment for a set of different k. The variance shown below is a fraction of 1.. A "variance kept" value of 1. corresponds to no information loss in terms of eigenvalues left out.
We show the result for 3 of the experiments, for the words *was*, *is* and *one* respectively, as these are the words that have the highest number of occurrences within the SemCor dataset.

Table 4.1: Mean and standard deviation of the accuracy of a linear classifier trained on the 2 most common classes of WordNet meanings for the word *was*.

dimensionality	variance kept	accuracy (mean / ± stddev)
10	0.27	0.82/±0.03
20	0.41	0.81/±0.04
30	0.50	0.85/±0.03
50	0.63	0.92/±0.03
75	0.73	0.94/±0.02
100	0.81	0.95/±0.02

Table 4.2: Mean and standard deviation of the accuracy of a linear classifier trained on the 2 most common classes of WordNet meanings for the word *is*.

dimensionality	variance kept	accuracy (mean / ± stddev)
2	0.09	0.57/±0.02
10	0.29	0.82/±0.03
20	0.42	0.82/±0.04
30	0.51	0.83/±0.03
50	0.72	0.85/±0.04
75	0.78	0.84/±0.04
100	0.79	0.85/±0.03

Table 4.3: Mean and standard deviation of the accuracy of a linear classifier trained on the 2 most common classes of WordNet meanings for the word *one*.

dimensionality	variance kept	accuracy (mean / ± stddev)
2	0.10	0.55/±0.10
3	0.14	0.51/±0.05
10	0.34	0.59/±0.08
20	0.50	0.76/±0.03
30	0.62	0.77/±0.02
50	0.76	0.83/±0.06
75	0.87	0.87/±0.05
100	0.94	0.87/±0.05

There are many permutations We now also employ multi-class classificiation.

Table 4.4: Mean and standard deviation of the accuracy of a linear classifier trained on the the 4 most common classes of WordNet meanings for the word *was*.

dimensionality	variance kept	accuracy (mean / ± stddev)
2	0.08	0.38/±0.03
3	0.11	0.38/±0.04
10	0.28	0.65/±0.03
20	0.43	0.76/±0.04
30	0.53	0.83/±0.03
50	0.67	0.93/±0.01
75	0.77	0.95/±0.01
100	0.83	0.95/±0.01

We get very similar accuracies for "time", "made", "thought". However, these tables are left out as we do not deem these to be statistically significant. To make sure that no inconsistencies happened during testing, we also analysed the confusion matrices, which all were balanced across class-pairs, as the transpose elements were usually off by only 1-4 samples, and the majority of the samples were correctly predicted (and thus on the diagonal of that matrix) .

The reader can see that for a learned hyperplane, the classification accuracy reaches more than 75% for dimensions of more than 20. This is also the case for multiclass classification problems, which removes the possibility that overfitting could have occurred for lower dimensions (due to the sheer amount of permutations that a hyperplane can take place in higher dimensions), which is also supported by the number of samples which is higher than the reduced dimensionality k.

4.2 On the Clusterability of meaning within sampled BERT vectors

Modality-detection can tell us a lot about the nature of a dataset. It can tell us what the predominant categories of a dataset are. Given a set of clusters

amongst vectors $x_1, \ldots, x_n$ organized in a matrix X, the different clusters can be interpreted as the different modes of the data. This partitioning can be interpreted similarly to what principal components are, and what different categories are possible for these categories.
There are different ways to check if a set of vectors is clusterable, this includes theoretical ways to determine clusterability, as for example depicted in [6, 78]. However, we are most interested in the practical clusterability of BERT, and as such, we decide to follow a brute-force approach to find clusters within BERT.
Because the experiment on linear separability has shown us, that different semantics are at different locations of the embedding space, we now want to investigate how "obvious" this clustering can be found, and whether a clustering can be found in the first place, or if the transitioning between the semantic categories is a smooth transition, or constitutes a hard partinioning amongst context vectors.
This experiment can be seen as an extension to the previous experiment. In the previous experiment, we try to train a discriminative model. In this experiment, we try to come up with a quasi-generative model which is able to detect modalities of the underlying true distribution, theoretically allowing us to generate new data samples.

4.2.1 Experiment setup

Initially, we tried to cluster the matrix of embedding vectors X using standard approaches with no hyperparameter optimization. However, all of the algorithms we manually tried, and with the default hyperparameters would fail and return -1 or similar messages, indicating that no clustering was found. We suspect that this is the case as the vectors produced by BERT are very smoothly distributed across the span of X.

As such, we had to adapt our experiment setup to be more sophisticated and implement a fast brute-force using random search and [16] and Bayesian optimization [133]. We use the Pytorch Adaptive Experimentation platform

[2] which automatically chooses which of the two policies to use, given the dimensionality and number of parameters to search for. Specifically, our experiment setup is precisely described by the following algorithm. We introduce two algorithm. First algorithm calculates the clustering overlap between the predicted clustering $\hat{y}$ and the true cluster labels as defined by the WordNet semantic class labels y. The input to this algorithm is the clustering model that we want to use, which is one of the algorithms introduced in section (3.1) and the chinese whispers algorithm introduced above, a possible dimensionality to which the embeddings are reduced to using a predefined dimensionality reduction algorithm such as PCA, or UMAP, and finally the target word w for which we want to calculate this overlap score.

Algorithm 3: Checks sampled BERT vectors for clusters by meaning

Input: w_{target}: The target word whose BERT vectors we want to cluster;
DimRed: The dimensionality reduction method;
k : The latent dimensionality for the dimensionality reduction method;
ClusterAlgorithm: The clustering algorithm to use;

Result: The associated cluster-id with each sampled sentence, and the adjusted random index.

$\mathbf{D}, \mathbf{y} \leftarrow$ sample up to 500 sentences from the SemCor corpus which include the word w_{target}, along with the corresponding WordNet meaning-id, and compensate more sentences by sampling from the news corpus if less than 500 sentences are available in the SemCor sentence. We set y = -1 whenever no labeling information is available, which is the case if we don't sample data from the SemCor corpus.;

$\mathbf{X} \leftarrow BERT(\mathbf{D})$ i.e. pass each sentence through BERT and retrieve the resulting word embedding x_w as defined in the above section;

$\mathbf{X}, \mathbf{y} \leftarrow oversample(\mathbf{X}, \mathbf{y})$ such that we don't have dominating classes (all except for $y = -1$).;

$\mathbf{X} \leftarrow StandardScaler(\mathbf{X})$ such that all the data is normalized;

$\mathbf{X} \leftarrow DimRed(\mathbf{X}, k)$ such that all the data is projected to a lower latent dimensionality k;

$model \leftarrow ClusterAlgorithm(\mathbf{X})$;

$\hat{\mathbf{y}} \leftarrow model.predict(\mathbf{X})$;

$score \leftarrow AdjustedRandomIndex(\hat{\mathbf{y}}[\mathbf{semcor}], \mathbf{y}[\mathbf{semcor}])$;

return $score, \hat{\mathbf{y}}$;

Running this over all possible clustering algorithms, and taking the mean across a set of words, we hope to get a reliable estimate of which clustering approach generalizes best across words. This is depicted in the following algorithm

Algorithm 4: Algorithm to find the best model with the best fitting parameter configuration.

Input: w_{target}: The target word whose BERT vectors we want to cluster;

Result: The model which has the highest adjusted random index score between its predicted clustering, and the true underlying clustering as given by the SemCor WordNet class labels.

```
for clusterModelClass in clusterModels:
    for modelConfiguration in clusterModelClass.hyperparNext() :
        for w, X, numberOfSenses, trueClustering, knownIndices
        in crossvalidationDataIterator:
            score = Algorithm2(w, *configuration)
            if score > maxScore:
                maxScore = score
                bestModel = clusterModelClass
                bestConfig = modelConfiguration
return maxScore, bestModel, bestConfig
```

Because SemCor only had limited resources, and often a certain number of samples (more than 10) must be provided per word for the score to be significant enough, we were again limited to choosing some of the most occurring words within SemCor. Due to the limited size of SemCor, we also decided to only use words where all of the WordNet classes are present.

This limits us to the following choices for the target words w, with number of WordNet classes in paranthesis next to it: `have` (20), `live` (19), `report` (13), `use` (13), `test` (13), `know` (12), `write` (10), `tell` (9), `state` (11), `limit` (9), `allow` (10), `enter` (9), `concern` (7), `learn` (6), `seek` (6), `final` (5), `central` (3), `critic` (3), `topic` (2), `obvious` (1), `kitchen` (1), `pizza` (1). Because one of the implicit goals of the clustering algorithm is detecting the number of modes, we uniformly pick words `was`, `thought`, `made`, `only`, `central`, `pizza`. Cross-validating over a different number of cluster classes scores our algorithm in the multi-modality detection aspect of the task. Due to the limited size of the SemCor dataset, we aggregate the matrix of context vectors X with sentences from

an unsupervised news corpus as well. Because we don't have labels for the unsupervised corpus, however, we only apply scoring on the samples for which we have the WordNet labels (i.e. sentences sampled from SemCor).

The scoring scheme we use is the random adjusted index *ARI* [101], [56] which compares to what extent two clusterings are identical. This is necessary, because clusteres across clustering episodes will not produce the same cluster labels when run repeatedly. The ARI is computed by going through all possible permutations of the cluster-labels, and normalizing the score of the match for each permutation by a normalizing value. Because all possible permutations are taken into account, the ARI is *adjusted for chance*.

$$ARI = \frac{\sum_{ij}\binom{n_{ij}}{} - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n}{2}} \tag{4.1}$$

where $n_{i,j}$ is the number of items that are present in both clustering assignment, $a_i = \sum_j n_{i,j}$, $b_j = \sum_i n_{i,j}$ for two clustering a and b.
The resulting score is between $[-1.0, 1.0]$, where a score of 0 implies completely assignment of cluster-labels into random buckets, -1.0 implies a negative correlation, and 1.0 implies perfect similarity.

4.2.2 Results

To keep the presentation of the results concise, I will be going over the most meaningful results. This is the case for when we set $k = 20$.
Although initially it was apparent that adding additional dimensions to the dimensionality reduction technique would improve the clustering performance we soon realized that introducing the chinese whispers algorithm would show that trimming down the dimensionality to $k = 20$ achieved the best results. We assume that this is the case as the 20 principal components

with biggest eigenvalues capture enough about semantics such that the rest of the vectors can be left out. To keep the discussion focused, please refer to section (C.1) to see the results for $k = [50, 100]$ for different clustering methods, exluding the adapted chinese whispers algorithm.
The next two tables show results where we keep the dimensionality at $k = 100$ and keep the sample size at $n = 500$ and $n = 10000$.

Table 4.5: The maximum ARI score achieved during hyperparameter optimization for the different models as described by experiment for $k = 100$ and $n = 500$.

Clustering Model	**ARI Score**
Affinity Propagation	0.168
Chinese Whispers	**0.298**
DBScan	0.201
HDBScan	0.242
MeanShift	0.167
Optics	0.167

Table 4.6: The maximum ARI score achieved during hyperparameter optimization for the different models as described by experiment for $k = 100$ and $n = 1000$.

Clustering Model	ARI Score
Affinity Propagation	0.170
Chinese Whispers	0.249
DBScan	**0.260**
HDBScan	0.234
MeanShift	0.167
Optics	0.197

We quickly realize (together with the results in the appendix) that the sample does not have much effect on the clustering performance. This is also the case because the corpora are exhaustive, and especially SemCor does not provide enough additional labels to make adding additional samples meaningful. Because of this, we change the sample size at either $n = 500$ and $n = 1000$ for subsequent trials.

Table 4.7: The maximum ARI score achieved during hyperparameter optimization for the different models as described by experiment for $k = 20$ and $n = 1000$.

Clustering Model	**ARI Score**
Affinity Propagation	0.165
Chinese Whispers	**0.349**
DBScan	0.167
HDBScan	0.273
MeanShift	0.226
Optics	0.167

So far, we had not added the `[SEP]` token to the sentence that BERT would input. The following table includes the `[SEP]` at the end of the sentence.

Table 4.8: The maximum ARI score achieved during hyperparameter optimization for the different models as described by experiment for $k = 20$ and $n = 1000$.

Clustering Model	**ARI Score**
Affinity Propagation	0.316
Chinese Whispers	**0.457**
DBScan	0.170
HDBScan	0.298
MeanShift	0.251
Optics	0.167

It is interesting to see that adding the `[SEP]` significantly increases the performance of the clustering algorithms. However, this is also expected, as BERT is trained using the `[SEP]` token. Adding 1 or more `[PAD]` at the end of the sentence does not improve the score any further.

Qualitative evaluation

So far, we have looked at the Adjusted Random Index Score as a way to measure how well the datasamples were clustered by meaning. However, because this is an arbitrary score, it is also important to do a qualitative

analysis, and go over examples to see what this clustering scheme results in. The below clusterings form partitions of the sentences to be clustered. Figures (4.8) - (**??**) show a a partition that is created when we run the above received best clustering algorithm on the word `arms`, which is polysemous. WordNet records 10 different semantic classes for this word. To keep this work concise, we will only show this as an explicit example. We refer to appendix (**??**) for some more examples. Because it is difficult to describe each partition through a concise and general statement, we show a few sampled words which represent the given semantic cluster.

```
                                                    - -
          Ms.  Gotbaum tried to slide her handcuffed arms from her back to her front
                                                    - -
          That magisterial use of the upper body and arms is her physical signature
entered the stage in pairs and forcefully stretched their arms and legs
                                ripped from patients' arms as they were carried away
                                                    - -
```

Figure 4.1: Partition 1 for the word `arms`. This cluster contains `arms` in the context of hancuffed arms. Notice that the sentiment is usually one of surprise, and rather negative.

```
                                - -
     She swooped him up into her arms and kissed him madly
                                - -
          I place my son in her arms and I pray that it somehow comforts her
perfect babies ...into the loving arms of middle class+ Americans
    which he falls back into her arms like a baby
     Sometimes I took it into my arms and felt its surprising heft
                                - -
```

Figure 4.2: Partition 2 for the word `arms`. This cluster contains `arms` in the context of arms where one person hugs or loves another. Notice that this usually includes a positive sentiment.

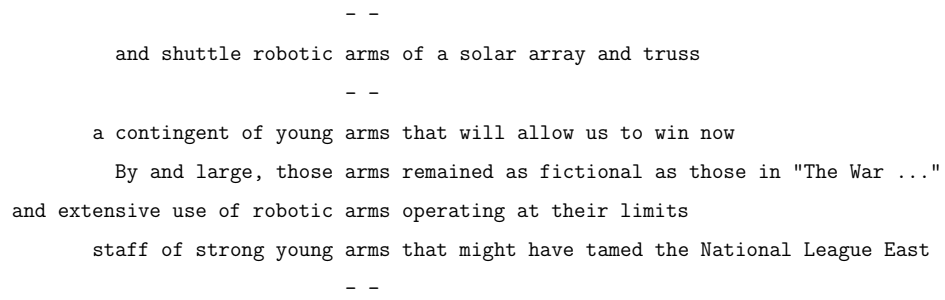


Figure 4.3: Partition 3 for the word `arms`. This cluster contains `arms` in the context of strong arms, usually in a competitive context.

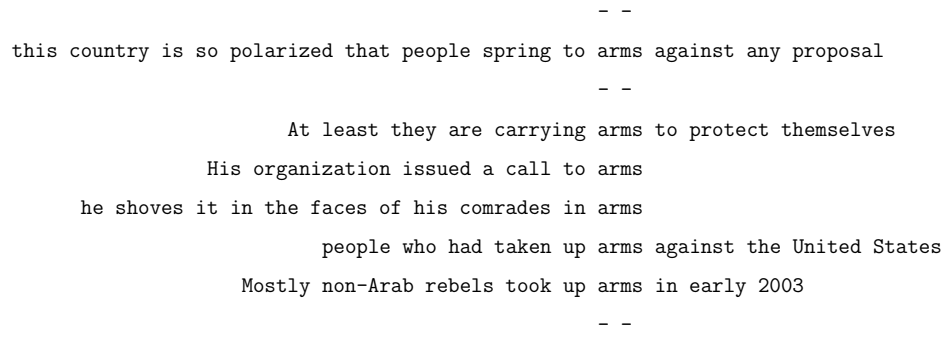
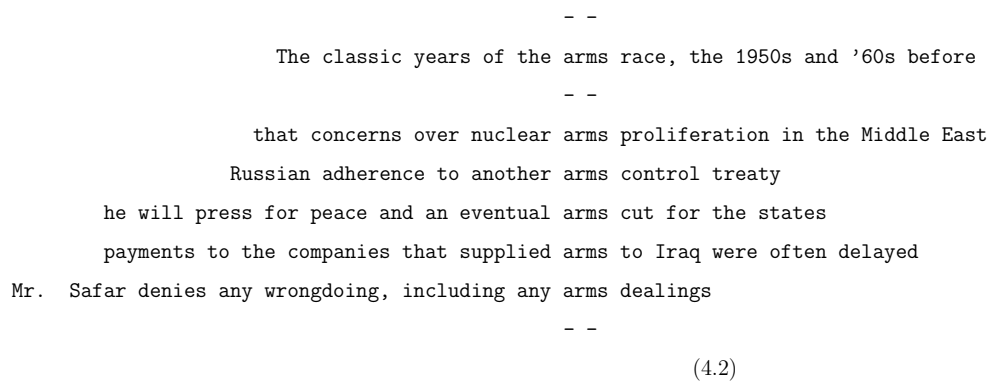


Figure 4.4: Partition 4 for the word `arms`. This cluster contains `arms` in the context of individuals and countries. This uses the semantic definition of arms which is analogous to `weaponry`.



(4.2)

Figure 4.5: Partition 5 for the word `arms`. This cluster contains `arms` in the context of countries (no individuals). This uses the semantic definition of arms which is analogous to `weaponry`. One can notice that these sentences usually refer to nuclear arms.

```
                                          - -
          leaned back in his chair and, with arms crossed,
                                          - -
God, fully in name, is at the bottom with his arms out wide.
    If you feel yourself falling, spread your arms
                                  Agamemnon, arms raised ...  barely contained violence
                    Mr.  James sat with his arms folded, his head lowered
      she felt tears in her eyes and held her arms out in simple joy
                                          - -
```

Figure 4.6: Partition 6 for the word `arms`. This cluster again refers to the arms of a person, however usually with a positive / optimistic sentiment.

As one can see, the different partitions do not quite capture semantics. Primarily, the different partitions capture context - which should be no surprise when we are sampling context embeddings. On a second look, however, the second strongest linguistic feature seems to be sentiment. This is also confirmed by other samples, which more often than not include strong notions of sentiment features. Although we cannot manually capture all possible words, as there is no formal definition for linguistic features within BERT vectors to our knowledge, where these concepts are purely man-made, we can thus conclude that BERT puts more weight on sentiment than on semantics, and that the semantics are a by-product of the context. This shows that more fundamental work needs to be done before we can formalize semantics.

4.3 Correlation between Part of Speech and Context within BERT

In the previous section, we have when we partition the sampled context vectors into clusters, sentiment becomes a prevalent factors next to semantics. Although the reason for this cannot be trivially concluded, this could also be a consequence of the nature of language. In this section, we are interested to what extent part-of-speech information entails semantic information. We pick part-of-speech because there are robust classifiers built on simple models with high accuracy rates (92%). Our hypothesis is that part-of-speech has

a strong entailment towards semantics. One example of this manifestation is by *frozen verbs*. A frozen verb is a noun which achieved it meaning through its verb form. The word `the run`, for example became a noun through its verb form `to run`, and thus is considered a frozen verb. Although this section is more a qualitative analysis of whether this hypothesis is true, due to the selective nature of samples that we can look at qualitatively. We finish this experiment by conducting a more quantitative approach.

4.3.1 Experiment setup

We test the hypothesis "semantics implies PoS" by conducting the following experiment. We define a set of words, including frozen verbs, which can appear as at least two different part-of-speech tags. These words are listed in the Appendix under section (C.2). For this experiment, we choose a word w_t from the above list and iterate over all WordNet meaning classes that this word presents. For each of the WordNet classes, we then measure the percentage of the most dominant part-of-speech tag. We refer to this percentage measure as the *dominance* of the majority class. If this percentage is close to 100%, (and generally above 50%) we can confidently say that meaning strongly implies part-of-speech, at least as measured in the SemCor dataset. More specifically, the algorithm looks as follows:

Algorithm 5: Analyzing dominance of part-of-speech within WordNet meaning clusters.

Input: $\mathbf{w}_{\text{analyse}} = [w_1, \ldots, w_N]$: The list of target words for which we want to calculate the dominance percentages for;

Result: A list of dominance percentages. Specifically, for each WordNet class for all the words, the distribution of fractions of how often the prevalent part-of-speech was occurrent.

```
out = []
for w in w_analyse:
    for wordNetClass in w.wordNetClasses() :
      for posLabels
      in wordNetClass:
        if length(posLabels) <= 1:
            continue
        dominantLabel = Counter(posLabels).mostCommon(1)
        domFraction = dominantLabel.count() / length(posLabels)
        out.append(domFraction)
return out
```

If all of the sampled target words w_t for all the sentences have the same assigned PoS tag, then the score results in a value of 1.0. If the dominant PoS tag occurs only half the time, this number decreases to 0.5. If there are three PoS tags, and all occur equally often, this percentage would drop down to 0.33 Please notice that in this experiment again, we only view simple PoS tags (i.e. "noun", "verb", "adjective", "pronoun"), and not the more complex ones listed above.

For a qualitative investigation, we visualize some of these words using the tensorboard projector, using both UMAP and PCA as the dimensionality reduction method. We color the samples by their respective part of speech tags.

4.3.2 Results

Quantitative Evaluation

The results for this experiment reveal that there is a strong correlation between part of speech and semantics. The following graph shows the cumulative plot of dominance amongst all sampled words that have been investigated.

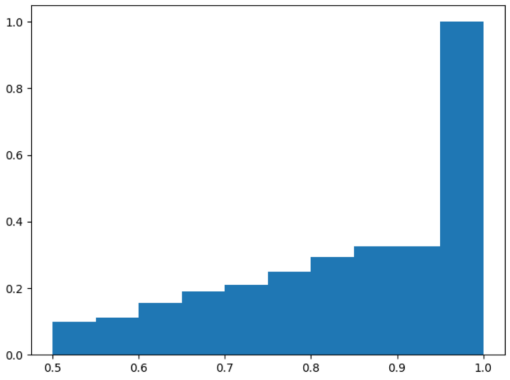


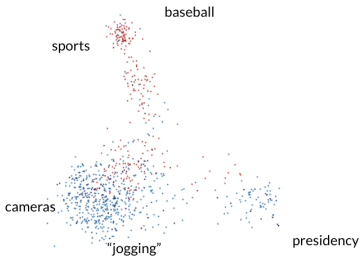
Figure 4.7: Cumulative dominance of the most occurring cluster. Dominance of a part of speech tag is measured by the percentage cover that the majority class intakes.

It is important to notice that some words did not have enough samples in the SemCor dataset for a significant valuation. We intentionally left out classes which had less than 5 samples. Also, SemCor is biased, as was previously stated. However, the results show that quite a majority of the sampled word-classes only contain a single part-of-speech tag, implying that part-of-speech is strongly correlated to meaning.
Because a numerical value is not intuitively interpretable, we continue with a qualitative evaluation of this phenomenon.

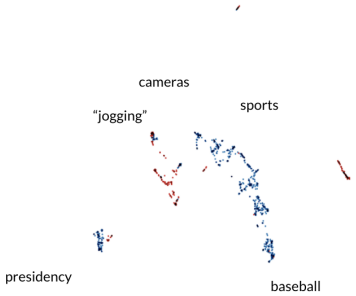
Qualitative Evaluation

To do a qualitative evaluation of the this hypothesis we visualize the BERT vectors using dimensionality reduction techniques. Due to the disadvantages that each dimensionality reduction techniques carry, we choose to look at X

using both PCA and UMAP. We only look at the polysemous words `block`, `run`, `cold` limited number of figures to keep the analysis concise
Figures (??) - (??) show the tensorboard visualizations. Please keep in mind that different colors imply different part of speech tags. Because the specific value does not play an inportant role, we leave out this legend.

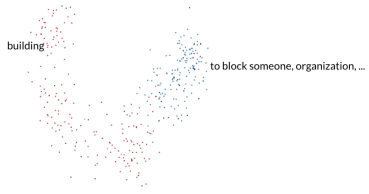


(a) Dimensionality reduction using PCA

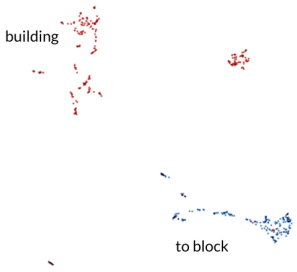


(b) Dimensionality reduction using UMAP

Figure 4.8: PCA and UMAP visualizations for context embeddings X sampled for the word $w =$`run`.

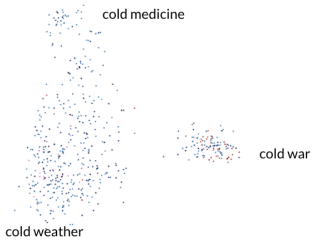


(a) Dimensionality reduction using PCA

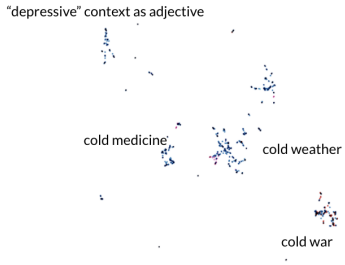


(b) Dimensionality reduction using UMAP

Figure 4.9: PCA and UMAP visualizations for context embeddings X sampled for the word $w =$`block`.



(a) Dimensionality reduction using PCA



(b) Dimensionality reduction using UMAP

Figure 4.10: PCA and UMAP visualizations for context embeddings X sampled for the word $w =$`cold`.

Following trends are apparent: PCA shows a very smooth transition from different semantic classes, whereash UMAP stronlgy shows that clusters are forming between different semantic classes. This is also true for part-of-speech tags, however, we cannot generalize this as well, because part-of-speech tags strongly correlate with semantics.
Furthermore, one can see that globally, BERT does seem to organize the context embeddings by semantics. This is apparent in figures (**??**) and (**??**), where we can see that different blobs (UMAP) and extremes (PCA) in the figures respond to different semantic classes. Especially in Figure (**??**) (a), we can see that in the case of UMAP, the central clsuter strongly mixes nouns with adjectives (red and blue cluster), and that in the case of PCA, the context embeddings are constrained in such a way that embeddings which mix different semantic classes meet in the middle. Looking at the different blobs in UMAP we can again confirm the hypothesis that semantics strongly correlates to part-of-speech, as most clusters have a strong predominant color.

Chapter 5

Exploiting subspace organization of semantics of BERT embeddings

So far we have tried to understand how language features are better represented within the BERT language model. Now we want to understand how different modifications to the BERT model affect the performance on language modelling task. Because GLUE is a very general language modelling benchmark, we proceed with GLUE. Our analysis will be limited to BERT only, although similar changed can also be made to modifications of BERT including ALBERT [66], DistillBERT [107]. We will keep the contents of this analysis focused on BERT only, as all transformer models share a similar architecture and as we expect the changes to similarly affect the different models. As such, the focus lies on the ablation study with the BERT model forming our baseline. To make the discussion easier, we will refer to any of the models modified by our experiments **BERnie**. Please refer to (**??**) for a refresher on how sentences are tokenized inside BERT, and how tokens are converted to the latent representation and finally into context embeddings through the BERT model.

Before we continue, we want to introduce the concept of a *split-word*. A

split-word is a word for which we will introduce additional embedding vectors in the BERT model, by some criterion that depends on the experiment setting. A split-word will in most cases have high-variance amongst embedding-vectors sampled through BERT. The motivation behind this is that introducing more specialized embeddings for tokens which result in context embeddings that have high variance will allow to model more complex distributions, which could approximate the underlying true distribution better. This would result in multiple simple, well-parametrized probability distributions, rather than a single under-parametrized distribution which has high variance due to the low expressiveness.

Table 5.1: For each word in the SemCor dataset, the number of WordNet senses, and the mean variance of $n = 500$ sampled embedding vectors, across all embedding dimensions. The table shows a subset of the words with highest and lowest variance. There seems to be a strong correlation between number of WordNet senses and the variance amongst sampled BERT vectors.

Word	**Number of Senses recorded in WordNet**	**Mean Variance**
field	21	0.430
right	36	0.425
forward	12	0.424
general	10	0.422
stands	24	0.393
exchange	17	0.393
⋮	⋮	⋮
loans	3	0.238
rocks	9	0.241
ignore	5	0.275
gasoline	1	0.275
humans	2	0.275
announcement	2	0.275

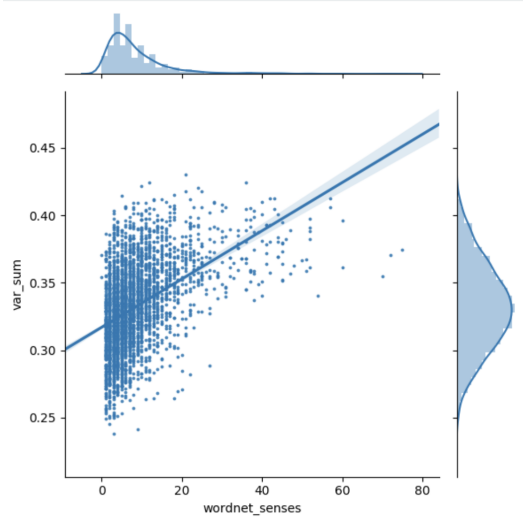


Figure 5.1: For each word w, we sample both the number of WordNet semantic classes that are recorded in the WordNet dataset. We also calculate the dimension-wise mean variance between $n = 500$ sampled vectors for the word w. This constitutes a point on this plot. We repeat this procedure for the most 20'000 most frequent words which consist of a single token (i.e. are not split up into further subtokens when the tokenizer is applied). We show that although the variance is relatively high, especially if only few WordNet classes are present, there is a correlation between the number of WordNet classes and the variance of sampled BERT context embeddings. The right and top distributions show histograms of how occurrent the variance and WordNet classes are respectively. Our assumption is that if we introduce additional embedding-vectors inside BERT for certain words, that more complex distributions can be captures for words that have a higher number of WordNet classes.

5.1 BERnie: Making the variances across BERT embeddings more equal

We will introduce two adaptions to the BERT model, BERnie PoS and BERnie semantics. We will present the results after having introduced both adaptations.

5.1.1 BERnie PoS

We want to start with a simple model first. Because we have seen in the above section, that there is a strong correlation between semantics and part-of-speech, the initial idea is to introduce additional embedding vectors which are able to capture semantics (rather than just purely word tokens) better. Because of the strong correlation, part-of-speech is used as a proxy for semantics. One of the advantages that this carries is that we can use the spaCy PoS tagger [3] to classify the PoS tags with high accuracy.
Our first model adaption is called **BERnie PoS**. In this case, we introduce new tokens, and new embedding vectors into the embedding layers of the transformers inside of BERT. These newly introduced embedding vectors are supposed to be able to capture more complex target distributions, as we introduce more weights for regions of semantic ambiguity, which have higher variance as can be seen in Figure (5.1).

Experiment setup

BERnie PoS introduces one new embedding vector for each possible PoS configuration of the split-words.
As an example, instead of having a single embedding vector for the word `run`, we introduce two new embeddings `run_ VERB` and `run\_NOUN`, which both replace the intial *run*-embedding. We will refer to `run\_VERB` and `run\_NOUN` as so called *sub-embeddings*, as these exploit the subspace structure of the context embeddings sampled for the word `run`. As for the tokenizer, we also introduce a mechanism which turns any occurrence of `run` into one of the sub-embeddings.
Specifically, our PoS modified pipeline would now look as follows.

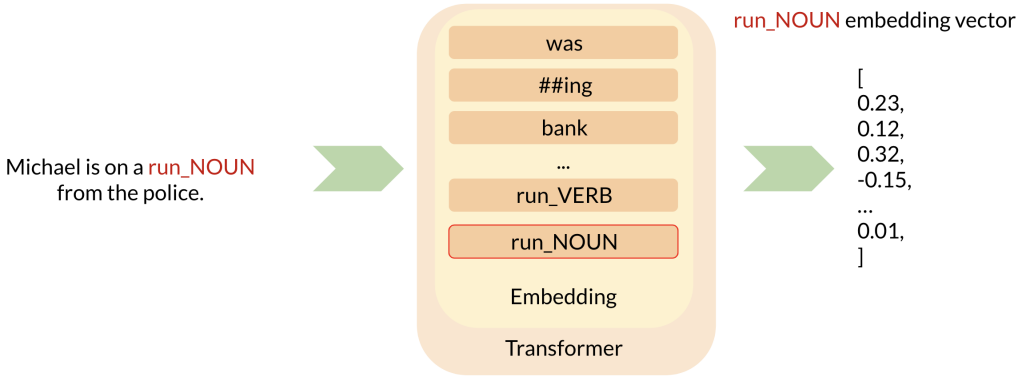


Figure 5.2: The PoS modified pipeline. The BERnie PoS model takes as input a sentence s. The sentence s is converted to a sequence of BERT tokens $[t_1, \ldots, t_m]$ as defined in a given vocabulary V. This vocabulary V is extended with the additionally introduced tokens for each of the split-words. For each target token t_{target}, we make the token more specific by converting the token to a more specialized token-representation, which specifies the part-of-speech information as part of the token. In this case, *run* becomes the more specialized run_VERB. Again, each item in the vocabulary V has a corresponding embedding vector inside the embedding layer of the transformer. This embedding vector is used by the intermediate layers of the transformer, and thus affects the downstream pipeline of the language model for any subsequent layers of the transformer.

To identify whether the occurring `run` in an example sentence is a verb or a noun, we use the spaCy part-of-speech tagger [3], which claims 92.6 % accuracy for this task.

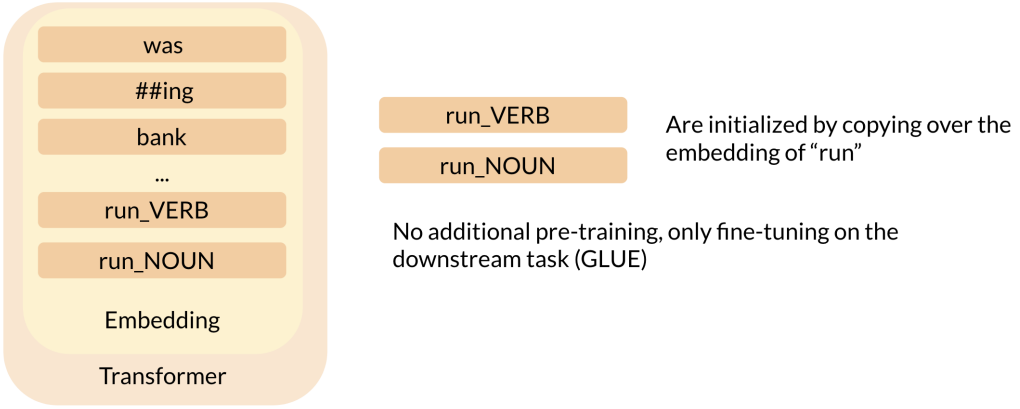
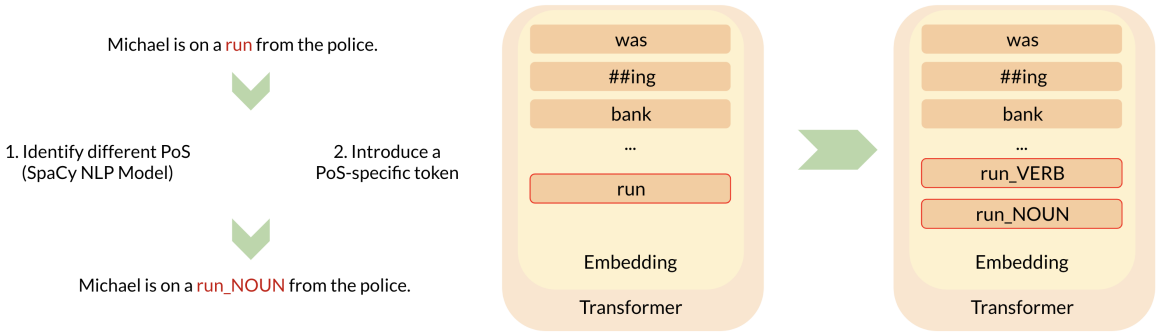


Figure 5.3: Inside the embedding layer of the transformer which occurs at each layer of BERT, we introduce more specific embeddings `run_VERB` and `run_NOUN`. The BERT model should now capture more expressiveness, as more weight got introduced for a part of the model which results in a probability distribution with high variance. The original `run` embedding is removed.

.



(a) The part-of-speech modified tokenizer pipeline.

(b) The part-of-speech modified BERT embedding inside the transformer that BERT uses.

Figure 5.4: The resulting, fully modified BERnie PoS pipeline.

During training, all weights of BERT are fine-tuned to the downstream GLUE task as described in section

5.1.2 BERnie Meaning

Similar to the above model, we are introducing new BERT tokens and their respective embeddings in the transformer embedding layer, similar to the above section 5.1.1. This time however, we do not introduce new tokens by part-of-speech, as we did in the previous section, but rather by the clustering algorithms above, which were supposed to cluster by WordNet classes 4.2. Thus, implicitly, we are hoping to cluster by semantics. The idea here is to adhere more strongly to the subspace structure as seen in section 4.3, following the investigation from section 4. Please notice that although we did optimized the development set to mimick the clustering as is the case analogously to the WordNet classes, that the resulting clusters are implicit latent variables that are not hard-coded, and thus cannot be intuitively understood without further analysis; and also because in most cases we cannot formally describe the concept of semantics and other linguistic properties inside of BERT.

Experiment setup

Again, we start with the standard BERT model, whose pipeline looks as follows.

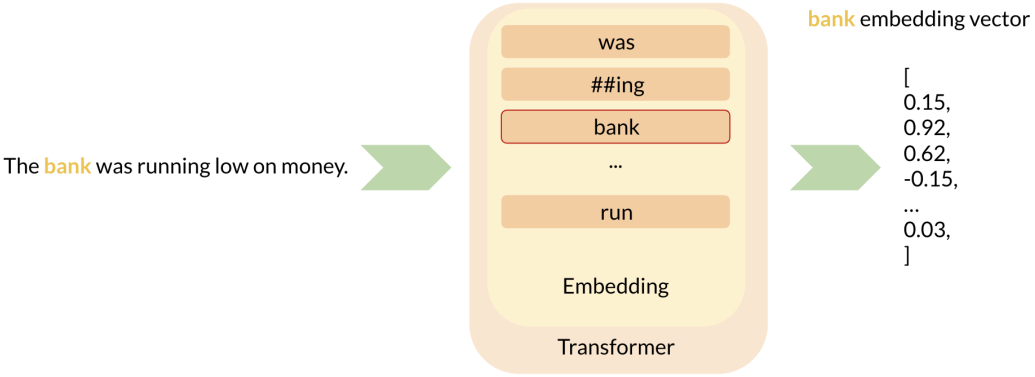
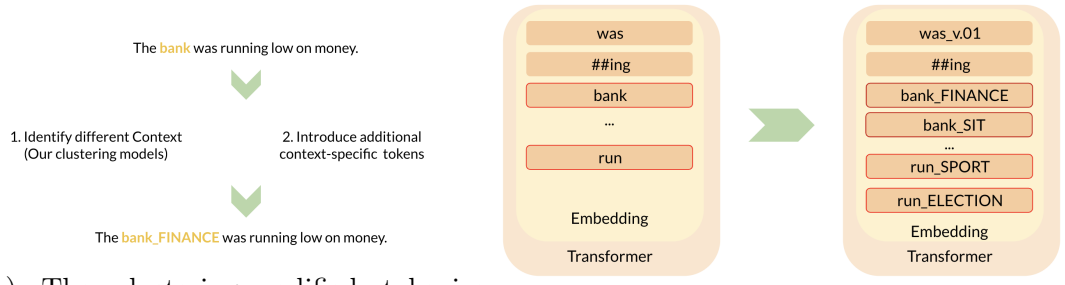


Figure 5.5: The semantic modified pipeline. The BERnie Cluster model takes as input a sentence s. The sentence s is converted to a sequence of BERT tokens $[t_1, \ldots, t_m]$ as defined in a given vocabulary V. This vocabulary V is extended with the additionally introduced tokens for each of the split-words. For each target token t_{target}, we make the token more specific by converting the token to a more specialized token-representation, which specifies the clustereing information as part of the token. In this case, *bank* becomes the more specialized $bank\_FINANCE$. Again, each item in the vocabulary V has a corresponding embedding vector inside the embedding layer of the transformer. This embedding vector is used by the intermediate layers of the transformer, and thus affects the downstream pipeline of the language model for any subsequent layers of the transformer.

We now also modify the tokenizer in such a way, that ambigious and polysemous words are replaced with a more specific token. As an example, we want to replace the word *bank* with a token, which captures whether or not the *bank* that we refer to implies a 1) financial institution, or 2) a river bank. Of course, the individual clusters contain context information, and not purely semantic information. For easier explainability, we assume that the different clusters referr to semantic clusters, but this is not always the case as we have seen before.

We use the our clustering approach from 4.2 as the intermediate model to distinguish on which bank the sentence refers to. We also introduce new embedding vectors that the new tokens correspond to.



(a) The clustering-modified tokenizer pipeline

(b) The clustering-modified BERT embedding

Figure 5.6: plots of....

After we have concluded these changed, we arrive at the following model, which has a modified tokenizer, and a modified BERT model. We call the corresponding model **BERnie Cluster**.

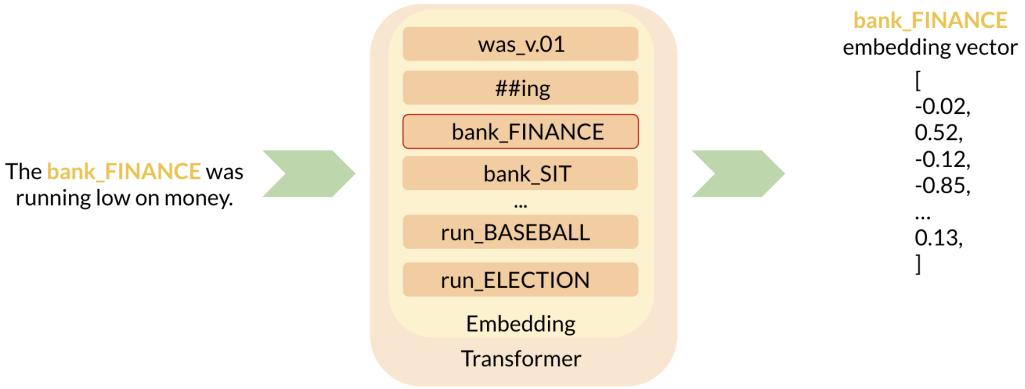


Figure 5.7: The BERT model takes as input a sentence s. The The resulting, fully modifid BERnie Cluster pipeline.

Results

We now do an evaluation of the introduced BERnie PoS and BERnie Cluster model. To understand the resulting performance as a language model, we refer to the GLUE benchmark dataset. Because we want to understand what these modifications have as an affect, we focus on an ablation study rather than absolute performance improvement to state-of-the-art methods like ALBERT. Weights which are instantiated specifically for the GLUE tasks

Task	Score Measure	BERT	BERnie PoS	BERnie Cluster
CoLA	Accuracy	**0.5739**	0.5263	0.5457
MRPC	Accuracy	**0.8223**	0.8199	0.8064
	F1	**0.8778**	0.8614	0.8684
	Mixed	0.8501	**0.8579**	0.8374
SST-2	Accuracy	0.9214	0.9203	**0.9266**
STS-B	Correlation	**0.8841**	0.8615	0.8574
	Pearson	**0.8860**	0.8621	0.8587
	Spearman	**0.8822**	0.8601	0.8567
QNLI	Accuracy	**0.9126**	0.9090	0.9020
RTE	Accuracy	**0.6462**	0.6083	*0.5722*
WNLI	Accuracy	*0.35915*	0.3947	**0.4649**
MNLI	Accuracy	0.8453		0.8334
SNLI	Accuracy	**0.8461**		0.8367
QQP	Accuracy	**0.9113**		0.9042
	F1	**0.8809**		0.8732
	Mixed	**0.8961**		0.8887

Table 5.2: GLUE Benchmark performance measures for the BERT model, BERnie PoS and BERnie Cluster. Higher scores are better. The task-wide best-performers are marked in bold. All values are the average scores of two runs.

(final layer) are once instantiated with a random seed of 42, and once with a random seed of 101. The scores are the mean of the two runs.

One can see that for most tasks, the BERnie model modifications performs slightly worse than the standard BERT model by a couple percentage points. However, he WNLI and RTE experiments are considerably deviating in performance through the addition of the additional embedding vectors. For the BERnie Cluster model, WNLI performs **10.56%** better than standard BERT, and RTE performs **-7.40%** worse than the standard BERT implementation. Similar results are observable for the difference between the unmodified BERT and the BERnie PoS model with **3.52%** improvement for WLNI and **-3.79%** decline in performance for the RTE task.

Also, compared to BERnie PoS, the BERnie Cluster model amplifies the performance-difference to BERT in both the negative and positive direction.

5.1.3 BERnie Meaning with additional pre-training

Our first idea is that the overall negative trend in performance might be due to the newly initialized vectors not being calibrated enough.
We assume that the BERnie require additional pre-training, as we are adding more specific embedding vectors, without fine-tuning them. We assume that this is necessary, as this is how the weights of BERT are also trained in the original paper [36] through the masked language model approach.

5.1.4 Experiment setup

We want to test to what extent pre-training the newly initialized embeddings helps improve the performance on the embeddings. We apply 1 epoch of pre-training on the news corpus described in Section 2.
We evaluate the models of the same GLUE benchmark tasks as in the previous section.
We instantiate the additional embeddings as described in the previous section. We then run one full epoch of training with the same training parameters as used for GLUE on the news.corpus.2007 dataset using a masked languag emodel methodology . We save this model and load it for the GLUE tasks. BERnie full Pre-training trains the entire embeddings, whereas BERnie partial Pre-Training fixates all embedding vectors except the ones for the ones that were newly added.
We refer to *full pre-training*, resulting in the **BERnie full Pre** model, when we allow the gradient to be update on all the embedding weights during pre-training. We refer to *partial pre-training*, resulting in the **BERnie partial Pre** model, when we only allow gradient updates for the (sub-)embeddings of the split-words. In this case, only the newly added embeddings vectors are calibrated, while all other embedding vectors are kept constant.

5.1.5 Results

Again, we ran the experiments and show the results as the average of two runs. Due to time-constraints and certainty of bugs for some experiments,

we only show the results for a subset of the GLUE tasks.

Task	Score Measure	BERnie	BERnie full Pre	BERnie partial Pre
CoLA	Accuracy	0.5457	0.5418	**0.5731**
SST-2	Accuracy	**0.9266**	0.9169	**0.9266**
QNLI	Accuracy	**0.9020**	0.5374	0.6700
RTE	Accuracy	**0.5722**	0.4784	0.4729
WNLI	Accuracy	**0.4649**	0.4225	0.4507

Table 5.3: GLUE Benchmark performance measures for the BERnie Cluster model without any additional pre-training, the BERnie Cluster with full pre-training and BERnie with partial pre-training. Higher scores are better. The task-wide best-performers are marked in bold. All values are the average scores of two runs.

As one can see, the additional pre-training does not improve the performance. However, we assume that this is due to the pre-training corpus not being similar to the pre-training corpus that was initially used for BERT (which is too big to be run for this test), or the hyperparamters not being optimal for this run. Finally, it could also be that the initialization for the newly addeed vectors was suboptimal, as we have seen strong deviations (of up to 5% in performance) over different random seeds for the GLUE tasks. Due to time-constraints, we decided to not move ahead with this experiment.

Chapter 6

Conclusion

Bibliography

[1] Computational intelligence lab: Lecture 5 embeddings. http://da.inf.ethz.ch/teaching/2019/CIL/lecture/CIL2019-05-Word-Embeddings.pdf. Accessed: 2020-05-11.

[2] facebook/ax: Adaptive experimentation platform. https://github.com/facebook/Ax. Accessed: 2020-05-04.

[3] spacy: Part-of-speech tagging. https://spacy.io/api/annotation#pos-tagging. Accessed: 2020-04-13.

[4] Translation task 2017 - acl 2017 - news corpus. http://www.statmt.org/wmt17/translation-task.html. Accessed: 2020-05-03.

[5] Understanding lstms. https://colah.github.io/posts/2015-08-Understanding-LSTMs/. Accessed: 2020-05-02.

[6] Margareta Ackerman and Shai Ben-David. Clusterability: A theoretical study. 2009.

[7] Alan Akbik, Tanja Bergmann, and Roland Vollgraf. Pooled contextualized embeddings for named entity recognition. *Proceedings of NAACL-HLT 2019*, pages 724–728, 2019.

[8] David Alvarez-Melis and Tommi S. Jaakkola. Gromov-wasserstein alignment of word embedding spaces. 2018.

[9] Nikolay Arefyev, Boris Sheludko, Adis Davletov, Dmitry Kharchev, Alex Nevidomsky, and Alexander Panchenko. Neural granny at semeval-2019 task 2: A combined approach for better modeling of semantic relationships in semantic frame induction. *Proceedings ofthe 13th International Workshop on Semantic Evaluation (SemEval-2019)*, page 31–38, 2019.

[10] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv*, 2016.

[11] Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. 2016.

[12] Collin F. Baker, Charles J. Fillmore, and John B. Lowe. The berkeley framenet project. *ACL/COLING, volume 1*, pages 86–90, 1998.

[13] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jau-

vin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

[14] Yoshua Bengio, Holger Schwenk, Jean-Sebastien Senecal, Morin Frederic, and Jean-Luc Gauvain. Neural probabilistic language models. *Innovations in Machine Learning*, pages 137–186, 2006.

[15] Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. The fifth pascal recognizing textual entailment challenge. In *TAC*, 2009.

[16] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research 13*, pages 281–305, 2012.

[17] Chris Biemann. Chinese whispers: An efficient graph clustering algorithm and its application to natural language processing problems. *In Workshop on Graph-based Methods for Natural Language Processing*, pages 73–80, 2006.

[18] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. 2017.

[19] Rishi Bommasani, Kelly Davis, and Cardie Claire. Bert wears gloves: Distilling static embeddings from pretrained contextual representations. 2019.

[20] Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*, 2015.

[21] Arthur Bražinskas, Serhii Havrylov, and Ivan Titov. Embedding words as distributions with a bayesian skip-gram model. 2018.

[22] Jane Bromley, Isabelle Guyon, and Yann LeCun. Signature verification using a "siamese" time delay neural network. 1994.

[23] Aylin Caliskan, Joanna J. Bryson, and Arvind Narayanan. Semantics derived automatically from language corpora contain human-like moral choices. *Science 356 (6334)*, pages 183–186–44, 2019.

[24] Jose Camacho-Collados and Mohammad Taher Pilehvar. A survey on vector representations of meaning from word to sense embeddings: A survey on vector representations of meaning. 2018.

[25] Ricardo J. G. B. Campello, Davoud Moulavi, and Joerg Sander. Density-based clusteering based on hierarchical density estimates. *PAKDD 2013: Advances in Knowledge Discovery and Data Mining*, pages 160–172, 2013.

[26] Xavier Carreras and Lluis Marquez. Introduction to the conll-2004 shared task: Semantic role labeling. *In Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004)*, 2004.

[27] Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia

Specia. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*, 2017.

[28] Gal Chechik, Varun Sharma, Uri Shalit, and Samy Bengio. Large scale online learning of image similarity through ranking. *JMLR*, 2010.

[29] Xinlei Chen, Alan Ritter, Abhinav Gupta, and Tom Mitchell. Sense discovery via co-clustering on images and text. 2019.

[30] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. 2005.

[31] Andy Coenen, Emily Reif, Ann Yuan, Been Kim, Adam Pearce, Fernanda Viégas, and Martin Wattenberg. Visualizing and measuring the geometry of bert. 2019.

[32] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 603–619, 2002.

[33] Alexis Conneau, Guillaume Lample, Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. Word translation without parallel data. 2017.

[34] Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190. Springer, 2005.

[35] Michael Denkowski. A survey of techniques for unsupervised word sense induction. 2009.

[36] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *Arxiv*, 2018.

[37] William B Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005.

[38] Matrin Ester, Hans-Peter Kriegel, Joerg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters. *KDD-96 Proceedings*, 1996.

[39] Kawin Ethayarajh. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. 2019.

[40] W. N. Francis and H. Kucera. A standard corpus of present-day edited american english, for use with digital computers. 1964.

[41] Brendan J. Frey and Delbert Dueck. Clustering by passing messages between data points. 2007.

[42] Philip Gage. A new algorithm for data compression. *The C Users Journal*, 1994.

[43] Octavian-Eugen Ganea, Gary Becigneul, and Thomas Hofmann. Hy-

perbolic entailment cones for learning hierarchical embeddings. 2018.
[44] Weifeng Ge, Weilin Huang, Dengke Dong, and Matthew R. Scott. Deep metric learning with hierarchical triplet loss. 2018.
[45] Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pages 1–9. Association for Computational Linguistics, 2007.
[46] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. 2006.
[47] Aric Hagberg, Dan Schult, and Pieter Swart. Networkx. `https://github.com/networkx/networkx`, 2006. Accessed: 2020-05-07.
[48] Bar-Roy Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. The second pascal recognising textual entailment challenge. In *Proceedings of the second PASCAL challenges workshop on recognising textual entailment*, volume 6, pages 6–4. Venice, 2006.
[49] Zellig Harris. Distributional structure. *Word*, 10, 1954.
[50] W. F. Hegel. Encyclopedia of the philosophical science: Science of logic. 1817.
[51] John Hewitt and Christopher D Manning. A structural probe for finding syntax in word representations. *Association for Computational Linguistics*, 2019.
[52] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Arxiv*, 1997.
[53] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. 2014.
[54] Matthew Honnibal and Ines Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear, 2017.
[55] Renfen Hu, Shen Li, and Shichen Liang. Diachronic sense modeling with deep contextualized word embeddings: An ecological view. *Proceedings ofthe 57th Annual Meeting ofthe Association for Computational Linguistics*, pages 3899–3908, 2019.
[56] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, pages 193–218, 1985.
[57] Shankar Iyer, Nikhil Dandekar, and Kornel Csernai. First quora dataset release: Question pairs, 2017.
[58] Ganesh Jawahar, Benoit Sagot, and Djame Seddah. What does bert learn about the structure of language? *Proceedings ofthe 57th Annual Meeting ofthe Association for Computational Linguistics*, pages 3651–3657, 2019.

[59] Sophie Jentzsch, Constantin Rothkopf, and Patrick Schramowski Kristian Kersting. On measuring social biases in sentence encoders. *AIES 2019 - Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pages 37–44, 2019.

[60] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. Spanbert: Improving pre-training by representing and predicting spans. 2019.

[61] Vidur Joshi, Matthew Peters, and Mark Hopkins. Extending a parser to distant domains using a few dozen partially annotated examples. 2018.

[62] Rafal Jozefowicz, Orriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. 2016.

[63] Mahmut Kaya and Hasan Sakir Bilge. Deep metric learning : A survey. *MDPI Symmetry*, 2019.

[64] Sneha Kudugunta, Ankur Bapna, Isaac Caswell, Naveen Arivazhagan, and Orhan Firat. Investigating multilingual nmt representations at scale. 2018.

[65] Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. Unsupervised machine translation using monolingual corpora only. *ICLR*, 2018.

[66] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *ICLR*, 2020.

[67] Jason Lee, Kyunghyun Cho, and Thomas Hofmann. Fully character-level neural machine translation without explicit segmentation. 2017.

[68] Hector Levesque, Ernest Davis, and Leora Morgenstern. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*, 2012.

[69] Yoav Levine, Barak Lenz, Or Dagan, Dan Padnos, Or Sharir, Shai Shalev-Shwartz, Amnon Shashua, and Yoav Shoham. Sensebert: Driving some sense into bert. 2019.

[70] Chaya Liebeskind, Ido Dagan, and Jonathan Schler. An algorithmic scheme for statistical thesaurus construction in a morphologically rich language. *Applied Artificial Intelligence Vol. 33*, pages 483–496, 2019.

[71] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, Veselin Stoyanov, and Paul G. Allen. Roberta: A robustly optimized bert pretraining approach. 2019.

[72] S. P. Lloyd. Least squares quantization in pcm. *Technical Report RR-5497 Bell Lab*, 1957.

[73] Shuang Ma, Daniel Mcduff, and Yale Song. M3 d-gan: Multi-modal

multi-domain translation with universal attention. 2019.

[74] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. *L. M. Le Cam and J. Neyman (Eds.), Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, pages Vol. 1, pp. 281–297, 1967.

[75] P. C. Mahalanobis. On the generalized distance in statistics. *Proc. Nat. Inst. Sci.*, 1936.

[76] Matej Martinc, Jozef Stefan, Institute Slovenia, Syrielle Montariol, Elaine Zosa, and Lidia Pivovarova. Capturing evolution in word usage: Just add more clusters? *In Proceedings ofACM Conference (Conference'17)*, 2020.

[77] Chandler May, Alex Wang, Shikha Bordia, Samuel R. Bowman, and Rachel Rudinger. On measuring social biases in sentence encoders. 2019.

[78] Diana Mccarthy, Marianna Apidianaki, and Katrin Erk. Word sense clustering and clusterability. 2016.

[79] Timothee Mickus, Denis Paperno, and Kees Van Deemter. What do you mean, bert? assessing bert as a distributional semantics model. 2019.

[80] Ankerst Mihael, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: ordering points to identify the clustering structure. *ACM SIGMOD Record 28, no. 2*, pages 49–60, 1999.

[81] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, 2013.

[82] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546, 2013.

[83] George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. Introduction to wordnet: An on-line lexical database. *International Journal of Lexicography*, pages 235–244, 1990.

[84] George A. Miller and Walter G. Charles. Contextual correlates of semantic similarity. *Language and cognitive processes 6*, pages 1–28, 1991.

[85] George A. Miller, Martin Chodorow, Shari Landes, Claudia Leacock, and Robert G. Thomas. Using a semantic concordance for sense identification. pages 240–243, 1994.

[86] Mehrad Moradshahi, Hamid Palangi, Monica S Lam, Paul Smolensky, and Jianfeng Gao. Hubert untangles bert to improve transfer across nlp tasks. 2019.

[87] Panagiotis Moutafis, Mengjun Leng, and Ioannis A. Kakadiaris. An overview and empirical comparison of distance metric learning meth-

ods. *IEEE Transactions on Cybernetics*, 2017.

[88] Roberto Navigli and Federico Martelli. An overview of word and sense similarity. *Natural Language Engineering*, pages 693–714, 2019.

[89] Jiazhi Ni, Jie Liu, Chenxin Zhang, Dan Ye, and Zhirou Ma. Fine-grained patient similarity measuring using deep metric learning. 2017.

[90] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[91] Maria Pelevina, Nikolay Arefyev, Chris Biemann, and Alexander Panchenko. Making sense of word embeddings. *Proceedings ofthe 1st Workshop on Representation Learning for NLP*, pages 174–183, 2016.

[92] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. *EMNLP*, page 1532–1543, 2014.

[93] Matthew E Peters, Mark Neumann, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *Arxiv*, 2018.

[94] Matthew E Peters, Mark Neumann, Luke Zettlemoyer, and Wen-Tau Yih. Dissecting contextual word embeddings: Architecture and representation. 2018.

[95] Mohammad Taher Pilehvar and Jose Camacho-Collados. Wic: the word-in-context dataset for evaluating context-sensitive meaning representations. 2019.

[96] Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Bjorkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. Towards robust linguistic analysis using ontonotes. *In Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152, 2013.

[97] Behrang Qasemizadeh, Miriam R L Petruck, Laura Kallmeyer, and Marie Candito. Semeval-2019 task 2: Unsupervised lexical frame induction. *Proceedings ofthe 13th International Workshop on Semantic Evaluation (SemEval-2019)*, page 16–30, 2019.

[98] Alec Radford, Karthk Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *Arxiv*, 2018.

[99] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *Arxiv*, 2019.

[100] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv*

preprint arXiv:1606.05250, 2016.

[101] William M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, pages 846–850, 1971.

[102] Nils Reimers, Benjamin Schiller, Tilman Beck, Johannes Daxenberger, Christian Stab, and Iryna Gurevych. Classification and clustering of arguments with contextualized word embeddings. 2019.

[103] Steffen Remus and Chris Biemann. Retrofitting word representations for unsupervised sense aware word similarities. *In Pro- ceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152, 2013.

[104] Eugénio Ribeiro, Vânia Mendonça, Ricardo Ribeiro, David Martins e Matos, Alberto Sardinha, Ana Lucia Santos, and Luísa Coheur. L 2 f/inesc-id at semeval-2019 task 2: Unsupervised lexical semantic frame induction using contextualized word representations. *Proceedings ofthe 13th International Workshop on Semantic Evaluation (SemEval-2019)*, pages 130–136, 2019.

[105] Oren Rippel, Manohar Paluri, Piotr Dollar, and Lubomir Bourdev. Metric learning with adaptive density discrimination. 2016.

[106] David Rumelhart, Geoffrey Hinton, and Ronald Williams. Learning internal representations by error propagation. 1985.

[107] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, 2019.

[108] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. 2016.

[109] Dinghan Shen, Pengyu Cheng, Dhanasekar Sundararaman, Xinyuan Zhang, Qian Yang, Meng Tang, Asli Celikyilmaz, and Lawrence Carin. Learning compressed sentence representations for on-device text processing. 2019.

[110] Peng Shi, Jimmy Lin, and David R Cheriton. Simple bert models for relation extraction and semantic role labeling. 2019.

[111] Jamin Shin, Andrea Madotto, and Pascale Fung. Interpreting word embeddings with eigenvector analysis. 2018.

[112] Yuqi Si, Jingqi Wang, Hua Xu, and Kirk Roberts. Enhancing clinical concept extraction with contextual embeddings. 2019.

[113] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural*

language processing, pages 1631–1642, 2013.

[114] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. 2016.

[115] Hyun Oh Song, Stefanie Jegelka, Vivek Rathod, and Kevin Murphy. Deep metric learning via facility location. 2017.

[116] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. *CVPR*, 2016.

[117] Juan Luis Suarez, Salvador Garcia, and Francisco Herrera. A tutorial on distance metric learning: Mathematical foundations, algorithms and experiments. 2019.

[118] Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. Ernie 2.0: A continual pre-training framework for language understanding. 2019.

[119] Ilya Sutskever, James Martens, and Geoffrey Hinton. Generating text with recurrent neural networks. 2011.

[120] Sang-Sang Tan and Jin-Cheon Na. Position atteention-based frame identification with bert: A deep learning approach to target disambiguation and semantic frame selection. 2019.

[121] Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy Lin. Distilling task-specific knowledge from bert into simple neural networks. 2019.

[122] Henry Tsai, Jason Riesa, Melvin Johnson, Naveen Arivazhagan, Xin Li, and Amelia Archer. Small and practical bert models for sequence labeling. 2019.

[123] Vahe Tshitoyan, John Dagdelen, Leigh Weston, Alexander Dunn, Ziqin Rong, Olga Kononova, Kristin A. Persson, Gerbrand Ceder, and Anubhav Jain. Unsupervised word embeddings capture latent knowledge from materials science literature. *Nature Vol 571 Issue 7763*, pages 95–98, 2019.

[124] Evgeniya Ustinova and Victor Lempitsky. Learning deep embeddings with histogram loss. *NIPS*, 2016.

[125] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Neural Information Processing Systems*, 2017.

[126] Luke Vilnis and Andrew McCallum. Word representations via gaussian embedding. *CoRR*, abs/1412.6623, 2014.

[127] Alex Wang and Kyunghyun Cho. Bert has a mouth, and it must speak: Bert as a markov random field language model. 2019.

[128] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Superglue: A stickier benchmark for general-purpose language understanding

systems. *NeurIPS*, 2019.

[129] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *ICLR*, 2019.

[130] Jian Wang, Feng Zhou, Shilei Wen, Xiao Liu, and Yuanqing Lin. Deep metric learning with angular loss. 2017.

[131] Wei Wang, Bin Bi, Ming Yan, Chen Wu, Zuyi Bao, Jiangnan Xia, Liwei Peng, and Luo Si. Structbert: Incorporating language structures into pre-training for deep language understanding. 2019.

[132] Xun Wang, Xintong Han, Weilin Huang, Dengke Dong, and Matthew R. Scott. Multi-similarity loss with general pair weighting for deep metric learning. 2019.

[133] Ziyu Wang, Masrour Zoghi†, Frank Hutter, David Matheson, and Nando de Freitas. Bayesian optimization in high dimensions via random embeddings. *AAAI Publications, Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

[134] Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*, 2018.

[135] Brendan Whitaker, Denis Newman-Griffis, Aparajita Haldar, Hakan Ferhatosmanoglu, and Eric Fosler-Lussier. Characterizing the impact of geometric properties of word embeddings on task performance. 2019.

[136] Gregor Wiedemann, Steffen Remus, Avi Chawla, and Chris Biemann. Does bert make any sense? interpretable word sense disambiguation with contextualized embeddings. 2019.

[137] Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics, 2018.

[138] Ludwig Wittgenstein. Philosophical investigations. 1953.

[139] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771, 2019.

[140] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, and Mohammad Norouzi et. al. Google's neural machine translation system: Bridging the gap between human and machine translation. 2016.

[141] An Yan, Fanbo Xiang, and Yiming Zhang. Embedding learning by optimal transport. 2019.

[142] Henghui Zhu, Ioannis Ch Paschalidis, and Amir Tahmasebi. Clinical concept extraction with contextual word embedding. 2018.

Appendix A

Background

A.1 Linguistic features

Other linguistic features are recorded in [129] and include

Coarse-Grained Categories	Fine-Grained Categories
Lexical Semantics	Lexical Entailment, Morphological Negation, Factivity, Symmetry/Collectivity, Redundancy, Named Entities, Quantifiers
Predicate Argument Structure	Core Arguments, Prepositional Phrases, Ellipsis Implicits, Anaphora/Coreference Active/Passive, Nominalization, Genitives/Partitives, Datives, Relative Clauses, Coordination Scope, Intersectivity, Restrictivity
Logic	Negation, Double Negation, Intervals/Numbers, Conjunction, Disjunction, Conditionals, Universal, Existential Temporal, Upward Monotone Downward Monotone, Non-Monotone
Knowledge	Common Sense, World Knowledge

Table A.1: Taken from [129] Table to test captions and labels

A.2 Gaussian Embeddings

[14] already used distributional representations to embed concepts from natural language processing into embeddings. [126] considers creating a Gaussian representation for each word token, resulting in Gaussian word embeddings.

This can be interpreted as a probabilistic and continuous extension to the discrete point vectors calculated by word2vec and GloVe. Each word is represented by a Gaussian distribution in high-dimensional space, with the aim to better capture uncertainty when doing algebraic operations on the word-embeddings, resulting in a model which expresses the relation between words and their representations better. Using KL-divergence as a distance metric, it can also express asymmetric relations more naturally than dot-products or cosine similarities do. Specifically, the newly emerging embedding for word w can be modelled by the tuple

$$\theta_w = (\mu_w, \Sigma_w)$$

and the probability density function of the word embedding x_w is expressed by

$$p(x; w) = \mathcal{N}(x; \mu_w, \Sigma_w) \tag{A.1}$$

where x is a point in the embedding space.

Given that the probability density function is given by a gaussian distribution, the loss functions which is minimized is adapted accordingly:

$$L_m(w, c_p, c_n) = -min(0, m - E_\theta(w, c_p) + E_\theta(w, c_n)) \tag{A.2}$$

where E is the energy function which calculates the distance measure between word w, a positive context word c_p (a word that co-occurs with the word w, and c_n), and a negative-sampled context word c_n (a word that does not co-occur with the word w, usually randomly sampled from the set of all words in the vocabulary $\mathcal{V}$). [126] proposes two different ways to compute the energy function, using a symmetric similarity, and an asymmetric similarity measure.

The energy function can be **symmetric**. In this case, the authors use an inner product of two gaussian distributions defined as:

$$E_\theta(w, c) = \int_{x \in \mathcal{R}^d} f(x)g(x)dx \tag{A.3}$$

$$= \int_{x \in \mathcal{R}^d} \mathcal{N}(x; \mu_w, \Sigma_w)\mathcal{N}(x; \mu_c, \Sigma_c)dx \tag{A.4}$$

$$= \mathcal{N}(0; \mu_w - \mu_c, \Sigma_w + \Sigma_c) \tag{A.5}$$

which results in a nice closed form representation.
The energy function can also be **asymmetric**. In this case, the authors refer to the KL-divergence, resulting in the following energy function minimized.

$$-E(w_i, c_j) = D_{KL}(c_j||w_i) \tag{A.6}$$

$$= \int_{x \in \mathcal{R}^d} \mathcal{N}(x; \mu_{w_i}, \Sigma_{w_i}) \log \frac{\mathcal{N}(x; \mu_{c_j}, \Sigma_{c_j})}{\mathcal{N}(x; \mu_{w_i}, \Sigma_{w_i})} dx \tag{A.7}$$

$$= \frac{1}{2} \left(\mathrm{tr}\left(\Sigma_i^{-1} \Sigma_j\right) + \left(\mu_i - \mu_j\right)^\top \Sigma_i^{-1} \left(\mu_i - \mu_j\right) - d - \log \frac{\det\left(\Sigma_j\right)}{\det\left(\Sigma_i\right)} \right) \tag{A.8}$$

The asymmetric distance property allows to capture relations such as "y entails x", without necessarily implying "x entails y".
Assuming training was conducted by one of the two presented models, one can use the resulting model parameters for two words x_{w_1}, x_{w_2} to calculate the uncertainty - which intuitively models the similarity between the two words w_1 and w_2. This is analogous to calculating the dot product between the two distributions $P(z = x^T y)$ through the close-form formulation of

$$\mu_z = \mu_x^T \mu_y \tag{A.9}$$

$$\Sigma_z = \mu_x^T \Sigma_x \mu_x + \mu_y^T \Sigma_y \mu_y + \mathrm{tr}\left(\Sigma_x \Sigma_y\right) \tag{A.10}$$

We then get an uncertainty bound, where c denotes the number of standard deviations away from the mean. This uncertainty bound can be interpreted as a hard cutoff to where one concept ends.

$$\mu_x^\top \mu_y \pm c \sqrt{\mu_x^\top \Sigma_x \mu_x + \mu_y^\top \Sigma_y \mu_y + \mathrm{tr}\left(\Sigma_x \Sigma_y\right)} \tag{A.11}$$

We can learn the parameters Σ and μ for each of these embeddings using a simple gradient-based approach, where we set constraints on

$$\forall i \quad \left\|\mu_i\right\|_2 \leq C \tag{A.12}$$

$$\forall i \quad mI < \Sigma_i < MI \tag{A.13}$$

These constraints can be used by using a Laplacian regularizer in the loss function which is then optimized using a gradient based approach. The method shows competitive scores to the Skip-Gram model, although usually

only with minor improvements depending on the benchmark-dataset.

A.3 Long Short-Term Memory

LSTMs are sequential recurrent neural networks [52]. LSTMs are an extension of the recurrent neural network *RNN* [106], but introduces a mechanism to solve the problem of error in the backpropagating gradient, and a "sotrage" mechanism which allows part of the RNN cell to be non-changed throughout backpropagation update mechanisms.
The internal cell of the LSTM includes the following mechanisms

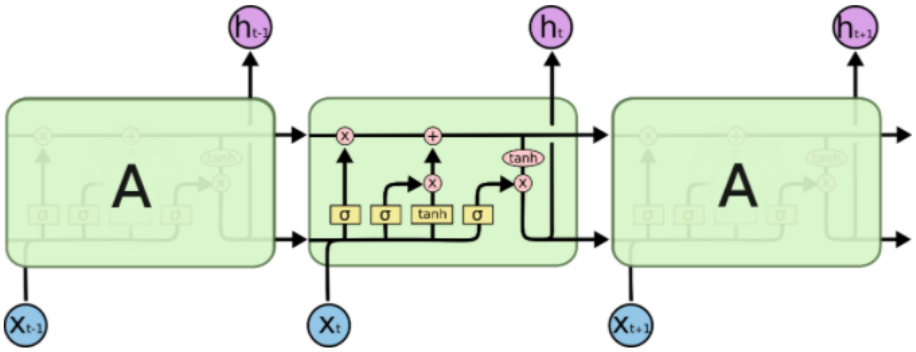


Figure A.1: Figure taken from [5]. The internals of the LSTM cell, and the recurrent flow which is repeated for three consecutive timesteps $t-1, t, t+1$. The LSTM produces an hidden representation at every steep h_{t-1}, h_t, h_{t+1} given some inputs x_{t-1}, x_t, x_{t+1}.

Notice that the flow has a direction, and that in this case, the flow moves towards a positive direction of the sequence (i.e. from $t-1=0$ to $t+1=2$ for example).
Specifically, LSTMs [52] are good at estimating the probability of word $w^{(t)}$ occurring given a previous history of words $w^{(t-1)}, \dots w^{(1)}$. In practice, the LSTM is a popular choice for sequence modeling tasks, as it can very well capture the following probability distribution.

$$p\left(w_1, w_2, \dots, w_T\right) = \prod_{k=1}^{T} p\left(w_k | w_{k-1}, \dots, w_2, w_1\right) \tag{A.14}$$

A.4 GLUE

Single Sentence Tasks

The Corpus of Linguistic Acceptability **CoLA** [134] consists of english sentences, where the task of the model is to predict whether or not the sentence is a valid english sentence. The Matthews correlation evaluates the model from a score of -1 to 1, where 0 corresponds to uniformly random guessing. The test set includes out of domain sentences.

```
label:1
The professor talked us into a stupor.
label:0
The professor talked us.
```

The Stanford Sentiment Treebank **SST-2** [113] consists of sentences from moview reviews and human annotations of their sentiment. This is a binary classificaiton (positive / negative) task.

```
label:1
comes from the brave , uninhibited performances
label:0
excruciatingly unfunny and pitifully unromantic
```

Similarity and paraphrase tasks

The Microsoft Research Paraphrase Corpus **MRPC** [37] is a corpus of sentence pairs automatically extracted from online news sources, with human annotations whether the sentences in the pair are semantically equivalent. Here, an F1-score is taken, because the class-sets are imbalanced.

```
quality:1
Prosecutors filed a motion informing Lee they intend to seek the death penalty.

He added that prosecutors will seek the death penalty.
quality:0
A former teammate , Carlton Dotson , has been charged with the murder.

His body was found July 25 , and former teammate Carlton Dotson has been charged in
```

The Quora Question Pairs **QQP** [57] dataset is a collection of question and answer pairs from the website Quora. The task it to check whether a pair of questions are semantically equivalent. Again, the F1-score is taken as a measure because the class-sets are unbalanced.

```
My Galaxy ace is hang?
Why are the people on Staten Island are racist?
0
```

```
Where can I learn to invest in stocks?
How can I learn more about stocks?
1
```

The Semantic Textual Similarity Benchmark **STS-B** [27] is a corpus of pairs of news headlines, video and image captions. Each pair is annotated with a similarity score from 1 to 5, and the task is to predict these scores. The evaluation is done using Pearson and Spearman correlation, as the determining factor is the relative ranking amongst scores.

```
The man is riding a horse.
A man is riding on a horse.
5.000
A man is playing a guitar.
A girl is playing a guitar.
2.800
```

Appendix B

Other lines of work

B.1 Clustering

Affinity Propagation was first introduced by [41]. Affinity propagation takes as input measures of similarity between pairs of data points. Real-values messages for multiple iterations are exchanged between data point until clusters start to emerge. Initialization happens by assigning the same class to close-enough points, message passing and the choice of which candidate sample to take on happens as defined by a *preference* threshold.

DBScan is a density based clustering algorithm [38] which initiates by finding representative samples that are in regions of high density, and expands clusters from there on. Due to the density criteria, the resulting clusterings don't have to be convex as is the case with k-means for example. **HDBScan** is a hierarchical extension of the DBScan algorithm [25]. **Optics** [80] is another extension of DBScan which keeps the cluster hierarchy for a variable neighborhood radius.

MeanShift is a centroid-based clustering algorithm [32]. After cluster-centers are randomly initialized, the centroids are updated by the mean of the cluster until convergence in the cluster-assignments is reached. A final post-processing step removes near-duplicate clusters.

B.2 Metric Learning and Disentanglement

Althogh difference metric spaces can be considered, we will be refering to Euclidean spaces for simplicity and unless otherwise stated.

Metric learning is the concept of learning a new space in which distances between data samples
Distance metric learning approaches the problem of learning a similarity metric.
We introduce two sets of data-sample pairs, S and D:

$$S = \{(x_i, x_j) : x_i and x_j should be similar\} \tag{B.1}$$
$$D = \{(x_i, x_j) : x_i and x_j should be dissimilar\} \tag{B.2}$$

As is also mentioned in [87], the proximity relation can also be captured through *relation triplets*

$$R = \{(x_i, x_j, j_l) : x_i is more similar to x_j than x_l\} \tag{B.3}$$
$$\tag{B.4}$$

The general notion of a distance in the euclidean space can be defined as

$$d_E(\boldsymbol{x_i}, \boldsymbol{x_j}) = \sqrt{(\boldsymbol{x}_i - \boldsymbol{x}_j)^\top (\boldsymbol{x}_i - \boldsymbol{x}_j)} \tag{B.5}$$

or equivalents as the l_2-norm

$$d_E(\boldsymbol{x}_i, \boldsymbol{x}_j) = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 \tag{B.6}$$

Most distance learning techniques propose different ways of learning a Mahalanobis distance [75]

$$d_M(\boldsymbol{x}_i - \boldsymbol{x}_j) = \sqrt{(\boldsymbol{x}_i - \boldsymbol{x}_j)^\top \boldsymbol{A}^{-1} (\boldsymbol{x}_i - \boldsymbol{x}_j)} \tag{B.7}$$

Because calculating d^2 is computationally simpler, and d^2 maintains most mathematical properties as calculating d, often d^2 is used for downstream calculations (i.e. optimization, distance measure), rather than d.
In the case of the Mahalanobis distance, A is a linear operator such as a matrix. However, A can also be generalized to a nonlinear operator by using the following formulation of the distance metric

$$d_M^2(x_i - x_j) = (x_i - x_j)^\top A^{-1}(x_i - x_j) \tag{B.8}$$
$$d_M^2(x_i - x_j) = (L(x_i - x_j))^\top (L(x_i - x_j)) \tag{B.9}$$
$$d_M(x_i - x_j) = \|L(x_i - x_j)\|_2 \tag{B.10}$$
$$d_M(x_i - x_j) = \|Lx_i - Lx_j\|_2 \tag{B.11}$$

where $A = L^T L$ and L can now be any function $L \in \mathbf{R}^n -> \mathbf{R}^d$. Whenever, $d < n$, the data is projected to a lower dimensional space d, which results in dimensionality reduction, which however breaks the convexity property if L is not invertible. In comparison to a dimensionality reduction such as PCA, it has been obsereved that learning a similarity measure can sometimes be more effective [28].
Specifically, the data is projected into another The authors in [87] argue that due to this property of measuring similarity between pairs, the metric learning models are able to generalize better across classes.

Non-Deep Metric Learning

Although we focus on deep metric learning during our experiments, due to the additional flexibility that deep neural networks provide, a good understanding of the underlying mechanisms is useful.
[117] provides a good introduction into the literature of metric learning algorithms.
We first define by a proper definition of a distance
Definition 1 *Let X be a non empty set. A distance or metric over X is a map $d : X \times X -> \mathbb{R}$ that satisfies the following properties:*

1. *Coincidence: $d(x, y) = 0 \iff x = y$, for all $x, y \in X$*
2. *Symmetry: $d(x, y) = d(y, x)$ for all $x, y \in X$*
3. *Triangle inequality: $d(x, z) \leq d(x, y) + d(y, z)$ for all $x, y \in X$*

The ordered pair (X, d) is called a metric space. We will be assuming a euclidean metric space unless otherwise specified.
Because the coincidence property is not always important to us, but because word-embeddings are more interested in measuring relative distances, we will also consider mappings konwn as *pseudoinstances* which relax the coincidence property to merely fulfill $d(x, x) = 0$. In the d dimensional euclidean space, the set of *Mahalanobis distances*, which is parametrized by positive semidefinite matrices are useful.
Using above similarity, dissimilarity and relation triplets, an optimal distance d from a set of distances $\mathcal{D}$ can be found

$$\min_{d \in \mathcal{D}} l(d, S, D, R) \tag{B.12}$$

where l is one of the loss metrics shown below.

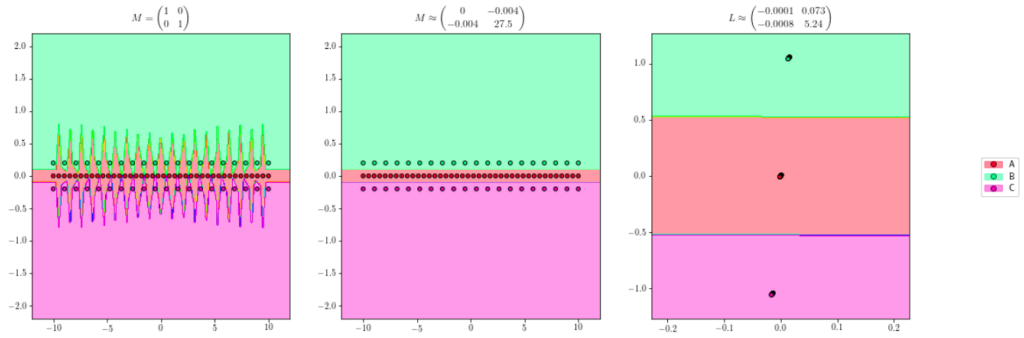


Figure B.1: Taken from [117]. The individual tiles show the kNN prediction regions by color for every point in the image. Using the unmodified euclidean distancee, this would result in classification regions on the left. The reader can see, learning an appropriate distance, the classification is much more effective (middle). Finally, the dimensionality is also reducable with this dimension while still matching the classification accuracy (right).

Although we will not go into too much detail, we will outline some algorithms that [117] mentions in their work.
We will first start with dimensionality reduction techniques, then move to nearest neighbors classifiers, follow with nearest centroids classifiers and finally go over some information theory based algorithms. Notice that for almost each one of these methods, kernelized versions are available.

Dimensionality Reduction techniques include principal component analysis *PCA*, LDA ANMM

Deep Metric Learning

Besides the non-deep proposed in the work above models , in[63] also introduces the deep generalization to these frameworks.

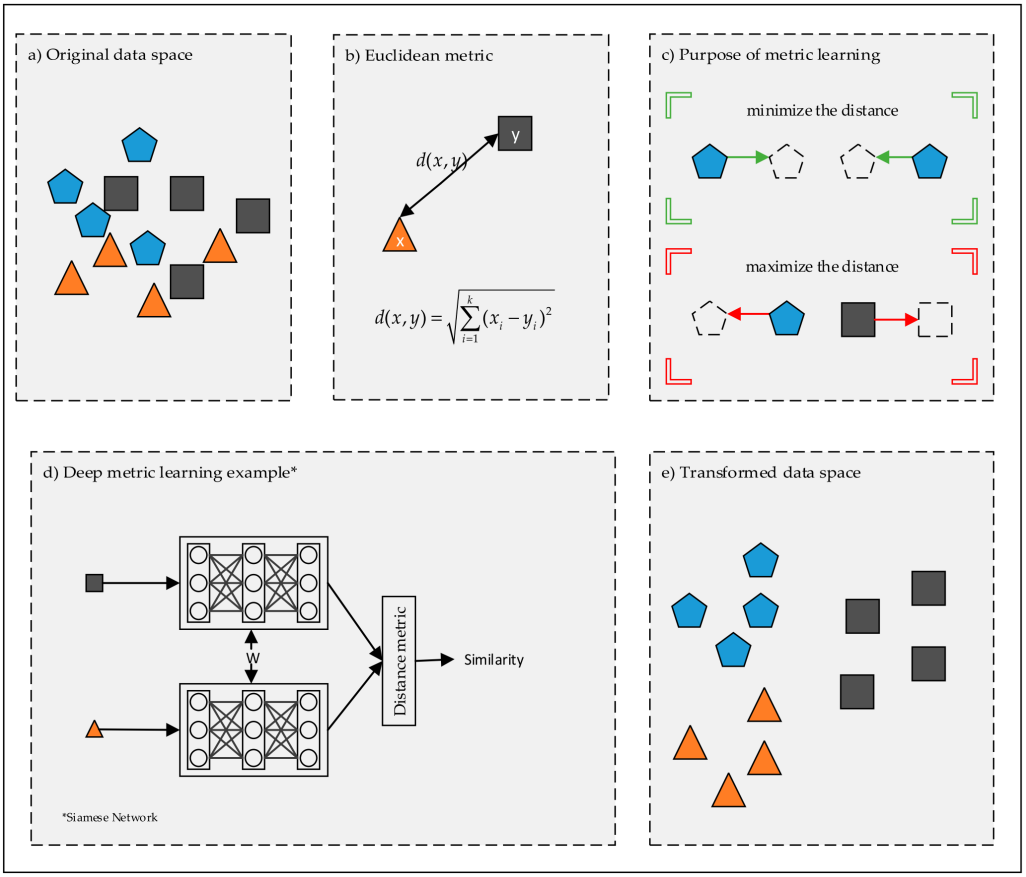


Figure B.2: Taken from [63]. An illustration of deep metric learning. The space is transformed in such a way, that similar object are closer to each other, and dissimilar objects are moved away from each other.

The most prominent model is the siamese network [22], [30], [46], where the network consists of two identical sub-networks joined at their outputs. We provide an example the architecture looks that looks as follows:

```
class Siamese:

    def __init__(input_dim, latent_dim):
        self.fully_connected = nn.Linear(input_dim, latent_dim)

    def forward(input1, input2):
        latent_1 = self.fully_connected(input1)
        latent_2 = self.fully_connected(input2)

        distance = torch.sqrt(
```

```
            torch.sum((latent_1 - latent_2) * (latent_1 - latent_2))
        )

        return distance, latent_1, latent_2
```

Two samples are ingested by the neural network simultaneously. Both samples are passed through identical weights. Training occurs by minimizing the contrastive loss.

$$L_{\text{Contrastive}} = (Y)\frac{1}{2}\left(D_W\right)^2 + (1-Y)\frac{1}{2}\left\{\max\left(0, m - D_W\right)\right\}^2 \tag{B.13}$$

where we follow the convention that $Y = 1$ whenever the given sample-pairs are of the same class, and $Y = 0$ whenever the given sample-pairs are from different classes, and D_W is the output of the above neural network. Training can be highly unstable, and as such, the learning rate must be properly set, batches must have a balanced amount of both similar, and dissimilar pairs. Finally, training can fail if a single batch includes multiple domains of data.

For the triplet networks [53] which makes use of relation triplets, including an input X, a point similar to the input X^p and a point dissimilar to the input X^n, the following loss can be used for minimization.

$$L_{\text{Triplet}} = \max\left(0, \left\|G_W(X) - G_W\left(X^p\right)\right\|_2 - \left\|G_W(X) - G_W\left(X^n\right)\right\|_2 + \alpha\right) \tag{B.14}$$

where α forms a distance margin, similar to the length of the support vector in SVMs.

As the authors suggest, using a triplet logic results in more stable training and bigger margins between similarity classes. Analogous logic goes for quadrupel loss etc., but it is unsure to what extend these extensions increase performance. This general idea can also be extended to angular distance (using cosine distances), and non-euclidean spaces, however, we will not go into further detail on this as this is not the focus of this work.

Although we will not go into further detail, other loss functions include histogram loss [124], the structured loss [116] n-pair loss [114], magnet loss [105], angular loss [130], quadruple loss [89], clustering loss [115], hierarchical triplet loss [44] and the multi-similarity loss [132].

B.2.1 Embeddings for translation

A lot of work is done in the field of analysing embeddings for translation tasks, to further mitigate the black-box behvaior of neural network.
This includes [64] who use singular value canonical correlation analysis to compare hidden representations of language models between different languages. To arrive at a sentence-representation, they do mean-pooling over the outputted embedding-vectors of a sentence. They find interesting behavior in the arrangement of context-embedding in the Transfoerm-Big architecture proposed in the [125] paper, which is also inherently used in BERT and GPT. Interestingly, they show that different languages do not project into similar spaces, thus making zero-shot learning tasks between languages difficult.

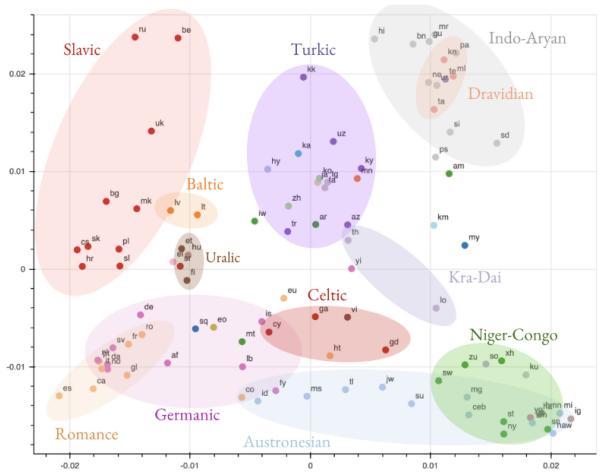


Figure B.3: From [64], visualizing clustering of the encoder representations of all languages, based on ther SVCCA similarity.

[65] uses a cyclic loss to apply unsupervised machine translation suign monolingual corpora only. Here, sentence-embeddings are learned over time, which follow the cyclic loss constraint, and minimize the sunsupervised translation loss.
Other work tries to disentangle the different linguisitc, visual and audio-based features by supplying multi-modal input at once [73]. A translation model is learned which can not only translate between different domains, but also between different modalities (i.e. from image to audio, audio to text, and any such combination).

B.3 Compressing the non-lexical out

Motivation

Experiment setup

We conduct three experiments.
First, we use a simple algorithm to what extent we can disentangle the semantic space within a single sampled word.
The, we use a simple algorithm to what extent we can disentangle the semantic space within multiple sampled words, where we assume that the underlying semantics should again be mutually exclusive (i.e. the sampled words are not independent towards each other.
Finally, we use a simple algorithm to what extent we can disentangle the semantic space within multiple sampled words, where we don't inject any assumptions upon the underlying semantics of the words, i.e. the words are independent towards each other.

Results

Appendix C

More Results

C.1 Finding the best clustering model

Quantiative Evaluation

Table C.1: The maximum ARI score achieved during hyperparameter optimization for the different models as described by experiment for $k = 50$ and $n = 500$.

Clustering Model	ARI Score
Affinity Propagation	0.001
DBScan	0.000
HDBScan	**0.328**
MeanShift	0.004
Optics	0.000

Table C.2: The maximum ARI score achieved during hyperparameter optimization for the different models as described by experiment for $k = 50$ and $n = 1000$.

Clustering Model	ARI Score
Affinity Propagation	0.000
DBScan	0.139
HDBScan	**0.271**
MeanShift	0.005
Optics	0.070

Table C.3: The maximum ARI score achieved during hyperparameter optimization for the different models as described by experiment for $k = 100$ and $n = 1000$.

Clustering Model	ARI Score
Affinity Propagation	0.000
DBScan	**0.215**
HDBScan	0.359
MeanShift	0.003
Optics	0.000

Qualitative Evaluation

```
                                                - -
              heavy withdrawals from the British bank Northern Rock reignited concern
                                                - -
credit card and other consumer loans, forcing the bank to set aside $3.4 billion
  by a mainland Chinese commercial bank in a U.S. bank
                               Investors fear the bank will be forced to write down
                         investors hoped that the bank had disclosed the
                                                - -
```

Figure C.1: Some figure

```
                                              - -
                           I would expect the bank by the trail to the left of the road to have been broken
                                             - -
The current slowed and swirled alongside a mud bank where cows had trodden to the water
                                       But the bank had positioned itself well
   provide plant scientists and farmers with a bank of genes
                         Upstairs at the large bank of cashiers
                                              - -
```

Figure C.2: Some figure

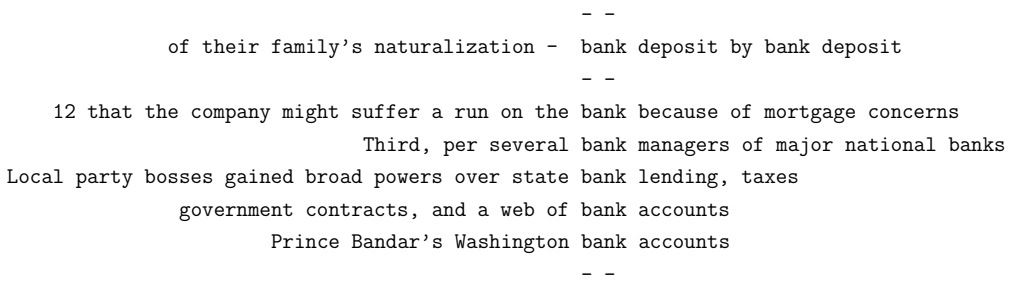


Figure C.3: Some figure

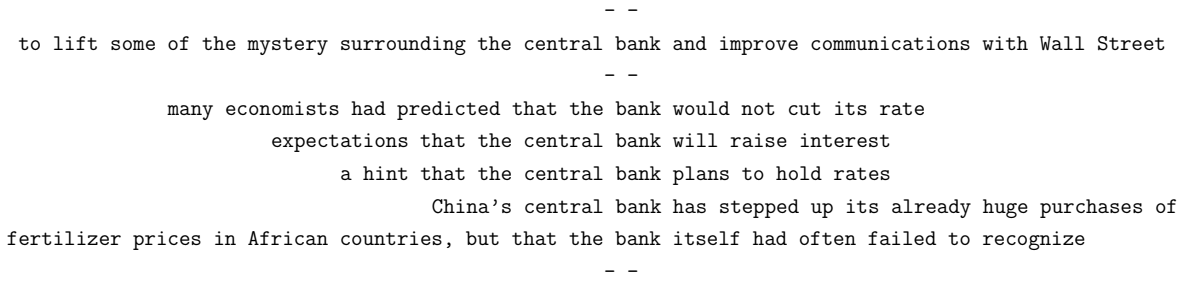


Figure C.4: Some figure

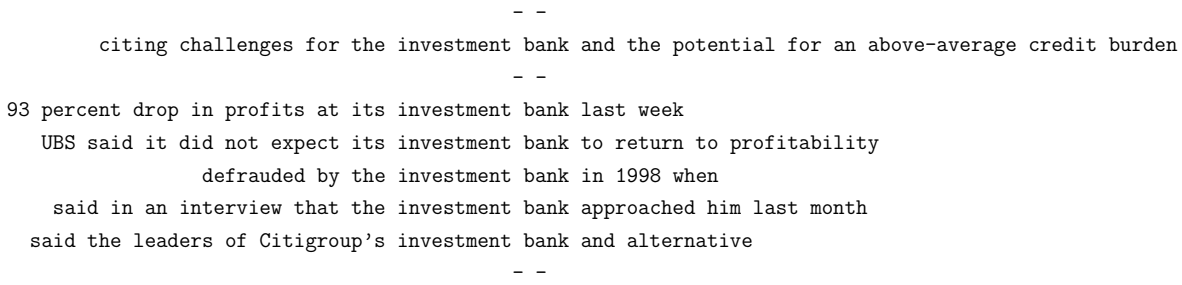


Figure C.5: Some figure

EXAMPLES FOR KEY CLUSTERING

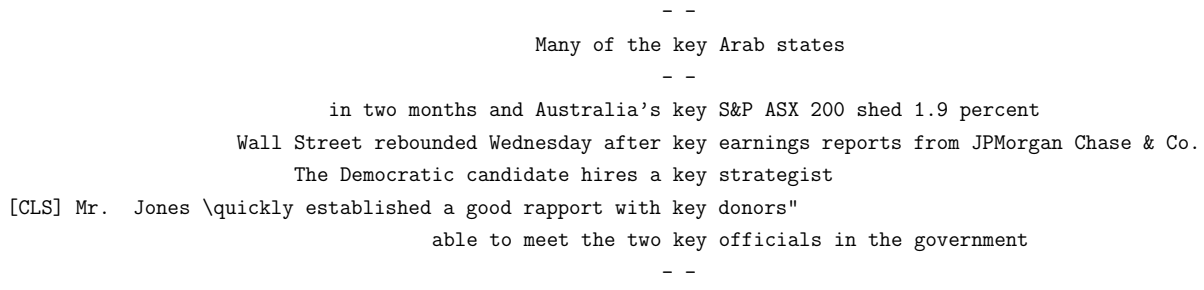


Figure C.6: Some figure

- -
former president of Trinity College, who played a key role in designing the test
- -
seen in the West as a key yardstick of the fairness of an election
treat ... as a key factor in its decisions about regulatory issues
which policy makers have called the key test for deciding whether to lower interest rates
9. 11. as a key element in pitch meetings
- -

Figure C.7: Some figure

- -
Interstate 5 is a key route connecting Southern and Northern California
- -
A key piece of new functionality for Ops Center
Youssef Squali at Jefferies & Co. says two key factors are driving the stock up
transforming connection with believers is a key element of evangelical Christianity
What would you say was the key element of your management style that allowed you to stabiliz
is a key indicator of retailer performance
in the West the key players were not a small group of intellectuals reading Greek
- -

Figure C.8: Some figure

- -
times change and technology advances, the key to the city symbolizes
- -
And an official, five-and-three-quarters-inch-long gold-plated pewter key to prove it
but it is small enough to fit onto a key chain
In it lay three keys on a key chain in the shape of a red speedbo
- -

Figure C.9: Some figure

- -
The Red Sox will now have all their key players from their 2007 championship team
- -
Mike Green scored 12 points and had a key assist in overtime as No. 22 Butler beat Virginia Tech
or taking the chance of losing a key player to injury
But they never led, could not get a key basket at crucial times and played like a team
but Cam Long stole the ball near the top of the key and ran out the clock
- -

Figure C.10: Some figure

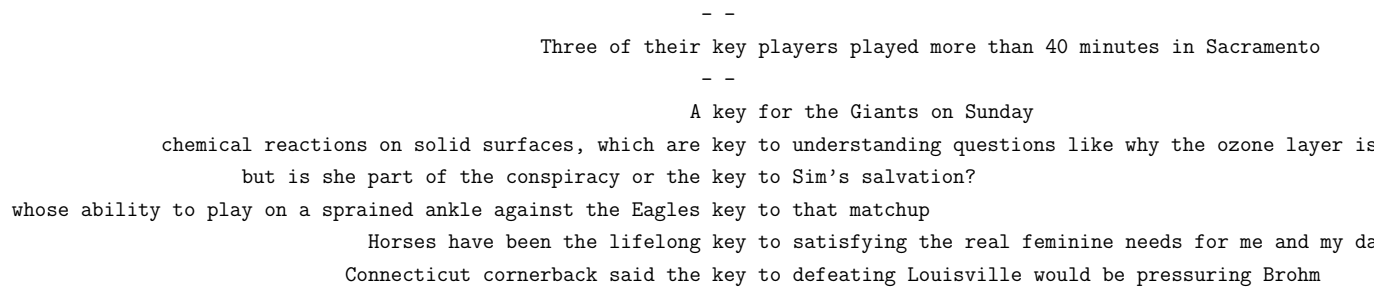


Figure C.11: Some figure

C.2 Frozen Verbs

The words under investigation include attack, act, die, kick, sand, address, aim, act, address, back, bear, block, catch, charge, crack, double, face, head, march, order, play, roll, saw, tie, train, treat, value, visit, wake, work, zone, act, address, aim, answer, back, balloon, bank, battle, bear, bend, blast, block, break, brush, catch, challenge, charge, cheer, color, cook, crack, curl, cycle, dance, design, die, double, doubt, dust, echo, end, estimate, face, finish, fish, flood, fool, frown, garden, glue, guard, guess, hammer, hand, head, hug, insult, iron, kiss, laugh, loan, love, man, march, milk, object, order, paddle, peel, permit, play, pop, practice, produce, punch, question, quiz, rhyme, rock, roll, run, sand, saw, skate, smell, surprise, thunder, tie, time, toast, trace, train, treat, trick, use, vacuum, value, visit, wake, walk, water, wish, work, x - ray, yawn, zone, cut, break, well, down, run, round, table, bank, cold, good, mouse, was, key, arms, thought, pizza, made, book, damn.

Appendix D

Applications

D.1 Using embeddings in other domains

[123] apply a word2vec type training methodology on chemical publications to encode relationships between chemical compounds.

D.2 Using embeddings in translations

Although word-vectors can be used for a variety of tasks, we will focus on some varieties of how these vectors are used for machine translation (of human languages) tasks.
Amongst the most prominent method is the MUSE model introduced by [33]. Here, a mapping W is found using either an unsupervised methodology by minimizing the batch-size cosine-distance between sampled word-vectors between two languages A and B, or a supervised methodology is devised using the procrustes algorithm.

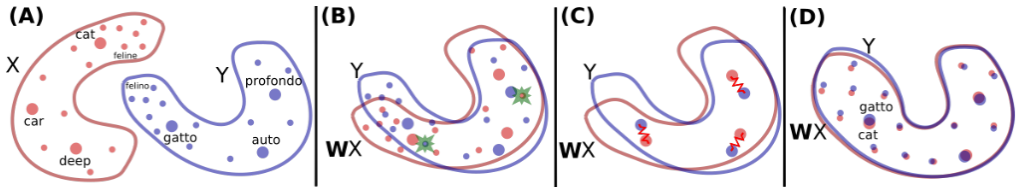


Figure D.1: Taken from [33]. Each

Specifically, the following loss-function optimizes over the space of possible mapping matrices W^*:

$$W^\star = \underset{W \in O_d(\mathbb{R})}{\operatorname{argmin}} \|WX - Y\|_{\mathrm{F}} = UV^T, \text{ with } U\Sigma V^T = \mathrm{SVD}\left(YX^T\right) \quad \text{(D.1)}$$

Unsupervised training is conducted using a discriminator and a mapping function, where the discriminator's task is to identify the origin-distribution, and the mapping function is supposed to fool the discriminator, resulting in a mapping function which maps all word-emebddings into a common global embedding space.
A generalization of these word vectors to point-vector probability-distributions are captured by [8]. Here, the point-vectors are interpreted as point-delta-distributions in the $d+1$ dimensional space. Common optimal transport methods are coupled with the Gromov-Wasserstein loss to find an orthogonal mapping which maps from vector-space X to vector space Y, and thus from the word-token of one language to another. One way to achieve this is to use the supervised loss-metric and solve the procrustes problem, finding correspondences between the columns of X and columns of Y through:

$$\min_{\mathbf{P} \in O(n)} \|\mathbf{X} - \mathbf{PY}\|_F^2 \quad \text{(D.2)}$$

with $O(n) = \left\{\mathbf{P} \in \mathbb{R}^{n \times n} | \mathbf{P}^\top \mathbf{P} = \mathbf{I}\right\}$. The resulting algorithm achieves similar performance to MUSE while requiring only a fraction of the computational cost, being able to get competitive results on CPU.
The unsupervised approach deals with finding a transportation map T that fulfills

$$\inf_T \left\{ \int_{\mathcal{X}} c(\mathbf{x}, T(\mathbf{x})) d\mu(\mathbf{x}) | T_{\#}\mu = \nu \right\} \quad \text{(D.3)}$$

,
where μ and ν are the point-delta distributions describing the word-embeddings in the probability space. The relaxed Kantorovich's formulation is then finally used to reduce this problem to finding a set of transportation plans in a polytopes.

$$\Pi(\mathbf{p}, \mathbf{q}) = \left\{\Gamma \in \mathbb{R}_+^{n \times m} | \Gamma \mathbb{1}_n = \mathbf{p}, \Gamma^\top \mathbf{1}_n = \mathbf{q}\right\}$$

Finally, the cost function

$$\min_{\Gamma \in \Pi(\mathbf{p}, \mathbf{q})} \langle \Gamma, \mathbf{C} \rangle$$

is minimized to find an optimal transportation plan.
Another extension is the work by

Word2Vec

BERT conditions on the rest of the input sentence.
BERT uses words, subwords and individual characters (in total 30'000) that are then used as tokens.
Idea is to do the following: Concepts (and thus words), are represented across multiple contexts. We can create probabilistic word-embeddings by sampling from a corpus the context of a specific word. From multiple samples of the context-embedding-vector, we take the mean and stddev, or create a GMM if there are multimodal logic (we can check this multimodality by running some sort of rejection sampling algorithm). Then we have a probability density function (one for each language), and can map one into another.
Perhaps we could split up too high-variance embeddings to multimodal embeddings, depending on their use-cases.
This allows for interpretability in polysemy sentences.
Not using more complex flows implies that the flow itself is not the bottleneck (they probably tried out other flows as well).
Are the individual word-embeddings going to be one big blob with high variance, or is it going to be a multi-modal distribution...?
Another task we may look at is, from a given word-embedding, sample it's context. Not entirely sure how to do this with BERT and co.
At what point do we overfit, and at what point do we generalize?

Artetxe bilingual token matching through unsupervised machine translation

- Input is cross-lignual word-embeddings - Build an unsupervised phrase-based statistical machine translation system - Derive phrase-based embeddings from input-word-embeddings by taking top 400'000 bigrams and 400'000 trigrams -¿ take arithmetic mean of word-embedding - score top 100 close phrases using softmax cosine similarity - generation of synthetic parallel corpus using this approach - Then use FastAlign to use parallel corpus to align words -