Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.

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 22, 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: 34’606

Signed:

Date:

Abstract

Word and contextual word embeddings can be considered as the fundamental unit in machine learning for natural language applications. For a variety of downstream tasks, any shortcomings in the embeddings will propagate on to the performance of the target task and worsen performance of the target task. A good understanding of the embeddings is required to address these shortcomings. Given that language representation models such as BERT only capture a black-box function between input and output, our aim is to understand how the subspace organization of context embeddings relates to linguistic features. We focus our analysis on *semantics*, which captures meaning, as this is one of the most important features of language. We analyse the BERT language model, as it provides a good balance between popularity, close to state-of-the-art performance, and generalizability to other modern language models as it is based on the transformer architecture.

Our main contributions are the following: (1) We test for linear separability and ability to partition the space into semantic clusters within the sampled BERT vectors. We show that while linear separability is possible, it appears difficult to cleanly cluster the set of contextual word embeddings produced by BERT into different semantic classes as defined by WordNet. We conclude that BERT organizes the embedding vectors not only by semantics, but rather by context alone. This can easily introduce undesirable properties such as strong sentiment, leading to considerable bias in downstream tasks. (2) We analyze the relationship between part-of-speech and semantics. We show that there exists a strong correlation in languages, which is reflected in the embeddings produced by modern language models captures. This affects the conclusion of some related work which analysed language models that capture semantics, but which did not account for the correlation between multiple linguistic features such as part-of-speech and semantics. (3) Finally, we introduce additional model-parameters for words whose sampled embedding vectors have high variance over the embedding space. We investigate what downstream tasks are most affected by and use this as a proxy to understand what information is captured the most by the directions with high variance.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

One of the goals of Natural Language Understanding (NLU) is to develop algorithms that are able to operate with human language. However, it is not yet apparent that machines can think, or even operate with semantic concepts. In most of these cases, the underlying language model uses heuristics or black-box functions which simply maps an input text to output text merely satisfying the constraints set up by the target task.

These shortcomings can be addressed by modelling written text in such a way that linguistic features are captured. Models can then be used for other machine learning tasks, referred to as *downstream 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 a meaningful correspondence to the concepts captured by these embeddings.

Language representation models, also called Language Models (LM) are a category of NLU models that can be used to generate word embeddings, sentence embeddings, and also intermediate representations that can be used for downstream tasks. Most commonly, language models create vector representations by taking into consideration the context in which a word token occurs. Because embeddings - due to their strong usage in downstream tasks - can be considered as a fundamental unit in NLU, any shortcomings will influence the performance of the target task. Improving the way these embeddings capture linguistic features has strong implications for a wide category of tasks including Named Entity Recognition (*NER*), sentiment understanding, and automatic language-translation. However, most of the modern LMs are so complex that they are considered black-box models.

We believe that providing a better understanding of the underlying principles will pave the way for future research to better address these issues. The semantic attribute of a word is amongst the most important linguistic features. It tells us what we mean when we say `dog` or `car`. It can also be ambiguous, when we use words such as `bank`, requiring us to look at the context to understand what meaning of `bank` one refers to. Due to the significance of meaning in language, we will focus our analysis on how modern LMs capture semantics. Notice that the relation between language and thought is a much more fundamental and philosophical one. Here, we assume that language is just an expression of thought, but that the two concepts are not equivalent.

Our main contributions are the following:

1. We test to what extent the semantic subspace has multiple modes which correspond to meaning in language. We test this by analysing linear separability between semantic classes, and test for clusterability in these structures. We show that although linear separability is possible, it is nearly impossible to find modes in the data which correspond to semantic classes.

2. We analyze the relationship between the two linguistic features of part-of-speech and semantics. We show that these show a strong correlation in languages and that this correlation is reflected in modern language

models. This observation affects the conclusion made in some related work that analysed language models for semantics, but that did not account for the correlation between part-of-speech and semantics.

3. Finally, we introduce additional parameters into language models for words whose sampled embedding vectors have high variance over the embedding space. We investigate what downstream tasks are most affected and use this as a proxy to understand what information is captured the most by the directions with high variance.

In this work we focus on BERT, as it provides a good balance between popularity, close to state-of-the-art performance, and generalizability to other modern language models.

Our findings show that although BERT does not capture a simple semantic subspace, but organizes the embedding vectors more broadly by 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 particularly sentiment is more strongly captured in modern language models, allowing for language stereotypes and leading to stronger cases of bias in language models.

We will start with summarizing background work (Section 2), show existing analyses on the inner workings of BERT (Section 3), conduct tests that do not modify the BERT model (Section 4) and make modifications to the embeddings in BERT to understand what effect this has on downstream tasks (Section 5). We will finally draw conclusion and show potential for future work (Section 6). We adopt the terminology used in background and related work, and thus also refer to BERT as a language model, even though it cannot capture arbitrary probability distributions such as $p(w_t|w_{t-1}, \ldots, w_1)$.

Chapter 2

Background

[50] 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 entails. This representation is often captured in the form of a word token w, such as `bank` or `cat`. The *distribution hypothesis* of language was introduced in [50], and describes the idea that semantics and other linguistic features for a word w can be purely extracted through the context that word w occurs in. 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 $m_{\text{cat-animal}}$. [50] 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 trivial and may even be considered in a philosophical context [51, 144], 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, ranging from phonological features, morphological and syntactic features to semantic features. We will first introduce some linguistic features relevant to this work.

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 that of `bank` can entail is $|M_{\text{bank}}| = |\{m_{\text{bank-financial instution}}, m_{\text{bank-sea bank}}\}| = 2$ (In fact, there are many more meanings of `bank`, but we will keep the examples concise). The true numbers vary on the chosen granularity and the language at hand. The choice of granularity is a non-trivial question as this determines at what point one concept stops and another concept starts. These definitions 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 [2]. In this work, we will be using the spaCy python package [57] whenever we need to predict the part-of-speech class of a word-tokens and limit ourselves to high-level part-of-speech which consist of nouns, verbs, and adjectives. For a more extensive list of linguistic features, please refer to section A.1 in the Appendix.

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. We will go over some forms that tokens can take on, starting with the most trivial ones. In a sentence such as

Example 1

`The man travelled down the road.`

the set of word-tokens would constitute of $\{\text{down}, \text{man}, \text{road}, \text{the}, \text{traveled}\}$. 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.

Characters can be considered as another set of tokens language. Looking at example 1, 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

$$[t, h, e, '''', m, a, n, '''', t, r, a, v, e, l, e, d, '''', d, o, w, n, '''', t, h, e, '''', r, o, a, d]$$

Choosing characters as the basic token levels has shown considerable success and popularity in language modeling [126] and translation [72].

Byte pair encodings [43] presents another popular mechanism to tokenize sentences, and also shows success in the case of language modeling and translation [115]. 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, similar to Huffman encoding. 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 a byte-pair token. The main idea behind this tokenization is to construct a vocabulary of frequent subwords, allowing to generalize to unseen words.

WordPiece tokenizer [146] 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 Example 1 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 (Equation 2.1). 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 describe 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 [44] have also been conducted.

Distance: Our goal is to build an embedding space where the distance between word-embeddings corresponds to the relation between words (i.e. the semantics entailed by their tokens). We introduce the concept of *distance* $d : \mathcal{X} \times \mathcal{X} \mapsto [0, \inf)$, which captures a relation between two 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)
3. $d(x, y) \leq d(x, z) + d(z, y)$ (triangle inequality)

One consequence of the above properties 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 to word c. For convenience, whenever we input a, b, c into the distance function $d(\cdot, \cdot)$, the word-embeddings for the corresponding word-tokens shall be used to numerically calculate the distance. Also, please notice that we did not include 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 $\mathbf{X}$. One can now follow the idea of the *distributional structure* of language [50], which states that all linguistic features for a word w can be captured by looking at its context words. In terms of word vectors, we want to maximize the probability that w occurs in the context window of w' 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' a *context word* for w, and consequently we want to maximize the probability for any two co-occurring words w, w':

$$p\left(w|w'\right)$$

In many cases, maximizing the likelihood probability corresponds to minimizing the global distance between co-occurring words.

$$\forall w, w' : \qquad \max p\left(w|w'\right) \iff \min d(w|w') \tag{2.2}$$

Please note that we do not state that the minimization problem is the dual of the maximization problem.

Exploiting the distributional structure of language: Our goal is to learn distances between words by exploiting the distributional structure of a set of sentences. A set of sentences that we use for this task is called a *corpus.* One of the early formalizations of representing the distributional structure of words is explained in [13], which argues that a sequential statistical model can be constructed to estimate the true posterior of a sequence of tokens. In it's most naive form, this would imply that we can estimate the probability of a sequence of words $\mathbf{w}$ as

$$p(\mathbf{w}) = \prod_{t=1}^{T} p\left(w^{(t)}|w^{(t-1)}, \ldots, w^{(1)}\right) \tag{2.3}$$

where $\mathbf{w} = w^{(1)}, \ldots, w^{(T)}$ is the sequence of words and $p(\mathbf{w})$ 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

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

Fixing the context window to n words for each computation (i.e. convolution), introduces the concept of *n-grams.* An n-gram is a sequence that consists of n tokens. n-grams can also occur with characters, where n characters are fed into 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 that 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

$$p(\mathbf{w}) = \prod_{t=1}^{T} \prod_{\triangle \in \mathcal{I}} p\left(w^{(t)}|w^{(t+\Delta)}\right) \tag{2.5}$$

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

Loss Objective: Given that computational power is limited, we are usually interested in learning a parametrized model that captures this underlying distribution behind equation (2.5). 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.

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

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

Intuitively the above distributions implement the idea behind [50] very well, specifically the property that the meaning of a word is captured by the context that it occurs in. 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. Training a model using the skip-gram approach does not change the search space of models $\theta \in \Theta$. The optimal model for the CBOW formulation will *likely* be different from the optimal model for the skip-gram formulation, i.e. $\hat{\theta}_{\text{CBOW}} \neq \hat{\theta}_{\text{skip-gram}}$ will likely be different, however. 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, the loss captured through equation (2.2) would

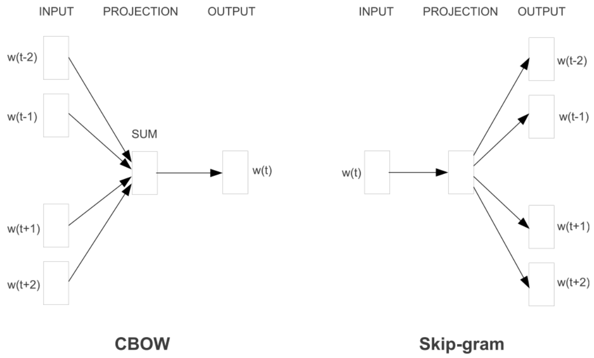


Figure 2.1: Figure taken from [88]. The CBOW model (left) predicts the current word based on the context. The skip-gram model (right) predicts surrounding words given the current word.

be reformulated into

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

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 parametrization 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 routine on the loss term and updating the weights accordingly.

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 in equation (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 equation (2.2) and minimizes words within the same context.

Here, we introduce a *log-bilinear* [56] model where the log-probability is defined as

$$\text{log} p_\theta(w|w') = \langle \mathbf{x}_w, \mathbf{x}_{w'} \rangle + b_w + \text{ const.} \tag{2.8}$$

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. The model parameters consist of the word-embeddings $\theta = \{(x_w, b_w) \in \mathbb{R}^{d+1} | \forall w \in V\}$. We can use a gradient-based loss-minimizing (notice the log-term) 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 window 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. The distance would be minimized if all word-embeddings would collapse onto a single point. No term forces unrelated words to move away from each other, although this is a property that we are interested in. Such a term would result in more isotropic vector models, that has also shown practical performance [40].

Word2Vec

One of the most prominent word vector models is proposed in [88, 89]. Here, a neural network with a single embedding layer can be trained to transform one-hot-vectors $\in \{0,1\}^{|\mathcal{V}|}$ (representing words w in a vocabulary $\mathcal{V}$) into an embedded representation $x_w \in \mathbf{R}^{d+1}$. As the neural network consists of a single layer, this boils down to a linear embedding matrix W acting as a lookup table. Both a continuous bag of words 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

$$\begin{aligned} \mathcal{L}(\theta; \mathbf{w}) = \sum_{t=1}^{T} \sum_{\Delta \in \mathcal{I}} [& \\ & b_{w^{t+\Delta}} + \langle \mathbf{x}_{w^{(t+\Delta)}}, \mathbf{x}_{w^{(t)}} \rangle \\ & - \log \sum_{v \in \mathcal{V}} \exp \left[\langle \mathbf{x}_v, \mathbf{x}_{w^{(t)}} \rangle + b_v \right] \end{aligned} \tag{2.9}$$

As one can see, the loss function takes as input the bilinear loss from the basic model. This is complemented by a regularizing term (last line), such that random samples are not all next to each other in the embedding space produced. This idea is referred to as *negative sampling.*

Backpropagation is used to optimize the network's weights. Also, highly frequent words are optionally subsampled and their frequency is re-weighted using the formula

$$P(w_i) = 1 - \sqrt{\frac{t}{n(w_i)}}$$

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

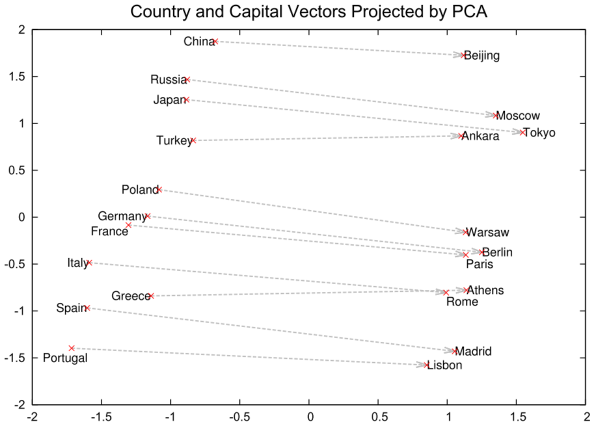


Figure 2.2: Figure taken from [89]. A 2-dimensional PCA projection of the 1000-dimensional skip-gram vectors of countries and their capital cities. The proposed model is able to automatically organize concepts and learn implicit relationships between them. No supervised information was provided about what a capital city is.

GloVe

For the global vectors for word representation *GloVe* [99], 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.

$$\begin{aligned}\mathcal{H}(\theta; \mathbf{N}) &= \sum_{i,j} f\left(n_{ij}\right) (\underbrace{\log n_{ij}}_{\text{target}} - \underbrace{\log \tilde{p}_{\theta}\left(w_i | w_j\right)}_{\text{model}})^2 \\ &= \sum_{i,j} f\left(n_{ij}\right) \left(\log n_{ij} - \langle x_i, y_j \rangle\right)^2\end{aligned}$$

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 should not have too much weight for the loss-term (there is a cutoff at some point, as the number of word-occurrences increase exponentially). This way, rare words are considered noise and slowly cancelled out.

Similar work to word2vec and GloVe exists (i.e. fastText [19]). Further extensions include word embeddings as probability densities using a Bayesian skip-gram approach [22], where the resulting model architecture is similar to the word2vec neural network, but includes a probabilistic latent module. Another line of work [147] aims to exploit the formal properties of Wasserstein distances to learn static word embedding. Please refer to section A.2 in the appendix where we present Gaussian Embeddings [132] for an explanation of what such a probabilistic extension could look like.

2.2.2 Contextual Word Embeddings

In contrast to static word embeddings, contextual word embeddings rely not only on the target word w as input, but also require a subset of the context $[c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_T]$ around w 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 also the subsequent context words $c_{\text{subsequent}} = w^{(t-d-1)}, \ldots, w^{(1)}$.

$$(c_{\text{previous}}, w, c_{\text{subsequent}}) \mapsto (x_w) \in \mathcal{X}^{d+1} \tag{2.10}$$

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

Example 2

`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

$$p(w^{(t-d)}|c_{\text{previous}}, w^{t-d}, c_{\text{subsequent}}) = p(w^{(t-d)}|w^{(t)}, \ldots, w^{(t-d+1)}, w^{(t-d)}, w^{(t-d-1)}, \ldots, w^{(1)}) \tag{2.11}$$

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 recent years.

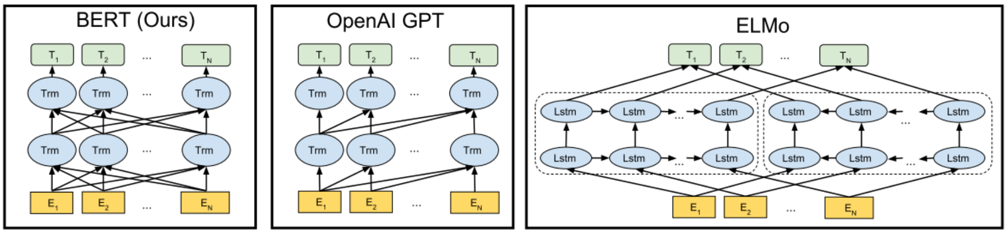


Figure 2.3: Figure taken from [37]. BERT uses a bidirectional transformer, which reads in all the input for both the previous and subsequent context. OpenAI GPT (next section) uses a left-to-right Transformer, reading in only the previous context. ELMo uses a bidirectional LSTM which naturally captures a single direction per LSTM, thus both previous and subsequent context is read.

2.2.3 ELMo

One of the simplest models to produce contextual word embeddings, is the *Embeddings from Language Models* (ELMo) model proposed by [100].

ELMo contains the Long Short-Term Memory *LSTM* memory cell as its 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. $[\ldots, x_{t+1}, x_t, x_{t-1}, \ldots]$), 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

$$\sum_{k=1}^{T} \left(\log p \left(w_k | w_{k-1}, \ldots, w_1; \Theta_x, \vec{\Theta}_{LSTM}, \Theta_s \right) \right. \tag{2.12}$$

$$\left. + \log p \left(w_k | w_{k+1}, \ldots, w_T; \Theta_x, \overleftarrow{\Theta}_{LSTM}, \Theta_s \right) \right) \tag{2.13}$$

where p is the output of an LSTM model parametrized by its 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. The forward LSTM reads in the previous context c_{previous} whereas the backward LSTM reads in the subsequent context $c_{\text{subsequent}}$. 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 contextual word 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 a 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 [65]. This pre-trained language model would then be extended by another linear layer or RNN, 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 frozen and updated during fine-tuning. Thus, only the weights of the task-specific parameters are optimized.

2.2.4 The Transformer Architecture

Although ELMo creates contextual word embeddings, the performance is limited to LSTMs. 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

$$\begin{aligned} p(w_1, \ldots, w_T) = p(w_T | w_{T-1}; h_{T-1}, \theta) \\ \ldots p(w_t | w_{t-1}; h_{t-1}, \theta) p(w_{t-1} | w_{t-2}; h_{t-2}, \theta) \\ \ldots p(w_1; \mathbf{0}, \theta) \end{aligned} \tag{2.14}$$

where h_i is the forward-propagation vector outputted by the LSTM cell at timestep i, which are supposed to "store" information. 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 [131] (depicted in Figure 2.4) addresses this shortcoming. It introduces the concept of attention for use by modern language models, which can model relations between any two input tokens w_i, w_j with $i \neq j$ and $|i-j| > 1$. For a more detailed treatment of the attention mechanism, we refer the reader to [11]. The attention mechanism takes all the intermediate token representations for the full sample sentence (not exclusively for the previous or subsequent context) and calculates a correlation score between the hidden representations of each timestep. The transformer outputs an intermediate representation that can be converted to softmax probabilities through a linear head.

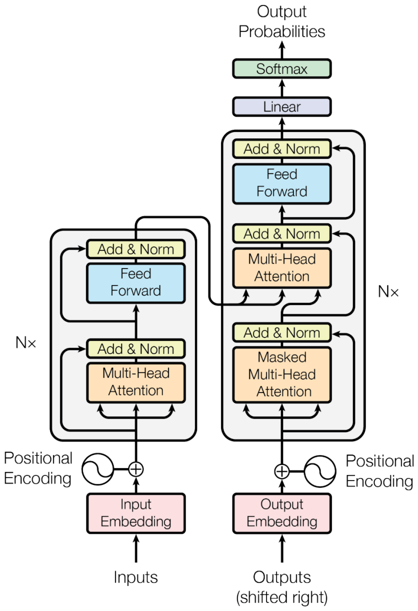


Figure 2.4: Figure taken from [131]. The architecture of the transformer module which encapsulates multiple attention modules.

This allows for the model to explicitly capture and exploit a similarity metric between any two input elements of the sequence. This has implications for the complexity of the model. Instead of calculating the raw probability of (2.3), 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. This decreases the number of computation-steps through which information flows.

2.2.5 BERT: Bidirectional Encoder Representations from Transformers

The Bidirectional Encoder Representations from Transformers **BERT** [37] is a language representation model that makes use of the transformer architecture in contrast to ELMo, which uses recurrent neural networks as its underlying modules. Although there are different versions 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 as is interpreted as the contextual word embedding for each token w_i of the input-sequence s.

The main advantage of BERT is the more direct information flow through the attention mechanism. 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 word w_t and w_k which are passed as the sequence $[w_1, \dots, w_t, \dots, w_k, \dots, w_T]$ 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 $k < t-1$. Stacking the transformer for multiple layers allows BERT to capture a complex structure of language.

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.

Example 3

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

Example from [37]. A sentence where 15% of the tokens are replaced with the [MASK] token. During the first phase of pre-training, the weights of the BERT model are optimized in such a way to predict the true underlying words. The word to be predicted is `the`.

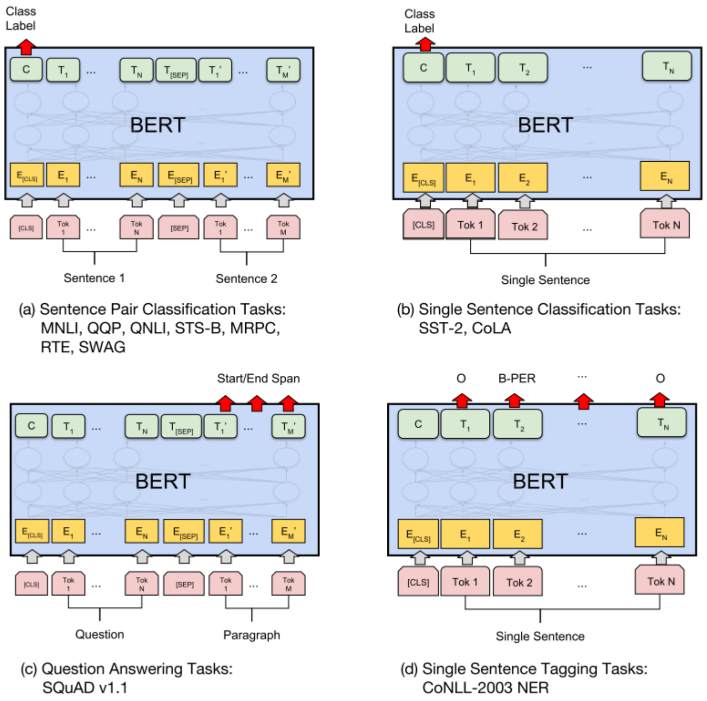
The second pre-training phase is a **Next Sentence Prediction** task, where BERT is supposed to predict a sentence s_i given its predecessor sentence s_{i-1} in a full-text corpus. This is a binarized task, which means that given inputs s_{i-1}, s_r, BERT is supposed to predict whether or not sentence $r = i$ (i.e. whether 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.

Example 4

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

Example from [37]. Two input sequences that 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`.

Pre-training is applied on a 3.3 billion word corpus. For fine-tuning to specific downstream tasks, the learning rate is put lower than during pre-training. The gradient updates all model weights. One can apply an additional linear layer on top of the T hidden representations produced, use RNNs or more complex models, or use specialized outputs in BERT for downstream prediction tasks.



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

(b) Single Sentence Classification Tasks: SST-2, CoLA

(c) Question Answering Tasks: SQuAD v1.1

(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

Figure 2.5: Figure taken from [37]. Ways to fine-tune BERT on different GLUE 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, and 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. These tokens, which are in the vocabulary of the BERT tokenizer, are converted to indices, 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 contextual word embeddings $[h_1, \ldots, h_T] = [x_1, \ldots, x_T]$.

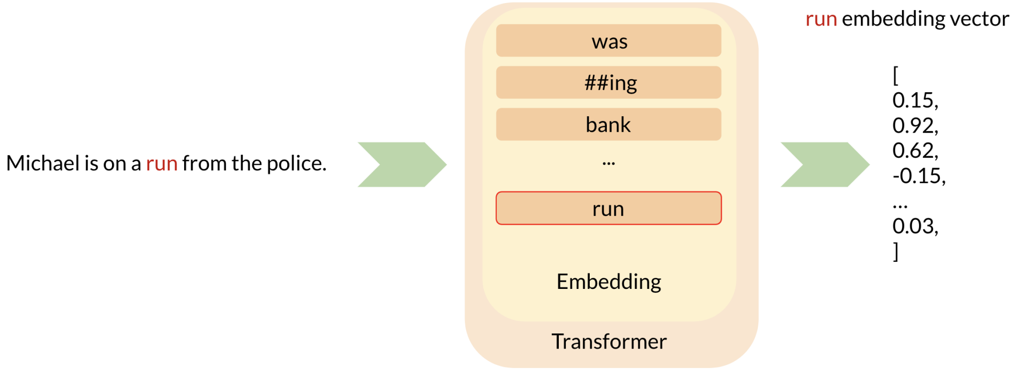


Figure 2.6: The BERT model takes as input a sentence s. The sentence s is converted to a sequence of BERT tokens $t_1, \ldots, t_m$ using the WordPiece tokenizer. Each item in the vocabulary V has a corresponding embedding vector inside the embedding layer of the transformer. This embedding vector is used by the intermediate layers of the transformer, and thus affects the downstream pipeline of the transformer for any subsequent layers of the transformer.

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 language models using transformers. These include GPT [104] and GPT-2 [105], allowing the model to only consider the previous context (instead of the previous and subsequent context). GPT and GPT-2 also include more fundamental modifications to the transformer module, most notably introducing layer normalization [10] at various positions in the module. Other language models which extend BERT models include ALBERT [71] - which is widely considered one of the state-of-the-art transformer models, DisilBERT [114], HUBERT [93], ERNIE [125], RoBERTa [77], SpanBERT [63] and StructBERT [137]. Other lines of work compress representations produced by BERT [116], or interpret BERT as a Markov random field [133], formalizing the process of sampling sentences.

2.3 GLUE benchmark dataset

The GLUE benchmark dataset was first introduced by [135]. Please note that most state-of-the-art models achieve human-level accuracies for standard GLUE benchmarks, which is the reason for the introduction of SuperGLUE [134] with even more advanced language tasks. We will be using the standard GLUE benchmarking, as the basic BERT model does not out-perform human-level scores on this dataset.

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 2.7: Table taken from [135]. 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.

We use an accuracy score for the majority of benchmarks. Some tasks have unbalanced classes or rank metrics, resulting in the F1-score or a correlation metric as the scoring function. The RTE and WNLI datasets are the most interesting tasks for us as these are often considered the most difficult two tasks in the GLUE benchmark. Thus we will give a short outline of these benchmarking tasks. For a description of all other GLUE tasks, please refer to section A.4 in the Appendix.

Inference Tasks

The Recognizing Textual Entailment **RTE** consists of a series of annual textual entailment challenges. Data is combined from RTE1, RTE2, RTE3 and RTE5 [35] [49] [15] [46]. 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`.

Example 5

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

Example 6

```
Money raised from the sale go into Hepburn’s family trust.
Proceeds go to Hepburn’s family.
entailment
```

The Winograd NLI **WNLI** dataset uses the original Winograd Schema Challenge dataset [74], which is a reading comprehension task where the model 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 that test negatively when overfitted. Some example sentences include

Example 7

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

Example 8

```
Mark told Pete many lies, 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 originally introduced in [90]. WordNet is a semantic reference system similar but not identical 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. The first version of WordNet contains 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. 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 these shortcomings through the concept of a *lexical matrix*.

Lexical matrix: The expression *word* is often referred to as both an utterance and its associated concept. [90] 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*. Figure 2.8 depicts 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 encodes multiple meanings, implying that the word-form is polysemous. If multiple entries exist for a single row, the two words express the same underlying concept, and thus the two words-forms are synonyms.

WordNet refers to sets of synonyms as *synsets*, which are a collection of word-forms that together determine a single meaning. WordNet represents a

Word			Word	Forms	
Meanings	F_1	F_2	F_3	$\dots$	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 2.8: Table taken from [90]. 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.

Part of Speech	Definition
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”

Figure 2.9: Example output for WordNet 3.1 noun propositions for the word `bank`. In total, 18 different concepts are recorded.

semantic unit through such synsets. synonyms are words, where one word is interchangeable for another. Because words in a synset are interchangeable due to their synonymous nature, WordNet organizes words into nouns, verbs, adjectives, and adverbs, as the syntactic rules of language must also stay conform. The authors mention that synonyms should be best thought of a continuum along which similarity of meaning can be graded. However, through the introduction of synsets, the authors determine similarity in terms of a binary event, something which is either present, or not present. The two tables 2.9 and A.2 show examples of how WordNet 3.1 introduces different semantic classes for the words `bank`. For a similar table covering the different semantic classes of the word `was` please refer to section A.5 in the Appendix.

2.4.1 SemCor dataset

The SemCor 3.0 corpus is an aggregation of the various Brown corpora [41], where each noun, adjective, and verb is tagged with their respective ground-truth WordNet 3.0 sense. SemCor was part of the early WordNet project, initially introduced in [92]. We provide some examples to give an idea of what the SemCor dataset looks like. Looking at the example sentence

Example 9

```
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 WordNet semantic class ids (first 10 items only) are

Example 10

```
None, Texas.noun.0, halfback.verb.1, None, does.verb.6,
not.adverb.0, even.adverb.10, know.verb.1, None, team.noun.0
```

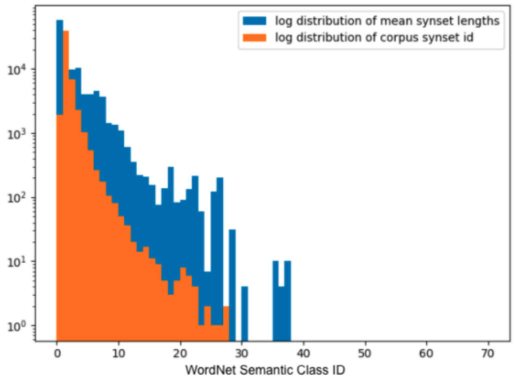


Figure 2.10: A cumulative plot over all words with WordNet senses within SemCor 3.0 and their respective frequencies. The SemCor data is biased. Words with a low WordNet sense index (i.e. close to 0) occur more often than words that have a high WordNet sense index (i.e. above 5). There would be no bias if the two distributions would overlap. The skew could be a natural effect of how word lower WordNet indecies are assigned to frequent commonly used words.

Chapter 3

Related Work

Previous work analyses to what extent language models capture sentiment, part-of-speech, and semantics. We will present a few relevant papers for each of these categories.

3.1 Creating Synsets from Static Word Embeddings

[98, 18, 109, 9, 36] provide examples by which semantic embeddings or synsets are created using knowledge graphs or static word embeddings.

[98, 18] use the Chinese Whispers algorithm initially proposed in [17] to cluster for synsets in word2vec vectors and the JoBimText framework [18] specifically. [36] uses 3-grams and creates a graph by drawing edges between words (nodes) that co-occur in the same phrase. For static word vectors, an ego-network is created for each concept in the vector space whose correlation matrix $G(X)$ is the adjacency matrix for a resulting subgraph $\tilde{G}$

Generalizing these approaches, one arrives at synsets as follows:

1. Learn static word embeddings.
2. Build a graph of nearest neighbours based on vector similarities.

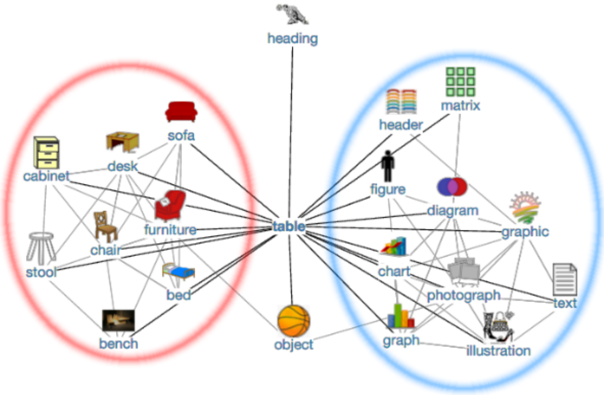


Figure 3.1: Figure taken from [98]. An ego-network of the word `table` is created. Then, clustering is applied on the ego-network, to identify different semantic sets *synsets*.

3. Cluster word senses using ego-network and applying the Chinese Whispers algorithm.
4. Aggregate the word-vectors with respect to the induced senses

The biggest limitation lies in the word vectors or underlying data structure, as this already needs to capture the relevant information for clustering.

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

Table 3.1: Table taken from [98]. 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.

[109] skip the clustering step and makes use of existing WordNet semantic class definitions, averaging the word vectors of synsets to arrive at synset vector representation. Such methodologies prove to be useful in data-mining and document classification problems [108]. Notice that if we have words that consist of multiple tokens, popular methods include applying a simple mean, max, or min pooling over all token-embeddings (see [20, 7, 84]) to arrive at a word-embedding. This is true for both contextual and static word embeddings.

3.2 Quantifying word sense disambiguation

Datasets have been proposed to benchmark word sense disambiguation performance for static word embeddings [24, 53]. [102] addresses the problem of a missing benchmark dataset for *contextual word embeddings* to test contextual word 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 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`):

Example 11

```
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
```

Example 12

```
The expanded window will give us time to catch the thieves.
You have a two-hour window of clear weather to finish work.
label:True
```

A more extensive list of datasets separated by knowledge-based approaches and distributional-based approaches can be found in [26, 76, 95]. Generally, it is assumed that occurrences in numerous different contexts implies polysemy. [26] argues that about 80% of all words in WordNet 3.0 are monosemous, with less than 5% having more than 3 senses, underlining this property for practical effectiveness.

3.3 Semantic subspace inside BERT

There are mixed results when it comes to showing semantic structures in the contextual word embeddings produced by BERT. We want to emphasize that most work deals with measuring the difference between two distinct words $w_1 \neq w_2$, thus not always measuring phenomena such as polysemy.

3.3.1 Geometric analysis

[40, 86] use tools such as silhouette scores, and other measurements based on sampled contextual embedding vectors to draw conclusion on the semantic subspace of embedding vectors produced by BERT.
[40] investigates the question of whether contextual word 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. Models analyse include ELMo, BERT and GPT-2. [40] argues that in all models, contextual word embeddings are not isotropic, and thus not uniformly distributed with respect to direction. Instead they are anisotropic, occupying a narrow cone in the vector space. In GPT-2 alone, two randomly chosen words a cosine-similarity close to 1. 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. Finally, the SemCor corpus is used 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 w contextualized representations for a given layer. It gives us an upper bound on how well a static word embedding could replace word's contextualized representations and is calculated by

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

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

If both intra-similarity and self-similarity are low, then the model contextualizes words in that layer by giving each one a context-specific representation that is still distinct from all other word representations. Low self-similarity implies a nuanced contextualization. The authors conclude that all the language models considered are anisotropic in their sampled contextual word embeddings. Due to this anisotropic property, the evaluations need 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 distributed across the full vectorspace. The authors argue that this is inherent to the process contextualization. [40] further show that stopwords, such as `the`, `of` and `to` 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. In the analysis it is shown that, 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, [40] takes the first principal component for each contextual word embedding (for different words $w_1, \ldots, w_N$) and benchmark this using tests designated for static word embeddings. They show that for layers 1 and 2 of BERT, these word-embeddings often outperform word sense disambiguation benchmarks.

[86] 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 contextual word embedding produced by BERT. Oddly, they find out 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 contextual word embedding than the inputted word token does. This is strongly manifested in contextual word embeddings with changing parity of the sentence position, i.e. when w_i where i is odd / 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 analyses 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 contextual word embedding space. [86] analyses if words with the same WordNet semantic class are projected to the same space inside BERT when sampled in different contexts. They make use of the silhouette score [112], and a clustering scheme of words - where a cluster is defined by words in the same semantic classes. Similar words are not projected to similar

points in the context space. The authors use the Wiktionary dictionary [5] as an alternative to semantic classes defined by WordNet. They conclude that polysemous words result in degraded cohesion scores and decreased silhouette scores, and thus are more spread across the embedding space. Despite all this, transformer-based LMs show great success for downstream tasks. Finally, [86] argues that the cosine similarity is a good distance measure for semantic similarity between contextual word embedding vectors, analogous to what was observed with static word vectors [88]. The authors continue with a sentence-pair similarity analysis, as well as an analysis of the sentence-level structure, which is not of strong interest in our investigation.

We repeatedly find, that the correlation between certain linguistic features, such as part-of-speech and semantics, is not well noted when conclusions are drawn and examples are shown.

3.3.2 Probing for semantics

A *probing task* refers to adding a (linear) modification to an underlying module (such as BERT), and then measuring performance for a downstream task. This performance is used as an indication of how well the underlying module captures the concept scored by the task.
[117, 75, 142] revise modifications to BERT which allow BERT to predict sense labels or perform word sense disambiguation *WSD* tasks. [117] creates 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 and use the CoNLL 2005 [28] as well as the 2012 dataset [103] for labelled training, superseding prior non-BERT models with this simple BERT extension for the tasks of semantic role labelling. Similarly, [75] revise a modification to BERT resulting in *SenseBERT*. This predicts not only the word-form of the masked words but also the WordNet semantic classes (also called supersenses). A linear layer is appended on top of the final layer of BERT, making use of the final hidden representation. Training is done in a supervised fashion using the SemCor dataset and the

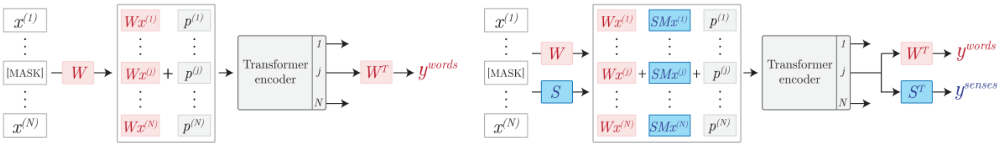
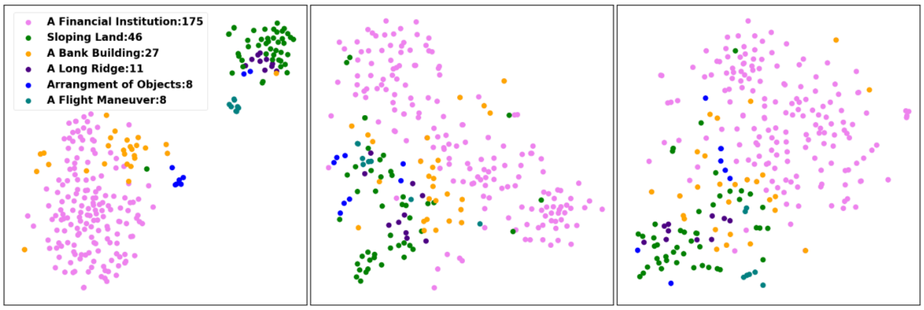


Figure 3.2: Figure from [75]. Standard BERT (left) and the SenseBERT adaptations (right), which include an embedding-encoder and an embedding-decoder specifically for WordNet senses.

Word in Context *WiC* task.
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 without 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.

Alternatively [142] uses a nearest neighbour classifier to measure the performance 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. Prior work on using recurrent neural networks to classify sense labels exists [66]. The authors argue that because for most NLP tasks, the fact that contextual word embeddings 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 contexts, we may assume polysemy of its meaning [91]. The authors use the concatenation of the averaged wordpiece vectors of the last four layers as the contextual word embedding.
The authors conclude that word-sense-disambiguation *WSD* can be surprisingly effective using solely contextual word embeddings. Different SensEval datasets are used for benchmarking, which also including SemCor.



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

Figure 3.3: From [142]. T-SNE plots of different senses of `bank` and their contextual word embeddings. The legend shows a short description of the different WordNet sensees and the frequency of occurrence in the training data.

[32] 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 consider what causes the separation between clusters. Given Figure 3.4, one cannot reliably say that the part-of-speech feature *causes* the partition into clusters. `die` has different meanings when interpreted as a verb and as a noun, which could also be the reason the clusters are so strongly visible.

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 only in marginal benefit. The underlying assumption is that the semantic features are encoded in an underlying subspace of the BERT context vectors. In contrast to [61], [32] the authors also note that this suggests that more semantic information may be available in lower layers.

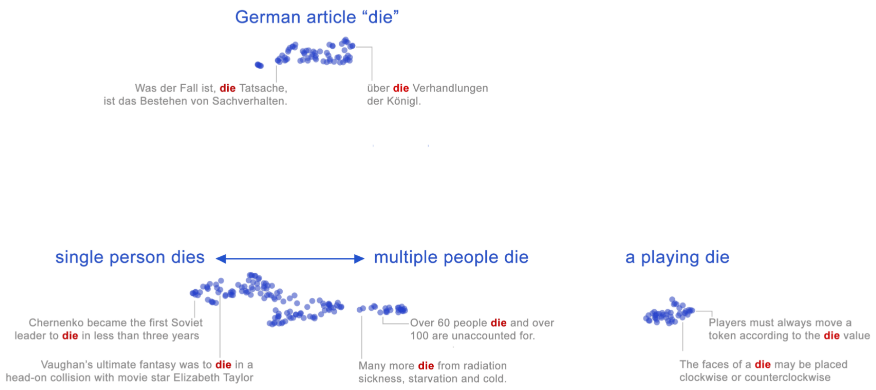


Figure 3.4: From [32]. Embeddings for the word `die` in different contexts, visualized through UMAP. The blue text describes general annotations. Notice that in this case, the two semantic classes `die` also have two distinct part-of-speech tags.

3.3.3 Word sense disambiguation in specialized domains

Semantic class prediction can be extended to other domains. For example, in the medical domain, different tokens can be interpreted as different medicines. `cold` can be interpreted as either a symptom or temperature. [119] uses contextual word embeddings to better extract clinic concepts. Instead of WordNet semantic classes, medical entities are introduced, using medical datasets annotated with clinical concepts and amongst others, making use of BioBERT [73]. The authors demonstrate state-of-the-art results for clinical concept extraction.

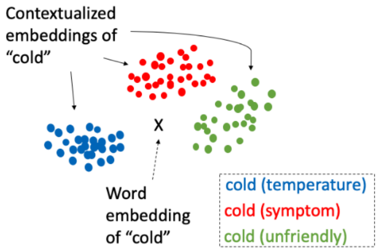


Figure 3.5: Figure taken from [119]. **Fictional** embedding vector points and clusters of `cold`. This is one of the results that we want to arrive at. Specifically, we desire distinct word-embeddings that capture the modes of the underlying probability distribution.

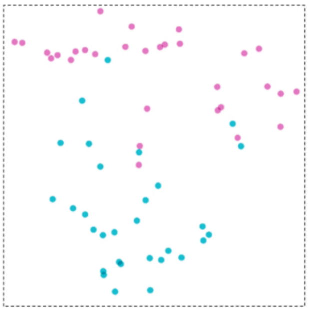


Figure 3.6: Figure taken from [119]. PCA visualizations using embedding vector of `cold` from BERT. The blue data points refer to `cold` as a temperature, whereas the red data points refer to `cold` as a symptom.

3.4 Syntactic subspace in BERT

Contextual word embeddings produced by BERT can also be analysed for syntactical properties.

3.4.1 Extracting Parse-Trees

[32, 61] analyse how BERT encodes parse-trees in their attention mechanisms, and the distance between the produced contextual word embeddings.

[32] argues that on a high level, semantic and syntactic features seem to be present in different subspaces. [61] specifies this by arguing that BERT en-

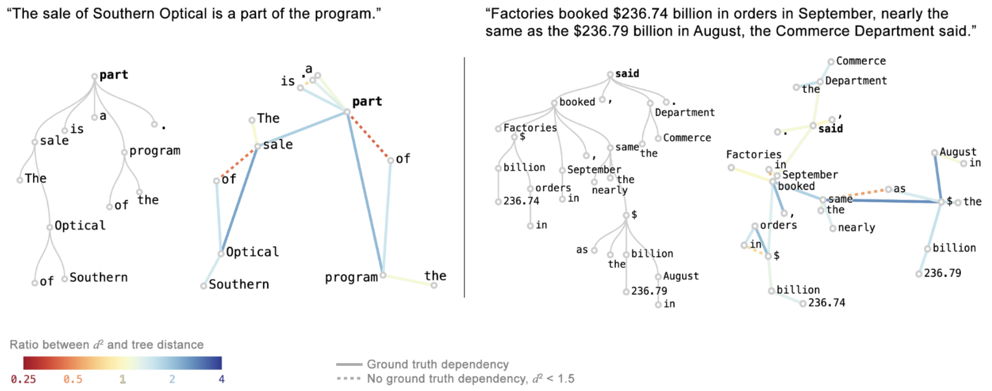

"The sale of Southern Optical is a part of the program."
"Factories booked $236.74 billion in orders in September, nearly the same as the $236.79 billion in August, the Commerce Department said."
part
is
a
.
sale
program
The
of
the
Optical
Southern
said
booked
Factories
$
billion
236.74
orders
in
September
same
nearly
as
August
Department
Commerce
236.79
Ratio between d^2 and tree distance
0.25
0.5
1
2
4
Ground truth dependency
No ground truth dependency, d^2 < 1.5

Figure 3.7: From [32]. Visualizing the embeddings of two sentences after applying the Hewitt Manning probe. Parse tree (left) in comparison to the PCA projection of the contextual word embeddings (right). Small deviations are apparently, but an obvious resemblance exists. The color of an edge determines the squared Euclidean distance.

codes surface features in bottom layers, syntactic features in middle layers, and semantic features in top layers. [32] mentions that there are fine-grained geometric representations of word senses, which are encoded in a relatively low-dimensional subspace. However, this conclusion stems mostly from qualitative evaluation. In slight contrast, [61] cluster the representations (using normalized mutual information) at different layers l of the model, and claim that lower layers encode phrasal information better, which is in slight contradiction to. [32] uses the Penn Treebank [82] to infer a dependency parsing scheme using a linear classifier with output of the transformer context vectors as input. They achieve above 85.8% accuracy for two-class classification, and 71.5% accuracy for multi-class classification. Prior work by [52] finds that BERT encodes a parse tree where the tree distance seems to correspond to specifically to the square of the Euclidean distance. A theorem is presented that shows that a tree with n nodes has a Pythagorean embedding in $\mathbb{R}^{n-1}$. [61] use a maximum spanning-tree approach to arrive at a dependency parsing tree extracted from the attention matrix depicted in Figure 3.8.

[101] do a similar analysis on a 4-layer deep LSTM-based language model. Similar to [61], they find that in general, lower layers mostly capture

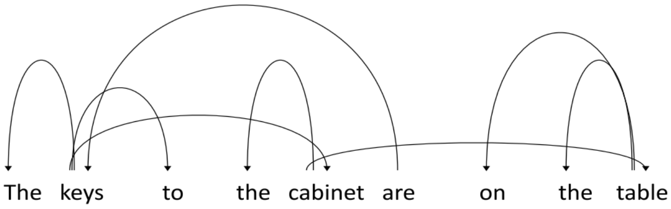


Figure 3.8: From [61]. Dependency parse tree induced from attention head in layer 11 in layer 2 using labelled root `are` as starting node with the maximum spanning tree algorithm

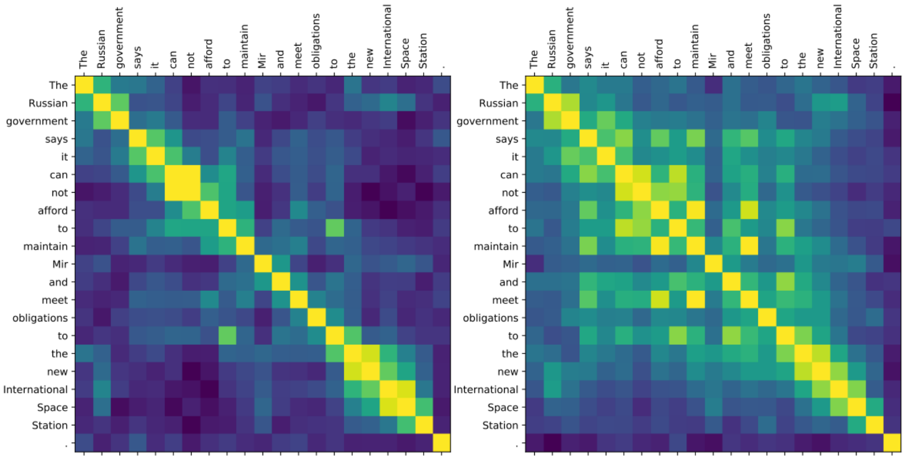
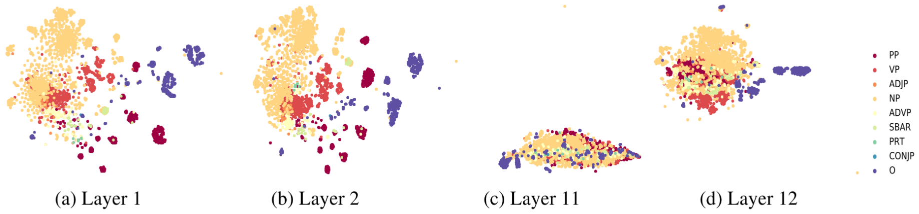


Figure 3.9: From [101]. The authors show different similarity intensities (cosine similarity) between token-pairs using the output of a 4-layer LSTM (left), and the output after the first layer (right).



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

Figure 3.10: From [61]. 2D t-SNE plot of span embeddings computed from the first and last two layers of BERT.

local information, while top layers represent longer-range relationships, as the attention matrix has local spikes. For bidirectional LMs, lower layers tend to place words from the same syntactic constituents in similar parts of the vector space. The authors argue that in bidirectional LMs, semantics is captured only to a small extent, whereas morphology alone is most often used to answer questions on word analogy tasks. The LM architectures learn syntax, as part-of-speech tags are captured at lower levels of the bidirectional LM layers. The authors further build a constituency parser with almost 80% accuracy, by only adding a linear classifier and the span representation onto the lower layers of the bidirectional LMs. A parse-tree is built using a greedy decoding step similar to [64].

3.4.2 Subspace representing part-of-speech, verbs and arguments

[110] aims to devise a verb-clustering and an argument-clustering approach that 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 ELMo and BERT contextual word embeddings. A graph is devised from the set of sampled contextual word 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

$$t_f = \frac{\mu + \sigma}{2} \tag{3.1}$$

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

$$t_a = \mu - 1.5\sigma \tag{3.2}$$

as a higher neighboring threshold is needed as the authors found out empirically. Clustering performance is measured by the number of clusters found, and a purity metric that describes how homogenous the created clusters are. Labels from FrameNet [12] are used for the homogeneity test. [101] 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.

3.5 Sentiment subspace in BERT

Because context also implies sentiment, the context embeddings produced by BERT introduce bias through the corpus that the LM was trained on. Language models can also be used to evaluate how meaning and sentiment for certain words change over time.

3.5.1 Bias

[84] measure social biases in BERT contextual word 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. [84, 25] find that in the embedding space, concepts such as `European American names` are much

more closely related to concepts such as `pleasant` than concepts such as `African American names`, which is a strong indication of bias. [25] follows a similar analysis, focusing much more on the properties of corpora used for training. The authors hint that word-embeddings like word2vec also contain biases, where male names are more biased with career settings than female names.

[84] analyse the social biases that are part of the contextual word embeddings and show that depending on the language model (ELMo, BERT), the amount of social bias deviates. Because contextual word embeddings capture more than just semantics, bias is a natural implication of context vectors. Although not very extensive on the topic of aggregation, the authors use techniques of mean-pooling, max-pooling, and last-pooling to 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 of the corpora also yields inclusion of human-like biases [62] by names, races, male-vs-female.

3.5.2 Change of language over time

This bias can also be seen in how language changes over time. [83] looks at how language models capture the change of meaning for different words 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 contextual 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, [58] quantitatively analyses how meaning changes using corpora from different historical epochs, including the 1890s, 1940s, 1960s, 1970s, 1990s, and 2000s. Specifically, they show strong changes in static pairwise word-vector similarities when trained on differently epoched corpora. They demonstrate through examples of the word *gay* and *alien* that the similarity between certain context-vectors changes through time.

Chapter 4

How is semantics captured through contextual word embeddings produced by BERT?

In the previous section we have seen that although considerable work has been done in quantifying the syntactic subspace produced by BERT vectors, little work has been done in quantitatively evaluating the semantic nature of the outputs of BERT contextual word embeddings with respect to polysemy. As such, we analyse BERT for how it organizes its semantic subspace for a given target word w.

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 that builds a discriminative model of the subspace, where labels are different semantic classes as defined by WordNet. The second experiment aims at identifying a similar subspace organization through an unsupervised algorithm, implicitly identifying a multimodal structure within the subspace we look at. Finally, we check how part-of-speech relates to semantics within the contextual word embeddings produced by BERT. This underlines the importance to distinguish between different linguistic features, especially when these are strongly correlated, something that has previously often led to premature conclusions (section 3). It also helps explain the nature behind some visualizations in 3.

4.1 On the Linear Separability of meaning within sampled BERT vectors

The first experiment takes as input a word of interest w and samples a set of sentences S from a corpus. In this case, the above mentioned SemCor corpus and the news corpus [3] which does not have any WordNet annotated classes are used. To see if there is any structure within BERT vectors corresponding to the different meaning of one word, we ask ourselves whether or not different meaning are located at different extremes of the embedding space produced by BERT, or whether multiple semantic clusters form at different locations of the embedding space with the same WordNet semantic class label. The latter scenario could occur as BERT is overparametrized [68]. We test this hypothesis by training a linear classifier that learns a separating hyperplane between the different WordNet classes for w. If a hyperplane exists with good separating accuracy, it is likely that different semantic concepts are located at different extremes of the embedding (sub-)space.

4.1.1 Experiment setup

To produce a contextual word embedding, we pass 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) through the BERT model. The output of BERT is a set of hidden representations which are interpreted as the contextual word embeddings $[x_{w_1}, \ldots, x_{w_i}, \ldots, x_{w_T}]$, where x_{w_i} corresponds to the contextual word embedding of w_i. This results in a feature matrix X where each row corresponds to the sampled sentence when repeated n times. The columns correspond to the feature-dimensions of the contextual word 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

sentences 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 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 `was`, `is`, `be`, `are`, `more`, `one`, `first`, `only`, `time`. Although these words are not the most intuitive polysemous words (as would `bank` be, for example), the limited "obviousness" should allow this experiment to reject the hypothesis less easily. To work with similar input to the classification algorithms, we apply a standard scalar such that the input data X is normalized around 0.0 with a standard deviation 1.0. We also test if dimensionality reduction techniques improve performance. 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 hyperplanes in the solution space. Projecting X to a lower dimensionality works like a regularizer. This restricts the set of candidate hyperplanes. Note that 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 experimental setup is captured through algorithm 1.

Algorithm 1: Checks sampled BERT vectors for linear interpretability by meaning

```
Input: A target word $w_{\text{target}}$; 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 $w_{\text{target}}$, along with the
  corresponding WordNet meaning-id;
$\mathbf{X} \leftarrow BERT(\mathbf{D})$ i.e. pass each sentence through BERT and retrieve the
  resulting contextual 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 the sklearn [97] implementations for the Standard Scalar, PCA, and Logistic Regression. We use the Hugginface transformers library [145] for the BERT model, and also all subsequent modifications of BERT. The loss function we use is the $l1$ 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 three candidate 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. In high dimensions, there are many possible separating hyperplanes due to candidate model space growing exponentially with the number of dimensions. We refer the reader to Appendix C.1 for more results.

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

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

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 *one*.

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

Table 4.3: Mean and standard deviation of the accuracy of a linear classifier trained on the 4 most common classes of WordNet meanings for the word *was*.

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 due to limited sample sizes. To make sure that no inconsistencies happen during testing, we also analyse the confusion matrices, which all were balanced across class-pairs. 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 at low dimensions.

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. The key takeaways follow a similar interpretation as what principal components are, showing us the features and latent dimensions with greatest differences. Investigating the difference between clusters can show us how different classes are different from each other.

There are different ways to check if a set of vectors is clusterable. This includes theoretical ways to determine clusterability, for example as described in [6, 85]. 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. The experiment on linear separability has shown us that different semantically annotated samples are at different locations of the embedding space. We now want to investigate how easily this clustering can be found, and if the transitioning between the semantic categories is a smooth transition, or constitutes a hard partitioning between context vectors.

4.2.1 Experiment setup

In section 4.1, we try to train a discriminative model. Now we try to come up with a quasi-generative model that is able to detect modalities of the underlying true distribution, which in theory could be used to generate new data samples. Consequently, this task is a more difficult problem than finding a separating hyperplane as done in section 4.1.
One of our goals is to identify semantic clusters within a given set of contextual word embeddings. Implicitly, this requires the algorithm to solve a multi-modality detection problem. This means that we cannot use algorithms such as k-means [78, 80], as this requires the number of clusters to be known prior to running the algorithm. 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 popular and non-parametric algorithms outside the domain of neural networks. Please refer to section B.1 in the Appendix for an intuitive description of how the algorithms we use work, as we assume that the reader is familiar with algorithms such as DBScan, Affinity Propagation, and MeanShift.

Initially, we tried to cluster the matrix of embedding vectors X using standard approaches with no hyperparameter optimization. However, all of the algorithms we applied would fail with the default hyperparameters, indicating that no clustering can be found. We suspect that this is the case as the vectors produced by BERT seem to be smoothly distributed across the span of X, as can be seen in the PCA visualizations. As such, we had to adapt our experiment setup and implement an efficient hyperparameter selection tool, making use of random search and [16] and Bayesian optimization [139]. We use the Pytorch Adaptive Experimentation platform [1] 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 algorithms 3 and **??**. Algorithm 3 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 including the Chinese Whispers algorithm (3.1). We optionally reduce the dimensionality of the input embeddings through either PCA, or UMAP. Finally, we provide the target word w for which we want to calculate this overlap score. Similar to [98] our goal is to cluster contextual word embeddings. However, we have to modify the Chinese Whispers algorithm as [98] applies this for static word embeddings and we are utilising BERT contextual word embedding. Our modified version of the Chinese whispers is described in Algorithm 2.

```
Algorithm 2: Checks sampled BERT vectors for clusters by meaning
Input: $X$: The set of contextual word 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 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 [48]. Compared to static word embeddings, contextual word embeddings suffer from the *hubness property* [34] for a variety of tasks. The hubness property describes that there are certain embeddings (called hubs), which in the high-dimensional space are connected to a majority of other embedding vectors. This leads to the Chinese Whispers algorithm to collapse onto a single cluster assignment for the entire dataset. To improve addresses the hubness problem mentioned in [34] we apply the cutoff formula.

$$\texttt{cutoff}(x) = \mu(X) + c\sigma(X) \tag{4.1}$$

where c is an adjustable hyperparameter. Hubs are removed if the connectivity is too high.

```
Algorithm 3: Checks sampled BERT vectors for clusters by meaning
Input: $w_{\text{target}}$: The target word whose vectors we want to cluster;
       $DimRed$: The dimensionality reduction method;
       $k$ : Dimensionality to reduce to;
       $ClusterAlgorithm$: 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 $w_{\text{target}}$, 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.;
$\mathbf{X} \leftarrow BERT(\mathbf{D})$ i.e. pass each sentence through BERT and retrieve the resulting contextual 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 get a reliable estimate of which clustering approach generalizes best across words. This is depicted in the algorithm 4.

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

```
Input: $w_{\text{target}}$: 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 has a limited number of labeled datasampels, and because 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 choose 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 the words `was`, `thought`, `made`, `only`, `central`, `pizza`. We apply 5-fold cross-validation to ensure a good performance of the algorithms. 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 adjusted random index *ARI* [107], [59] which compares to what extent two clusterings are identical. This is necessary because clusters 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.*

$$ARI = \frac{\sum_{ij} \left({}^{n_{ij}}\right) - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{a_i}{_2} + \sum_j \binom{b_j}{2}\right] - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n}{2}} \tag{4.2}$$

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 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, we will be going over the most meaningful results. This is the case when we set $k = 20$.
Initially it was apparent that increasing k improves the clustering performance. However, introducing the Chinese Whispers algorithm would show that trimming down the dimensionality to $k = 20$ achieved the best results.

Please refer to section **??** to see the results for $k = [50, 100]$ for clustering methods excluding the adapted Chinese Whispers algorithm. Tables **??** and **??** show results where we keep the dimensionality at $k = 100$ and keep the sample size at $n = 500$ and $n = 10000$.

Table 4.4: The maximum ARI score achieved during hyperparameter optimization for the derivative 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 4.5: 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 size 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 to $n = 1000$. Results are shown in Figures **??**.

Table 4.6: 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. Table **??** includes the `[SEP]` at the end of the sentence.

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	0.316
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. We also observe that adding one 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, it is also important to do qualitative analysis and go over examples to see what this clustering scheme results in. The clustering shown below forms a partition over all sampled sentences. Figures 4.7 - 4.7 show a partition created when we run the above received best clustering algorithm on the polysemous word `arms`. WordNet records 10 different semantic classes for this word. For brevity, we will only show this as an explicit example. We refer to appendix C.4 for more examples.

```
                                                          - -
               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`. The context is about handcuffed arms. The sentiment is usually one of negative surprise.

```
                                  - -
      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`. The context is about a persons arms, where one person hugs or loves another (positive sentiment)

```
                             - -
         and shuttle robotic arms of a solar array and truss
                             - -
       a contingent of young arms that will allow us to win now
         By and large, those arms remained as fictional as those in "The War ..."
and extensive use of robotic arms operating at their limits
       staff of strong young arms that might have tamed the National League East
                             - -
```

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

```
                                                   - -
this country is so polarized that people spring to arms against any proposal
                                                   - -
                        At least they are carrying arms to protect themselves
                 His organization issued a call to arms
      he shoves it in the faces of his comrades in arms
                           people who had taken up arms against the United States
                    Mostly non-Arab rebels took up arms in early 2003
                                                   - -
```

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`.

```
                                          - -
       leaned back in his chair and, with arms crossed,
                                          - -
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 4.5: Partition 5 for the word `arms`. This cluster again refers to the arms of a person, however usually with a positive / optimistic sentiment.

```
                                                - -
                       The classic years of the arms race, the 1950s and ’60s before
                                                - -
                     that concerns over nuclear arms proliferation in the Middle East
                   Russian adherence to another arms control treaty
        he will press for peace and an eventual arms cut for the states
        payments to the companies that supplied arms to Iraq were often delayed
Mr.  Safar denies any wrongdoing, including any arms dealings
                                                - -
```

Figure 4.6: Partition 6 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.

As one can see, the different partitions do not quite capture semantics. Although partitions exist that exclusively capture one semantic class (partition 4 `weaponry`), most partitions carry a similar semantic class (partition 1-3 which capture `body parts`), the differences in partition is either sentiment where positive feeling include the concept of ”hugging”, and negative feelings include ”handcuffed” or ”forceful stretching”. Primarily, the different partitions capture context - which should be no surprise when we are sampling contextual word embeddings. On a second look, however, a strongly expressed linguistic feature seems to be sentiment. This is also confirmed by other samples (see Appendix C.4) , which more often than not include strong notions of sentiment features. We cannot manually capture all possible words, as there is no formal definition for linguistic features within BERT vectors to our knowledge and these concepts are purely man-made. However, we can conclude that for certain examples, 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 seen that when we partition the sampled contextual word embeddings into clusters, different clusters do not only correlate to semantics but also other linguistic features. The underlying reason for this cannot be trivially concluded, as it could also be a consequence of the nature of language. Correlation between different linguistic features are not often paid attention to and can lead to premature conclusions. To explain why some visualizations and conclusions are drawn the way they were in section 3, we analyse 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 accuracy rates of up to 92.6% accuracy on common part-of-speech benchmarks. Our hypothesis is that part-of-speech has a strong entailment towards semantics. One example of this manifestation is by *nominalised verbs.* These are nouns that achieved their meanings through their respective verb form. The word `the run`, for example became a noun through its verb form `to run`, and thus is considered a nominalised verb. This section starts with a more quantitative analysis of whether this hypothesis is true. We finish this experiment by conducting a quantitative test.

4.3.1 Experiment setup

We test the hypothesis ”semantics implies part-of-speech” by conducting the following experiment. We define a set of words, including nominalised verbs, which can appear as at least two different part-of-speech tags. These words are listed in the Appendix under section (C.3). 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. The methodology is specified in algorithm 5.

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 $\mathbf{w}_{\text{analyse}}$:
    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 part-of-speech tag, then the score results in a value of 1.0. If the dominant part-of-speech tag occurs only half the time, this number decreases to 0.5. If there are three part-of-speech tags, and all occur equally often, this percentage would drop down to 33%. Most words only have two different type-of-speech classes or are strongly dominant when more classes are present. Please notice that in this experiment again, we only view simple part-of-speech 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 tensorboard projector, using both UMAP and PCA as the dimensionality reduction method. We color 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. Figure 4.7 shows the cumulative plot of dominance amongst all sampled words that have been investigated.

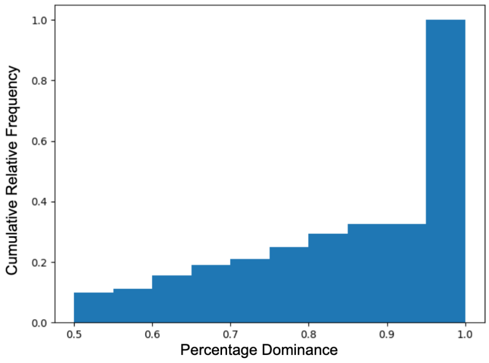
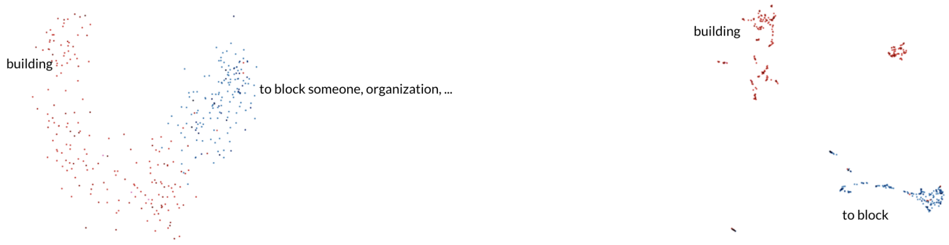


Figure 4.7: Cumulative dominance of the most occurring cluster. Dominance of a part of speech tag is measured by the percentage cover that the majority class intakes.



(a) Dimensionality reduction using PCA

(b) Dimensionality reduction using UMAP

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

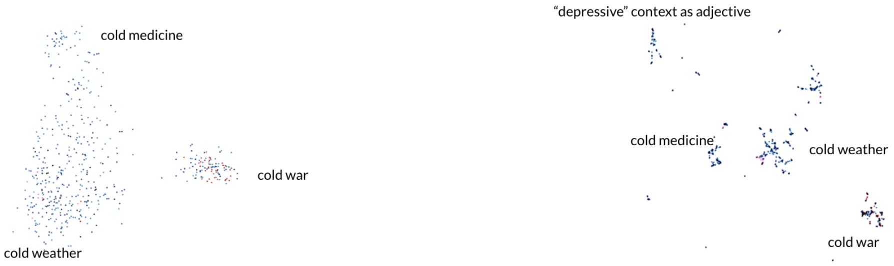
It is important to notice that some words did not have enough samples in the SemCor dataset for a significant valuation. We intentionally left out classes that had less than 5 samples. Also, SemCor is biased, as was previously stated. However, the results show that quite a majority of the sampled word-classes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

Qualitative Evaluation

To do a qualitative evaluation of 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` to keep the analysis concise.

Figures 4.8 - 4.9 show the tensorboard visualizations. We refer the reader to section C.2.1 in the Appendix for more examples. Please keep in mind that different colors imply different part of speech tags. Because the specific value does not play an important role, we leave out this legend.

The following trends are apparent: PCA shows a very smooth transition from different semantic classes, whereas UMAP strongly shows that clusters are forming between different semantic classes. This is also true for part-



(a) Dimensionality reduction using PCA

(b) Dimensionality reduction using UMAP

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

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 contextual word embeddings by semantics. This is apparent in figures C.12 and 4.9, where we can see that different blobs (UMAP) and extremes (PCA) in the figures respond to different semantic classes. Especially in Figure C.12 (a), we can see that in the case of UMAP, the central cluster strongly mixes nouns with adjectives (red and blue cluster) and that in the case of PCA, the contextual word 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 represented within the BERT language model. Now we want to understand how different modifications to the BERT model affect the performance on language modelling tasks. We proceed with GLUE, as it is a very versatile language modelling benchmark. 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 derivative models. As such, the focus lies on an 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 as **BERNIE**. Please refer to section 2 for a refresher on how sentences are tokenized inside BERT, and how tokens are converted to the latent representation and finally into contextual word 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, defined 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. Some split-words are present in Table 5 which sorts words by the number of WordNet meanings recorded, and the mean variance over 500 sampled contextual word embeddings. Introducing more specialized embeddings for tokens with high variance in contextual word embeddings will allow us to model more complex distributions. This happens because we introduce additional weights that can parametrize a specific token distribution, allowing a better approximation of the underlying true distribution. 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.

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

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 the number of WordNet senses and the variance amongst sampled BERT vectors.

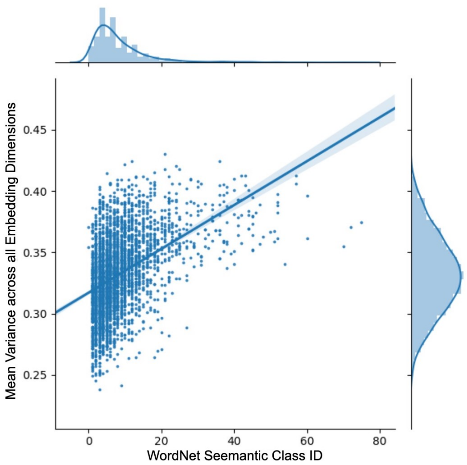


Figure 5.1: For each word w, we sample both the number of WordNet semantic classes that are recorded in the WordNet dataset. We also calculate the dimension-wise mean variance between $n = 500$ sampled vectors for the word w. This constitutes a point on this plot. We repeat this procedure for the most 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 contextual word embeddings. The right and top distributions show histograms of how occurrent the variance and WordNet classes are respectively. Our assumption is that introducing additional embedding-vectors inside BERT for certain words allows to capture mode complex distributions, i.e. a more complex distribution for words that have a higher number of WordNet classes.

5.1 BERNIE: Equalizing variances across BERT embeddings

We will introduce two adaptions to the BERT model, BERNIE PoS and BERNIE Cluster. We will present the results after having introduced both adaptations. For both modifications, we will use the GLUE dataset to understand what property of the language model is most affected. In the case of BERNIE Cluster, we use the performance of the downstream task as a proxy to understand what information is captured by the different modes in the data. By introducing additional parameters for words with variance (Figure 5.1), we can see what these embeddings capture the most. This can be checked by seeing what downstream tasks gain the most performance through the additionally added parameters.

5.1.1 BERNIE PoS

We want to start with a simple model first. We have seen in the above section that there is a strong correlation between semantics and part-of-speech. As such, the initial idea is to introduce additional embedding vectors that are able to capture semantics better. Because of the strong correlation between the two factors, part-of-speech is used as a proxy for semantics. One of the advantages that this carries is that we can use the spaCy part-of-speech tagger [2] to classify the part-of-speech tags with high accuracy. This model adoption is called **BERNIE PoS**.

Experiment setup

BERNIE PoS introduces one new embedding vector for each possible part-of-speech 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 initial *run*-embedding. We will refer to `run_VERB` and `run_NOUN` as so called *sub-embeddings*, as these exploit the subspace structure of the contextual word embeddings sampled for the word `run`. As for the tokenizer, we also introduce a mechanism that turns any occurrence of `run` into one of the sub-embeddings. Specifically, our part-of-speech modified pipeline is presented in Figure 5.2.
To identify whether the occurring `run` in an example sentence is a verb or a noun, we use the spaCy part-of-speech tagger [2], which claims 92.6 % accuracy for this task. When initializing the embeddings for the newly introduced embeddings, we copy the weights of the original embedding. This is depicted in Figure 5.3. The final pipeline is presented in 5.4. During training, all weights of BERT are fine-tuned to the downstream GLUE task as described in section 2.

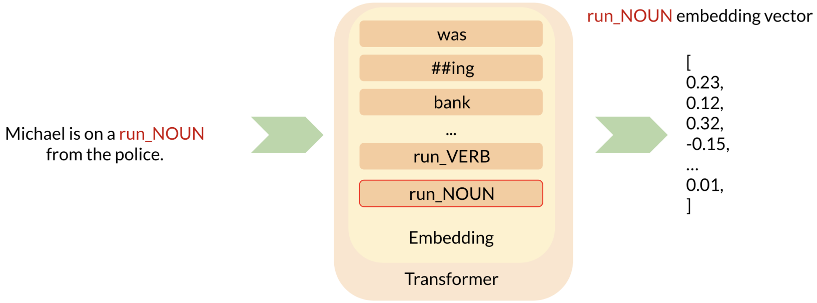


Figure 5.2: The part-of-speech modified pipeline. The BERNIE PoS model takes as input a sentence s. The sentence s is converted to a sequence of BERT tokens $[t_1, \ldots, t_m]$ as defined in a given vocabulary V. This vocabulary V is extended with the additional tokens for each of the split-words. For 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.

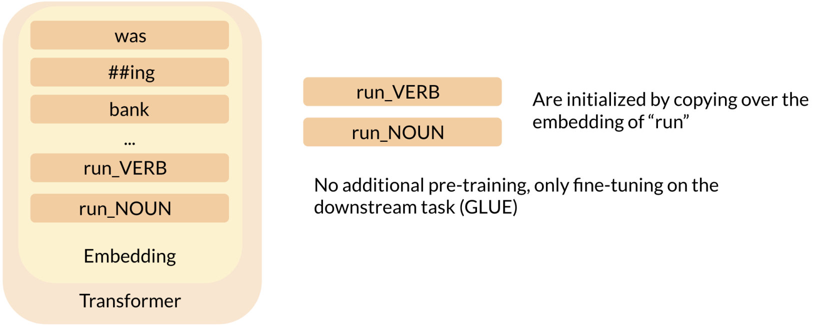
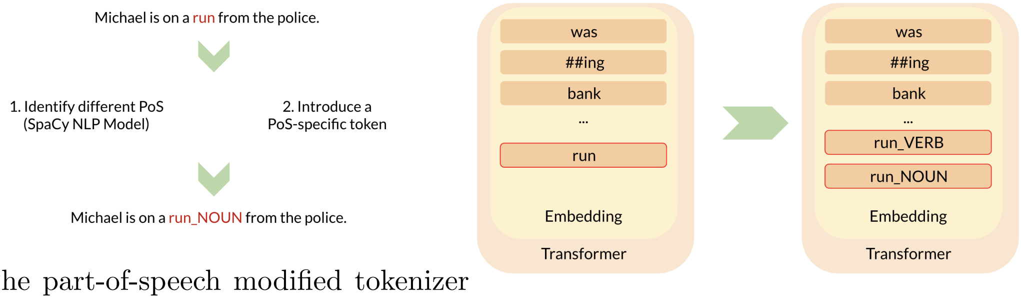

was
##ing
bank
...
run_VERB
run_NOUN
Embedding
Transformer
run_VERB
run_NOUN
Are initialized by copying over the embedding of "run"
No additional pre-training, only fine-tuning on the downstream task (GLUE)

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.

.


Michael is on a run from the police.
1. Identify different PoS (SpaCy NLP Model)
2. Introduce a PoS-specific token
Michael is on a run_NOUN from the police.
was
##ing
bank
...
run
Embedding
Transformer
was
##ing
bank
...
run_VERB
run_NOUN
Embedding
Transformer

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

(b) The PoS modified BERT embedding.

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

5.1.2 BERNIE Cluster

BERNIE Cluster introduces new tokens and their respective embeddings in the transformer embedding layer, similar to the previous 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. Although the hyperparameter optimization method optimized for the best overlap with coarseness of semantic classes as defined by WordNet, it is not quite interpretable to what these different clusters correspond to (as discussed in section 4.2). This is partially also because we cannot formally capture the concept of semantics or part-of-speech. In section 4.2) we had seen that these clusters do not correspond to WordNet semantic classes in general, however, we could also apply a per-case analysis of this. The aim of this experiment-model is to find out what these clusters correspond to the most, and we will use the GLUE benchmarks to understand what tasks this clustering-information benefits or detriments the most. This should give us a more generalizable idea of how the added parameters - dependant on the clustering assignments - affect downstream tasks. For clarity, we will make use of intuitive concepts like `bank_ FINANCE` and `bank_ SEA` for convenience, and because this is our core assumption that we aim to disprove. We will mention whenever this assumption does not seem to hold.

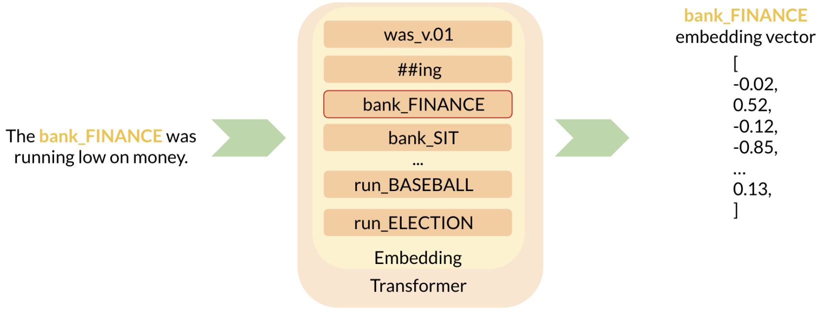
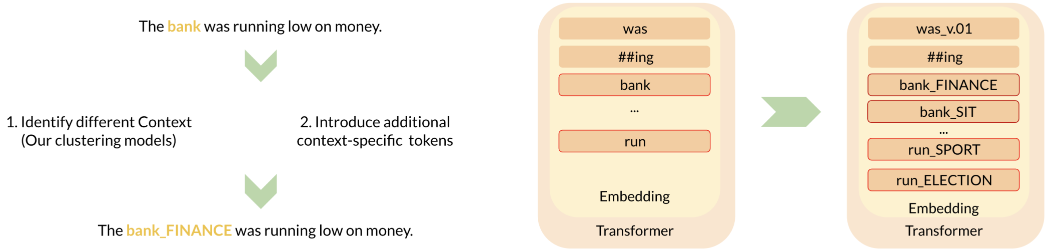


Figure 5.5: The semantic modified pipeline. The BERNIE Cluster model takes as input a sentence s. The sentence s is converted to a sequence of BERT tokens $[t_1, \ldots, t_m]$ as defined in a given vocabulary V. This vocabulary V is extended with the additionally introduced tokens for each of the split-words. 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.

Experiment setup

Again, our base model is the standard BERT model from section 2.2.5 . We now also modify the tokenizer in such a way, that split-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. We use the clustering approach introduced in Section 4.2 as the intermediate model to distinguish on which semantic class the sentence refers to. We also introduce new embedding vectors that the new tokens correspond to. However, please keep in mind that these clusters generally do not seem to refer to semantic classes (section 4.2), but we will use this as our assumption until we find data to contradict this. After we have concluded these changes, we arrive at the following model, which has a modified tokenizer, and a modified BERT model. We call the corresponding model **BERNIE Cluster**.


The bank was running low on money.
1. Identify different Context (Our clustering models)
2. Introduce additional context-specific tokens
The bank_FINANCE was running low on money.
was
##ing
bank
...
run
Embedding
Transformer
was_v.01
##ing
bank_FINANCE
bank_SIT
...
run_SPORT
run_ELECTION
Embedding
Transformer

(a) The clustering-modified tokenizer pipeline

(b) The clustering-modified BERT embedding

Figure 5.6: The resulting, fully modified BERNIE Cluster pipeline.

Results

We now evaluate the introduced BERNIE PoS and BERNIE Cluster models. Weights which are instantiated specifically for the GLUE tasks (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 GLUE tasks, the BERNIE model modifications have a negative effect. We assume that this is either because the newly initialized embeddings were not pre-trained like the other embeddings (on a 3.3 billion word-sized corpus), and also because the part-of-speech (accuracy of 92.6%) and clustering (accuracy less than 100%) models propagate an error term to the embedding-lookup task. On the other side, the WNLI and RTE experiments are considerably deviating in performance through the addition of the sub-embeddings. Both are often considered the most difficult GLUE tasks. For the BERNIE Cluster model, WNLI performs better (**10.56%**) than standard BERT, and RTE performs worse (**-7.40%**) 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.

WNLI and RTE are both natural language induction tasks 2. RTE focuses on a textual entailment task. It does not seem far off that performance here drops further, as most of the datasamples do not test for word disambiguation, but rather for the ability to capture the ”global” idea of the sentence.

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.

Thus, we assume that the newly added embeddings are not mature enough to distinctively capture the different concepts, but that too much noise was added (through the embedding-lookup mechanism, and the newly added embeddings), that performance deteriorates. In contrast, the WNLI task improves in task performance where the focus is on target words. Specifically, WNLI does not only look at the global idea captured by the sentence (as is done with RTE) but looking at individual words to understand what is the pronoun for a referent. It seems apparent that making the word-sense more explicit by introducing specialized tokens and embeddings can help cut ties with this task. Because WNLI has low performance, it could also be that the added weights capture other latent information within the embeddings, showing that the embedding projection is a bottleneck of the BERT model for tasks similar to WNLI. Also, compared to BERNIE PoS, the BERNIE Cluster model amplifies the performance-difference to BERT in both the negative and positive direction. We assume that this is because BERNIE Cluster implements a clustering module, which is able to partition the BERT space much more smoothly than the part-of-speech tagger does.

5.1.3 BERNIE Cluster with additional pre-training

One of the trends we see is the overall negative trend in performance. This might be due to the newly initialized vectors not being calibrated enough, i.e. not pre-trained on a large-enough corpus (as is done with the rest of the embedding vectors [37]), but only fine-tuned for GLUE tasks on limited data. Thus we measure the performance of BERNIE models after additional pre-training.

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 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 language model methodology as described in section 2. We save this model and load it for the GLUE tasks. BERNIE full Pre-training trains all the embeddings, whereas BERNIE partial Pre-Training fixates all embedding vectors except the ones that were newly added. **BERNIE full Pre** allows the gradients to update all the embedding weights during pre-training. **BERNIE partial Pre** only allows gradient updates for the embeddings of the newly added vocabulary items, i.e. the split-words. Specifically, only the newly added embeddings vectors are calibrated, while all other embedding vectors are kept constant.

5.1.5 Results

Again, we run 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 performance on most of the GLUE tasks. 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 hyperparameters not being optimal for this run. Finally, it could also be that the initialization for the newly added vectors was suboptimal, as we have seen strong deviations (of up to 5% in performance) over different random seeds for the GLUE tasks.

Chapter 6

Conclusion

We focus our analysis on how contextual word embeddings produced by BERT capture *semantics*. We analyse the outputs of the BERT language model, as it provides a good balance between popularity, close to state-of-the-art performance, and generalizability to other modern language models.

6.1 Main Contributions

Our main observations are the following:

1. We test for linear separability and ability to partition the space into semantic clusters within the sampled BERT vectors. We show that while linear separability is possible, it is nearly impossible to find modes in the data which correspond to semantic classes that are as distinctive as defined by WordNet. Although semantics is captured to a large extent, we conclude that BERT organized the embedding vectors imminently by context. This can easily introduce undesirable properties such as strong sentiment as observed in one of our experiments, leading to considerable bias in downstream tasks.
2. We analyze the relationship between the two linguistic features of part-of-speech and semantics. We show that these show a strong correlation in languages and that modern language models capture this strong

correlation. This affects the conclusion of some related work which had analysed language models which capture semantics, but which did not account for the correlation between part-of-speech and semantics and shows that more attention should be paid when drawing conclusion on semantics, whereas the underlying deciding principle would have been part-of-speech.

3. Finally, we introduce additional parameters in the form of new word embeddings for words whose sampled embedding vectors have high variance over the embedding space. We investigate what downstream tasks are most affected by these changes and use this as a proxy to understand what the clustering assignments correspond to. Most GLUE tasks have slightly negative performance in effect. The newly introduced embeddings strongly decrease RTE task performance, while the WNLI task performance is strongly improved

Our findings show that BERT does not capture an easily interpretable semantic subspace, it rather imminently organizes the embeddings purely by context. Although semantics is often captured, this can introduce undesirable features such as strong sentiment, position in sentence, and thus lead to considerable bias in downstream applications.

6.2 Future Work

Because we were only able to investigate some of our core hypotheses, different directions for future work are imminent. These could include or investigate:

Introducing the WiC benchmark: One of our main targets with the modifications made by BERNIE Cluster was to analyse what effect this modification has on semantics, specifically polysemy. An ablation study on other benchmarking datasets, most specifically the Word in Context benchmark which is included in SuperGLUE would have been a valuable addition to understand what our modifications have to the modelling power of language models.

Larger benchmarking dataset: Within our analysis, we have had numerous occasions where the statistical significance and testing power largely suffered due to the limited size of gold-standard datasets that capture semantics. SemCor is one of the most extensive datasets available in this domain but is still too small to do a proper evaluation on. This limits the choice of polysemous words analysed in Section **(4)**. A bigger labelled dataset would lead to a more expressive analysis.

Semantic subspace not apparent: An interesting line of future work could lie in analysing whether a semantic is only apparently locally (per word w), or whether a global principal component capturing semantics exists. Latter could be done using distance metric learning (see Appendix B.3).

Formalizing semantics: Intuitively, we have often seen that certain words such as `was` were over-defined in terms of semantic classes, where many dozen semantic classes are apparent. Generally, it is difficult to quantify "coarseness" between semantic concepts, apparent in section **(4)**. A more quantified definition would be beneficial in defining semantics.

Limited coherence in lines of work Different authors seem to analyse this topic using different tools, leading to partially incoherent statements. Also, the nature that linguistic features are strongly correlated (i.e. part-of-speech and semantics), is not properly noted by a lot of previous work, making conclusions on semantics when part-of-speech was analysed. Establishing a more formal framework could address this, and the previously mentioned issue.

Bibliography

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

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

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

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

[5] Wiktionary. `https://en.wiktionary.org/wiki/Wiktionary:Main_Page`. Accessed: 2020-05-11.

[6] Margareta Ackerman and Shai Ben-David. Clusterability: A theoretical study. *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics, PMLR 5:1-8*, 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. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 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:1607.06450*, 2016.

[11] Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *ICLR 2015*, 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 Jauvin. 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 *TACL*, 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. *Workshop on Graph-based Methods for Natural Language Processing*, pages 73–80, 2006.

[18] Chris Biemann and Martin Riedl. Text: Now in 2d! a framework for lexical expansion with contextual similarity. *Journal ofLanguage Modelling, 1(1)*, pages 55–95, 2013.

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

[20] Rishi Bommasani, Kelly Davis, and Cardie Claire. BERT wears GloVes: Distilling static embeddings from pretrained contextual representations. -, 2019.

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

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

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

[24] Elia Bruni, Nam Khanh Tran, and Marco Baroni. Multimodal distributional semantics. *Journal of Artificial Intelligence Research 49*, pages 1–47, 2013.

[25] 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.

[26] 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. *Journal of Artificial Intelligence Research*, 2018.

[27] 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.

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

[29] 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:1708.00055*, 2017.

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

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

[32] Andy Coenen, Emily Reif, Ann Yuan, Been Kim, Adam Pearce, Fernanda Viégas, and Martin Wattenberg. Visualizing and measuring the geometry of BERT. *Advances in Neural Information Processing Systems 32 (NIPS 2019)*, 2019.

[33] 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.

[34] Alexis Conneau, Guillaume Lample, Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. Word translation without parallel data. *arXiv:1710.04087v3*, 2017.

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

[36] Michael Denkowski. A survey of techniques for unsupervised word sense induction. -, 2009.

[37] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv:1810.04805*, 2018.

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

[39] Matrin Ester, Hans-Peter Kriegel, Joerg Sander, and Xiaowei Xu. A

density-based algorithm for discovering clusters. *KDD-96 Proceedings*, 1996.

[40] Kawin Ethayarajh. How contextual are contextualized word representations? comparing the geometry of BERT, ELMo, and GPT-2 embeddings. *EMNLP 2019*, 2019.

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

[42] Brendan J. Frey and Delbert Dueck. Clustering by passing messages between data points. *Science*, 2007.

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

[44] Octavian-Eugen Ganea, Gary Becigneul, and Thomas Hofmann. Hyperbolic entailment cones for learning hierarchical embeddings. *International Conference on Machine Learning (ICML) 2018*, 2018.

[45] Weifeng Ge, Weilin Huang, Dengke Dong, and Matthew R. Scott. Deep metric learning with hierarchical triplet loss. *European Conference on Computer Vision (ECCV)*, 2018.

[46] 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.

[47] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. *CVPR*, 2006.

[48] Aric Hagberg, Dan Schult, and Pieter Swart. Networkx. `https://github.com/networkx/networkx`, 2006. Accessed: 2020-05-07.

[49] 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 6:1, pages 6–4. Venice, 2006.

[50] Zellig Harris. Distributional structure. *Word*, 10, 1954.

[51] W. F. Hegel. Encyclopedia of the philosophical science: Science of logic. -, 1817.

[52] John Hewitt and Christopher D Manning. A structural probe for finding syntax in word representations. *Association for Computational Linguistics*, 2019.

[53] Felix Hill, Roi Reichart, and Anna Korhonen. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 2015.

[54] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *arXiv:1909.09586*, 1997.

[55] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. *arXiv:1412.6622*, 2014.

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

[57] Matthew Honnibal and Ines Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. in progress, 2017.

[58] 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.

[59] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, pages 193–218, 1985.

[60] Shankar Iyer, Nikhil Dandekar, and Kornel Csernai. First quora dataset release: Question pairs, 2017.

[61] 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.

[62] 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.

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

[64] Vidur Joshi, Matthew Peters, and Mark Hopkins. Extending a parser to distant domains using a few dozen partially annotated examples. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1190–1199, 2018.

[65] Rafal Jozefowicz, Orriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. *arXiv:1602.02410*, 2016.

[66] Mikael Kageback and Hans Salomonsson. Word sense disambiguation using a bidirectional lstm. *Proceedings of the 5th Workshop on Cognitive Aspects of the Lexicon (CogALex - V)*, pages 51–56, 2016.

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

[68] Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. Revealing the dark secrets of BERT. *EMNLP 2019*, 2019.

[69] Sneha Kudugunta, Ankur Bapna, Isaac Caswell, Naveen Arivazhagan, and Orhan Firat. Investigating multilingual nmt representations at scale. *EMNLP*, 2019.

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

[71] 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.

[72] Jason Lee, Kyunghyun Cho, and Thomas Hofmann. Fully character-level neural machine translation without explicit segmentation. *Transactions of the Association for Computational Linguistics (TACL)*, 2017.

[73] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. BioBERT: pre-trained biomedical language representation model for biomedical text mining. *arXiv:1901.08746*, 2019.

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

[75] Yoav Levine, Barak Lenz, Or Dagan, Dan Padnos, Or Sharir, Shai Shalev-Shwartz, Amnon Shashua, and Yoav Shoham. SenseBERT: Driving some sense into BERT. *ACL 2020*, 2019.

[76] 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.

[77] 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. *arXiv:1907.11692*, 2019.

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

[79] Shuang Ma, Daniel Mcduff, and Yale Song. M3 d-gan: Multi-modal multi-domain translation with universal attention. *arXiv:1907.04378*, 2019.

[80] 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.

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

[82] Mitchell P. Marcus, Mary Ann Marcinkiewic, and Beatrice Santorini. Building a large annotated corpus of english: the penn treebank. *Computational Linguistics*, 1993.

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

[84] Chandler May, Alex Wang, Shikha Bordia, Samuel R. Bowman, and Rachel Rudinger. On measuring social biases in sentence encoders. *arXiv:1903.10561*, 2019.

[85] Diana Mccarthy, Marianna Apidianaki, and Katrin Erk. Word sense clustering and clusterability. *Computational Linguistics, Volume 42, Issue 2 - June 2016*, pages 245–275, 2016.

[86] Timothee Mickus, Denis Paperno, and Kees Van Deemter. What do you mean, BERT? assessing BERT as a distributional semantics model. *Proceedings of the Society for Computation in Linguistics: Vol. 3 (2020), Article 34*, 2019.

[87] 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.

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

[89] 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.

[90] 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.

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

[92] George A. Miller, Martin Chodorow, Shari Landes, Claudia Leacock, and Robert G. Thomas. Using a semantic concordance for sense identification. *ARPA Human Language Technology Workshop*, pages 240–243, 1994.

[93] Mehrad Moradshahi, Hamid Palangi, Monica S Lam, Paul Smolensky, and Jianfeng Gao. HUBERT untangles BERT to improve transfer across NLP tasks. *arXiv:1910.12647*, 2019.

[94] Panagiotis Moutafis, Mengjun Leng, and Ioannis A. Kakadiaris. An overview and empirical comparison of distance metric learning methods. *IEEE Transactions on Cybernetics*, 2017.

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

[96] Jiazhi Ni, Jie Liu, Chenxin Zhang, Dan Ye, and Zhirou Ma. Fine-grained patient similarity measuring using deep metric learning. *ACM on Conference on Information and Knowledge Management*, 2017.

[97] 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.

[98] 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.

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

[100] Matthew E Peters, Mark Neumann, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *Proceedings of NAACL-HLT 2018*, page 2227–2237, 2018.

[101] Matthew E Peters, Mark Neumann, Luke Zettlemoyer, and Wen-Tau Yih. Dissecting contextual word embeddings: Architecture and representation. *arxiv:1808.08949v2*, 2018.

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

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

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

[105] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *arxiv:1904.02679*, 2019.

[106] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv:1606.05250*, 2016.

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

[108] Nils Reimers, Benjamin Schiller, Tilman Beck, Johannes Daxenberger, Christian Stab, and Iryna Gurevych. Classification and clustering of arguments with contextualized word embeddings. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

[109] Steffen Remus and Chris Biemann. Retrofitting word representations for unsupervised sense aware word similarities. *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152, 2018.

[110] 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.

[111] Oren Rippel, Manohar Paluri, Piotr Dollar, and Lubomir Bourdev. Metric learning with adaptive density discrimination. *Proceedings of the International Conference on Learning Representations*, 2016.

[112] Peter J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Computational and Applied Mathematics 20*, pages 53–65, 1987.

[113] David Rumelhart, Geoffrey Hinton, and Ronald Williams. Learning internal representations by error propagation. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*, pages 318–362, 1987.

[114] 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.

[115] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, 2016.

[116] 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. *ACL 2019*, 2019.

[117] Peng Shi, Jimmy Lin, and David R Cheriton. Simple BERT models for relation extraction and semantic role labeling. *arXiv:1904.05255*, 2019.

[118] Jamin Shin, Andrea Madotto, and Pascale Fung. Interpreting word

embeddings with eigenvector analysis. *32nd Conference on Neural Information Processing Systems (NIPS 2018)*, 2018.

[119] Yuqi Si, Jingqi Wang, Hua Xu, and Kirk Roberts. Enhancing clinical concept extraction with contextual embeddings. *Journal of the American Medical Informatics Association*, 2019.

[120] 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.

[121] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. *NIPS*, 2016.

[122] Hyun Oh Song, Stefanie Jegelka, Vivek Rathod, and Kevin Murphy. Deep metric learning via facility location. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

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

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

[125] 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. *Association for the Advancement of Artificial Intelligence 2020*, 2019.

[126] Ilya Sutskever, James Martens, and Geoffrey Hinton. Generating text with recurrent neural networks. *Proceedings of the 28 th International Conference on Machine Learning,*, 2011.

[127] Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy Lin. Distilling task-specific knowledge from BERT into simple neural networks. *arXiv:1903.12136*, 2019.

[128] Henry Tsai, Jason Riesa, Melvin Johnson, Naveen Arivazhagan, Xin Li, and Amelia Archer. Small and practical BERT models for sequence labeling. *EMNLP-IJCNLP 2019*, 2019.

[129] 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.

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

[131] 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.

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

[133] Alex Wang and Kyunghyun Cho. BERT has a mouth, and it must speak: BERT as a markov random field language model. *arXiv:1902.04094*, 2019.

[134] 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.

[135] 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.

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

[137] 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. *arXiv:1908.04577*, 2019.

[138] Xun Wang, Xintong Han, Weilin Huang, Dengke Dong, and Matthew R. Scott. Multi-similarity loss with general pair weighting for deep metric learning. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[139] 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.

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

[141] 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. *Third Workshop on Evaluating Vector Space Representations for NLP (RepEval 2019)*, 2019.

[142] Gregor Wiedemann, Steffen Remus, Avi Chawla, and Chris Biemann. Does BERT make any sense? interpretable word sense disambiguation with contextualized embeddings. *Conference on Natural Language Processing (KONVENS) 2019*, 2019.

[143] 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.

[144] Ludwig Wittgenstein. Philosophical investigations. -, 1953.

[145] 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:1910.03771*, 2019.

[146] 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. *arXiv:1609.08144*, 2016.

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

Appendix A

Background

A.1 Linguistic features

Other linguistic features are recorded in [135] 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: Table taken from [135]. Types of linguistic phenomena organized under four major categories.

A.2 Gaussian Embeddings

[14] already used distributional representations to embed concepts from natural language processing into embeddings. [132] considers creating a Gaussian

representation for each word token, resulting in Gaussian static 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 that 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 static word embedding x_w is expressed by

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

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 $\mathcal{V}$). [132] 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:

$$\begin{aligned} E_\theta(w, c) &= \int_{x \in \mathcal{R}^d} f(x) g(x) dx && \text{(A.3)} \\ &= \int_{x \in \mathcal{R}^d} \mathcal{N}(x; \mu_w, \Sigma_w) \mathcal{N}(x; \mu_c, \Sigma_c) dx && \text{(A.4)} \\ &= \mathcal{N}(0; \mu_w - \mu_c, \Sigma_w + \Sigma_c) && \text{(A.5)} \end{aligned}$$

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.

$$
\begin{aligned}
-E(w_i, c_j) &= D_{KL}(c_j||w_i) && \text{(A.6)}\\
&= \int_{x\in\mathcal{R}^d} \mathcal{N}(x;\mu_{w_i},\Sigma_{w_i})\text{log}\frac{\mathcal{N}(x;\mu_{c_j},\Sigma_{c_j})}{\mathcal{N}(x;\mu_{w_i},\Sigma_{w_i})}dx && \text{(A.7)}\\
&= \frac{1}{2}\left(\text{tr}\left(\Sigma_i^{-1}\Sigma_j\right) + \left(\mu_i - \mu_j\right)^\top \Sigma_i^{-1}\left(\mu_i - \mu_j\right) - d - \log\frac{\det\left(\Sigma_j\right)}{\det\left(\Sigma_i\right)}\right) && \text{(A.8)}
\end{aligned}
$$

The asymmetric distance property allows us 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 closed-form formulation of

$$
\begin{aligned}
\mu_z &= \mu_x^T\mu_y && \text{(A.9)}\\
\Sigma_z &= \mu_x^T\Sigma_x\mu_x + \mu_y^T\Sigma_y\mu_y + \text{tr}\left(\Sigma_x\Sigma_y\right) && \text{(A.10)}
\end{aligned}
$$

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.

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

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

$$
\begin{aligned}
&\forall i \quad \|\mu_i\|_2 \leq C && \text{(A.12)}\\
&\forall i \quad mI < \Sigma_i < MI && \text{(A.13)}
\end{aligned}
$$

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 [54]. LSTMs are an extension of the recurrent neural network *RNN* [113], but introduces a mechanism to solve the problem of error in the backpropagating gradient, and a "storage" 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

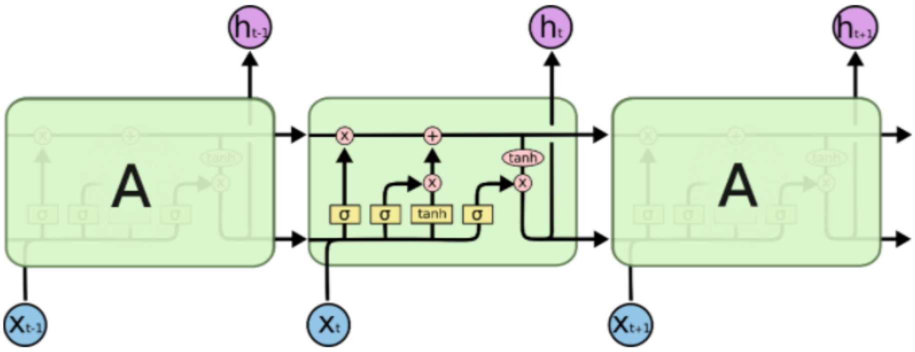


Figure A.1: Figure taken from [4]. The internals of the LSTM cell, and the recurrent flow which is repeated for three consecutive timesteps $t-1, t, t+1$. The LSTM produces an hidden representation at every steep h_{t-1}, h_t, h_{t+1} given some inputs x_{t-1}, x_t, x_{t+1}.

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 [54] are good at estimating the probability of word $w^{(t)}$ occurring given a previous history of words $w^{(t-1)}, \ldots w^{(1)}$. In practice, the LSTM is a popular choice for sequence modeling tasks, as it can very well capture the following probability distribution.

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

A.4 GLUE

A.4.1 Single Sentence Tasks

The Corpus of Linguistic Acceptability **CoLA** [140] 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.

Example 13

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

Example 14

```
label:0
The professor talked us.
```

The Stanford Sentiment Treebank **SST-2** [120] consists of sentences from movie reviews and human annotations of their sentiment. This is a binary classification (positive / negative) task.

Example 15

```
label:1
comes from the brave , uninhibited performances
```

Example 16

```
label:0
excruciatingly unfunny and pitifully unromantic
```

A.4.2 Similarity and paraphrase tasks

The Microsoft Research Paraphrase Corpus **MRPC** [38] 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.

Example 17

```
quality:1
Prosecutors filed a motion informing Lee they intend to seek
the death penalty.
He added that prosecutors will seek the death penalty.
```

Example 18

```
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 his shooting death .
```

The Quora Question Pairs **QQP** [60] dataset is a collection of question and answer pairs from the website Quora. The task is 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.

Example 19

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

Example 20

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

The Semantic Textual Similarity Benchmark **STS-B** [29] 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.

Example 21

```
The man is riding a horse.
A man is riding on a horse.
5.000
```

Example 22

```
A man is playing a guitar.
A girl is playing a guitar.
2.800
```

A.4.3 Inference tasks

The Multi-Genre Natural Language Inference Corpus **MNLI** [143] [21] is a crowd-sourced collection of annotated sentence pairs for a textual entailment task. For each premise and hypothesis sentence, the task is to predict whether the premise entails the hypothesis, contradicts the hypothesis, or does 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 mismatched MLNI tests on a training set which is cross-domain. The MNLI data samples consist of parse-trees, which is the reason we do not display these here. We will be working with the matched MNLI task only for convenience reasons.

The Stanford Question Answering Dataset is a question-answering dataset [106]. The authors of GLUE augment this dataset and create the Question answering Natural Language Induction **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.

Example 23

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

Example 24

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

A.5 WordNet

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 A.2: Example output for WordNet 3.1 noun propositions for the word `was`. In total, 14 different concepts are recorded.

Appendix B

Other lines of work

B.1 Clustering

Affinity Propagation was first introduced by [42]. Affinity propagation takes as input measures of similarity between pairs of data points. Real-values messages for multiple iterations are exchanged between data points 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 [39] which initiates by finding representative samples that are in regions of high density, and expands clusters from thereon. 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 [27]. **Optics** [87] is another extension of DBScan which keeps the cluster hierarchy for a variable neighborhood radius.

MeanShift is a centroid-based clustering algorithm [33]. 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 Static word embeddings and contextual word embeddings

The general idea behind distilling static word embeddings from contextual word embeddings is that one can arrive at a (1) compressed representation of the contextual word embedding, and (2) capture polysemous concepts through multiple static word embeddings rather than one embedding per token. We present a few ways static word embeddings can be distilled from BERT.
This includes distilling knowledge or minimizing BERT to smaller neural network models [127], [128], and most famously [114]. Compressed sentence representations can also be learned i.e. through random projections or architectures similar to autoencoders [116], amongst others, making use of a max-pool operator over sentence-tokens to arrive at a sentence embedding.

[118] 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.
[141] follows 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.

B.3 Metric Learning and Disentanglement

Although different metric spaces can be considered, we will be referring 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:

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

As is also mentioned in [94], the proximity relation can also be captured through *relation triplets*

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

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

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

or equivalents as the l_2-norm

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

Most distance learning techniques propose different ways of learning a Mahalanobis distance [81]

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

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

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

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 [30].
Specifically, the data is projected into another The authors in [94] 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.
[124] provides a good introduction to 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 known as *pseudo-dinstances* 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 the 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.

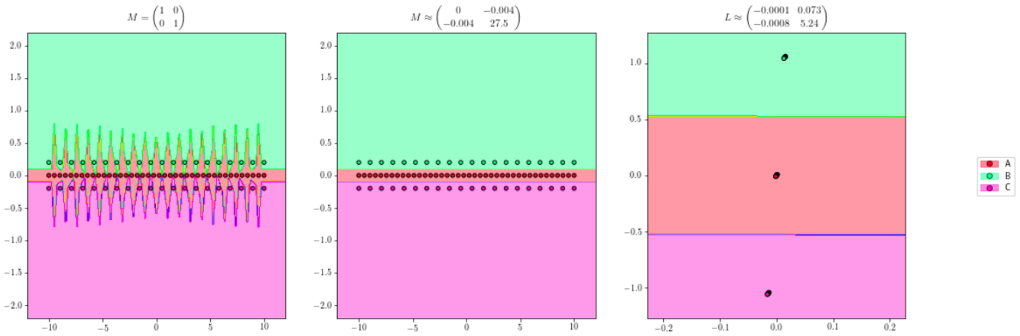


Figure B.1: Taken from [124]. The individual tiles show the kNN prediction regions by color for every point in the image. Using the unmodified euclidean distance, this would result in classification regions on the left. The reader can see, learning an appropriate distance, the classification is much more effective (middle). Finally, the dimensionality is also reducible with this dimension while still matching the classification accuracy (right).

[124] offers a survey on this work.

Deep Metric Learning

Besides the non-deep proposed in the work above models, [67] also introduces the deep generalization to these frameworks.

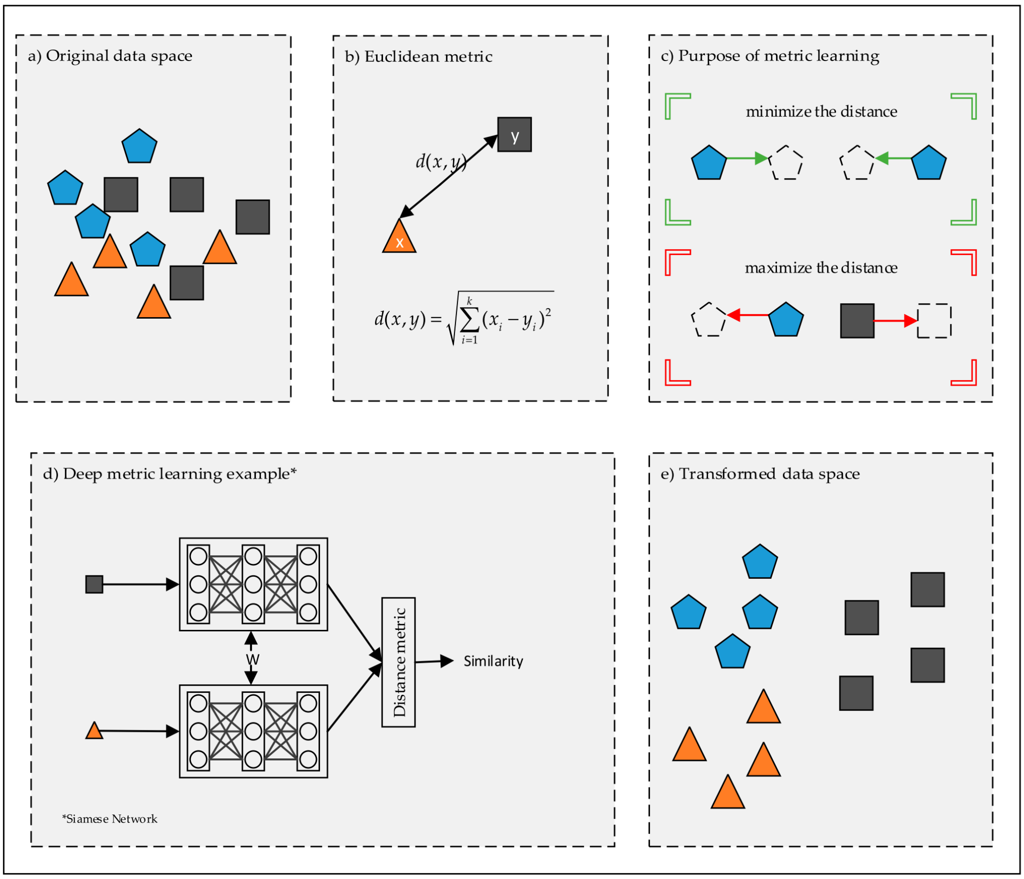


Figure B.2: Taken from [67]. An illustration of deep metric learning. The space is transformed in such a way, that similar object are closer to each other, and dissimilar objects are moved away from each other.

The most prominent model is the siamese network [23], [31], [47], where the network consists of two identical sub-networks joined at their outputs. We provide an example the architecture looks that looks as follows:

Example 25

```
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 [55] 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 that using a triplet logic results in more stable training and bigger margins between similarity classes. Analogous logic goes for quadruple loss etc., but it is unsure to what extent 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 [130], the structured loss [123] n-pair loss [121], magnet loss [111], angular loss [136], quadruple loss [96], clustering loss [122], hierarchical triplet loss [45] and the multi-similarity loss [138].

B.3.1 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 [69] 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 Transformer-Big architecture proposed in the [131] 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.

[70] 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 unsupervised translation loss. Other work tries to disentangle the different linguistic, visual, and audio-based features by supplying multi-modal input at once [79]. 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).

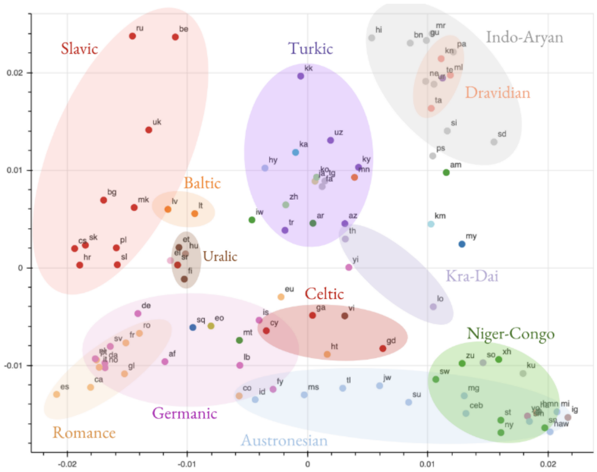


Figure B.3: From [69], visualizing clustering of the encoder representations of all languages, based on ther SVCCA similarity.

Appendix C

More Results

C.1 Linear Separability

Table C.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

C.2 Finding the best clustering model

Quantiative Evaluation

Table C.2: The maximum ARI score achieved during hyperparameter optimization for the different models as described by experiment for $k = 50$ and $n = 500$.

Table C.3: 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	0.001
DBScan	0.000
HDBScan	**0.328**
MeanShift	0.004
Optics	0.000

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

Table C.4: 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

We present an example of a partitioning over the word `bank`.

```
                                                  - -
               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: Partition 1 for the word `bank`. This cluster contains `bank` in the context of national banks as financial institutions. Notice that the sentiment is usually a negative one.

```
                                               - -
                            I would expect the bank to the left of the road
                                              - -
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: Partition 2 for the word `bank`. This cluster contains `bank` as a sequence of objects.

```
                                                  - -
              of their family’s naturalization -  bank deposit by bank deposit
                                                  - -
    12 that the company might suffer a run on the bank because of mortgage concerns
                               Third, per several bank managers of major national banks
Local party bosses gained broad powers over state bank lending, taxes
               government contracts, and a web of bank accounts
                       Prince Bandar’s Washington bank accounts
                                                  - -
```

Figure C.3: Partition 3 for the word `bank`. This cluster contains `bank` as a local, consumer bank (in contrast to institutional, national banks).

```
                                      - -
  lift mystery surrounding the central bank and improve communications with Wall Street
                                      - -
many economists had predicted that the bank would not cut its rate
         expectations that the central bank will raise interest
               a hint that the central bank plans to hold rates
                       China’s central bank has stepped up its purchases of dollar
  fertilizer prices in Africa, but the bank itself had often failed to recognize
                                      - -
```

Figure C.4: Partition 4 for the word `bank`. This cluster contains `bank` in the context of national banks as financial institutions. Notice that here, the context is usually about interest rates.

```
                                            - -
        citing challenges for the investment bank and the potential for a credit burden
                                            - -
93 percent drop in profits at its investment bank last week
   UBS said it did not expect its investment bank to return to profitability
                 defrauded by the investment bank in 1998 when
    said in an interview that the investment bank approached him last month
  said the leaders of Citigroup’s investment bank and alternative
                                            - -
```

Figure C.5: Partition 5 for the word `bank`. This cluster contains `bank` in the context of investment banks (in contrast to consumer, and institutional banks).

We now present a full partitioning over the samples of word `key`.

```
                                       - -
                            Many of the key Arab states
                                       - -
          in two months and Australia’s key S&P ASX 200 shed 1.9 percent
  Wall Street rebounded Wednesday after key earnings reports from JPMorgan Chase & Co.
       The Democratic candidate hires a key strategist
quickly established a good rapport with key donors"
                   able to meet the two key officials in the government
                                       - -
```

Figure C.6: Partition 1 for the word `key`. This cluster contains `key` as a synonym for the word `main`, which characterizes an important subject. The context is one between large institutions in politics and finance. The sentiment is generally positive.

```
                                          - -
president of Trinity College, who played a key role in designing the test
                                          - -
                     seen in the West as a key for the fairness of an election
                           treat ...  as a key factor in its regaulatory decision
       which policy makers have called the key test to lower interest rates
                             9.  11.  as a key element in pitch meetings
                                          - -
```

Figure C.7: Partition 2 for the word `key`. This cluster contains `key` as a synonym for the word `main`, which characterizes an important subject. The context is one of regulations and elections. The sentiment is serious.

```
                                           - -
                          Interstate 5 is a key route connecting Southern California
                                           - -
                                          A key piece of 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
                                       is a key indicator of retailer performance
                            in the West the key players were not a small group reading
                                           - -
```

Figure C.8: Partition 3 for the word `key`. This cluster contains `key` as a synonym for the word `main`, which characterizes an important subject. The context is one of groups of people. The sentiment is neutral to positive.

```
                                                  - -
         times change and technology advances, the key to the city symbolizes
                                                  - -
And an official, five-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
                                                  - -
```

Figure C.9: Partition 4 for the word `key`. This cluster contains `key` as a synonym for the word `main`, which characterizes an important subject. The context is one of sports and baseball. The sentiment is mixed.

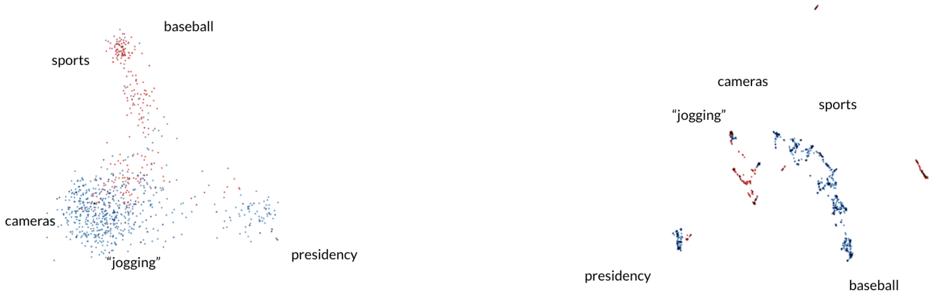
```
                                           - -
        The Red Sox will now have all their key players from the 2007 championship team
                                           - -
      Mike Green scored 12 points and had a key assist in overtime as Butler beat
           or taking the chance of losing a key player to injury
        But they never led, could not get a key basket at crucial times and played like
Cam Long stole the ball near the top of the key and ran out the clock
                                           - -
```

Figure C.10: Partition 5 for the word `key`. This cluster contains `key` as a synonym for the word `main`, which characterizes an important subject. The context is one about physical activity. The sentiment is mixed.

```
                                               - -
                                 Three of their key players played more than 40 minutes
                                               - -
                                              A key for the Giants on Sunday
chemical reactions on solid surfaces, which are key to understanding questions like
       but is she part of the conspiracy or the key to Sim’s salvation?
 to play on a sprained ankle against the Eagles key to that matchup
                  Horses have been the lifelong key to satisfying real feminine needs
                Connecticut cornerback said the key to defeating Louisville
                                               - -
```

Figure C.11: Partition 6 for the word `key`. This cluster contains `key` as a synonym for the word `main`, which characterizes an important subject. The context is mixed. The sentiment is mixed.

C.2.1 More qualitative evaluation



(a) Dimensionality reduction using PCA

(b) Dimensionality reduction using UMAP

Figure C.12: PCA and UMAP visualizations for contextual word embeddings X sampled for the word $w =$`run`.

C.3 Nominalised Verbs

In the BERNIE experiments, we define the following words as split-words:

`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

[129] 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 [34]. 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.

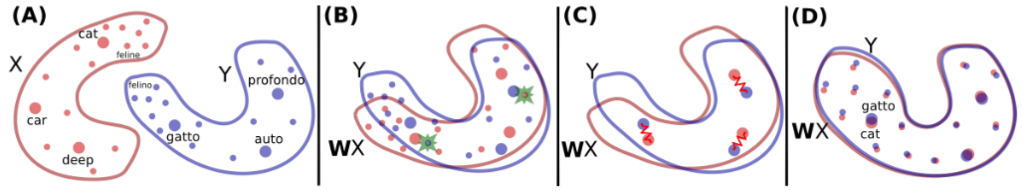


Figure D.1: Taken from [34]. Toy illustration of the embedding matching methodology using the MUSE model.

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

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

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-embeddings into a common global embedding space.

A generalization of these word vectors to point-vector probability-distributions is 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 that 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:

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

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

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

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 by finding a set of transportation plans in polytopes.

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

Finally, the following cost function is minimized to find an optimal OT plan

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