Exploring Multilingual and Contextual Properties in Word Representations from BERT

Pernille Aaby

Thesis submitted for the degree of

Master in Applied Computer and Information Technology (ACIT)

60 credits

Department of Computer Science

Faculty of Technology, Art and Design

OSLO METROPOLITAN UNIVERSITY

Spring 2022

Exploring Multilingual and Contextual Properties in Word Representations from BERT

Pernille Aaby

Exploring Multilingual and Contextual Properties in Word Representations from BERT

http://www.oslomet.no/

Printed: Oslo Metropolitan University

Abstract

Nowadays, contextual language models can solve a wide range of language tasks such as text classification, question answering and machine translation. These tasks often require the model to have knowledge about general language understanding, like how words relate to each other. This understanding is acquired through a pre-training stage where the model learn features from raw text data. However, we do not fully understand all the features the model learns through this pre-training stage. Does there exists information yet to be utilized? Can we make predictions more explainable? This thesis aims to extend the knowledge of what features a language model have acquired. We have chosen the model architecture BERT and have analyzed its word representations from two feature perspectives. The first perspective investigated similarities and dissimilarities between English and Norwegian word representations by evaluating their performance on a word retrieval task and a language detection task. The second perspective analyzed how a word representation changes if the word stands in the wrong context or if the word was inferred through the model without context.

Acknowledgement

I would really like to thank my supervisors Gustavo Mello and Anis Yazidi for the support and feedback on the project. I truly appreciate the time they have put into helping me in scoping and improving the project. I would also like to thank Fabrizio Palumbo for great discussions and useful feedback. Lastly, I would like to thank friends and family that has curiously listened and asked about what I do in my thesis.

Contents

List of Figures

List of Tables

Acronyms

AOC Average Over Contexts.

BERT Bidirectional Encoder from Transformers.

BoW Bag-of-Words.

CBoW Continuous Bag-of-Words.

CNN Convolutional Neural Network.

ISO Isolated Embeddings.

KNN K-Nearest-Neighbour.

ML Machine Learning.

MLM Masked Language Modelling.

NLP Natural Language Processing.

NSP Next Sentence Prediction.

RNN Recurrent Neural Network.

WSD Word Sense Disambiguation.

Chapter 1

Introduction

In recent years the field of Natural Language Processing has advanced significantly. We exploit machines to translate from one language to another, answer questions, classify words in grammatical categories, search documents for relevant passages, and more (Otter et al., 2020). Much of the recent success is due to the advent of transformer architecture (Vaswani et al., 2017), which was initially adopted to improve machine translation, though later, the architecture has been applied to a range of NLP tasks with a variety of models. BERT (Bidirectional Encoder Representations from Transformers)(Devlin et al., 2019) is one such transformer-based model. It surpassed the performance on several benchmarks like the question answering on SQuAD v1.1 (Rajpurkar et al., 2016) and SQuAD v2.0 (Rajpurkar et al., 2018), and language understanding tasks like GLUE (A. Wang et al., 2018) and MutliNLI (Williams et al., 2017), at its release time. However, just like other transformer-based models, we are not fully aware of why BERT performs so well.

Before BERT is trained for one specific task, it is pre-trained on raw text data. In the pre-training the model learns to encode words into *contextual word representations*. The contextual word representations contain different properties about the words. We are interested in understanding more about what information is there in the word representations after only pre-training. Are there language features which yet may be better utilized in task

solving? Can we make the models' predictions more explainable? For example, what if the model has acquired a helpful feature to machine translation, but the model is not currently used for this task. The more we know about the model and how it encodes words into features, the better we can utilize that information.

We are not the first ones interested in analyzing the pre-trained features of BERT, *probing* BERT has become such a popular area that it is called BERTology (Rogers et al., 2020). Probing involve creating a classifier with only a restricted set of parameters on top of the pre-trained features. If the classifier manages to solve the task, we assume that the necessary information already exists in the encoded word representations. It has already been proven through probing, that BERT can encode words with semantic (properties in terms of meaning) and syntactic (properties in terms of grammar) properties (Rogers et al., 2020).

Our aim for this thesis was to extend the probing of BERT through an exploratory analysis focusing on multilingual and contextual properties. In our analysis, we have analyzed multilingual information by comparing word representations in English and Norwegian and we have analyzed contextual information by comparing a word's representation in a real, in a wrong (misplaced) and without context. The core remained the same in both analysis: comparing word representations with the non-parametric method KNN (K-Nearest-Neighbors) and cosine similarity.

The first analysis presents an in-depth study of how the word representation for English words and Norwegian words share properties but also how they have distinctly different properties. We show similarity by proving the word representations' power to retrieve a correct translation from an English source to Norwegian target vocabulary. Previous work in word retrieval between other languages using BERT (Cao et al., 2020; C.-L. Liu et al., 2020) inspired us to do the same for English and Norwegian. We also demonstrate how word representations differ for two languages by detecting the language through language-specific features. To our knowledge, no previous work has used this

method to detect each word's language before.

The second analysis investigates the encoding of a word representation if they appear in the wrong context (induced words in sentences where they do not belong) and compares it to how the model represents a word without context. It has already been demonstrated that word representations without context perform poorly on semantics benchmarks (Bommasani et al., 2020; C.-L. Liu et al., 2020). We believe that our study can help explain some of the effects behind the poor performance and emphasize the importance of context on a word's representation.

The overview of this thesis is the following. In chapter 2 *Background and Related Work*, we give necessary background to the topic of NLP and word representations as well as the transformer architecture. Chapter 3 *Distilled BERT Embeddings for Norwegian* explains the method according to which we created a static set of word representations. In the next two chapters, we divided the experiments into two because each of them focused on a different perspective. Chapter 4 *Multilingual Representation Analysis* compares word representation from English and Norwegian while chapter 5 *Contextual Property Analysis* compares word embeddings from different contexts: real, wrong and without. In chapter 6 *Discussion* we discuss and compare the results from both experiments. Lastly, we sum up the analysis in chapter 7 *Conclusion*.

Chapter 2

Background and Related Work

2.1 Natural Language Processing

Natural Language Processing concerns all computational processing of natural language. Already in the 1950s, Turing (2009) defined a task that involved automated interpretation and generation of natural language. Even though the most trendy era of machine learning for NLP had not begun, there existed other rule-based applications (Pollard, 1987; Schank & Abelson, 1975; Schank & Colby, 1973; Weizenbaum, 1966). Today the field has expanded to solve various problems ranging from language understanding and language generation to speech recognition. This thesis will consider the language processing sub-field called natural language understanding and limit us to text data.

Natural language understanding characterizes as an AI-complete or AI-hard problem (Mallery, 1988). A simple algorithm cannot solve an AI-complete problem because it might run into *unexpected circumstances* in the real world. By unexpected circumstances in natural language, we mean that one will never be able to map all possible phrases of a language because there are always ways to make new ones. This phenomenon is also called the *knowledge acquisition bottleneck* (Gale et al., 1992). Consequently, researchers have put tremendous effort into creating different methods for language representation, a

way to represent language and finding patterns within it without depending on mapping it all.

2.2 Machine Learning

Machine learning is a data-driven technique where the machine learns patterns in data and later uses these patterns to make predictions or guesses on new data. It is used in computer vision, natural language, and numeric observations. It diverges from traditional rule-based machine processing because one does not explicitly tell the machine what decision to make, but rather the machine bases its decision on the learned patterns.

The learning part takes place in an adaptable model. The model includes specific parameters and numerical values that can change. To adapt the model to the data means that the model slowly changes its parameters to fit patterns it can find in the data. This period is called training or learning. After training, the model is ready to make predictions. This phase can also be called inference. The model does not adopt at inference time but is in a frozen state.

There is a distinction between supervised and unsupervised machine learning. In supervised learning, the input data has a desired output through labels. The learning in supervised machine learning is to make the model adapt to fit the output labels. Unsupervised machine learning, on the other hand, requires no labeled output. In this method, the model tries to find patterns in the raw data itself.

The objective for a machine learning model states what the model should find. We mentioned that in supervised learning, there exists a labeled dataset. An example of a labeled dataset can be a set of “dog“ and “cat” images, and then the labels are then category “dog” or “cat”. The objective then becomes to predict the right “dog” or “cat” label for an image. The objective of text processing can be to fill in a missing word in a piece of text. In this case, one removes the word in a previous process. Because the objective

is not from a label but rather from the raw data itself, this usually refers to unsupervised machine learning. The objective of an unsupervised machine learning technique can also be to gather data points into groups, where the number of groups or some threshold is given beforehand.

Neural networks are a sub-field of machine learning that mimics part of the neural structure in the brain to be able to learn complex patterns. The neurons can also be called cells, which enables the neural network to find relationships in the not linear data. Figure 2.1 is a simple illustration of a neural network. In computational neural networks, the neurons are structured in layers. In this case, all the neurons in one layer connect to the next layer. When a neural network has more than one or more hidden layers, all the layers between the input and output layer are called hidden layers; the neural network is called deep. Neural networks have become a prevalent technique in the last decade, both in computer vision and in NLP (Goodfellow et al., 2016).

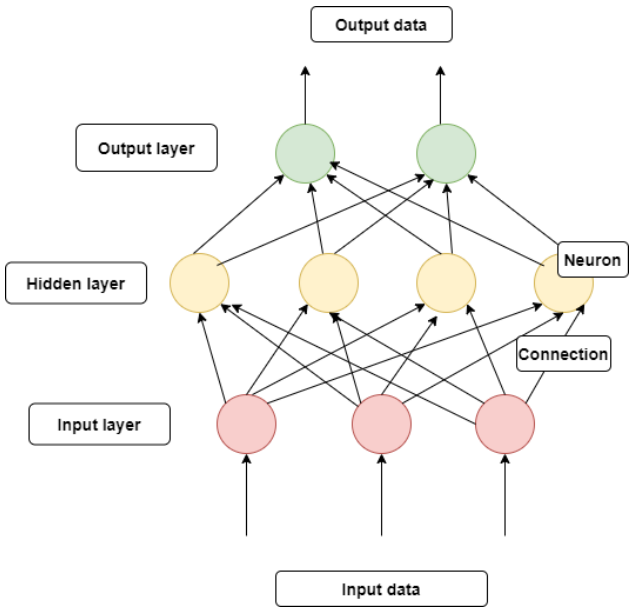


Figure 2.1: Simple neural network with fully connected layers. The circles represents neurons and the arrows represent connections. The model consists of one input layer, one hidden layer and one output layer. All the neurons from one layer is connected to all the neurons in the next layers. This is why we call it fully connected.

In addition to the connected neurons, neural networks often use activation functions. It is called an activation function because using such a function makes the output either activate or not. For example, a straightforward activation function could have a threshold of 0.5. If the calculated sum in the neuron were above this threshold, it would be one and otherwise zero. In this case, one would mean the neuron activated, and zero would mean not activated. The activation function changes the output of the neuron to be binary, like yes or no. Although this explanation is not entirely correct, given that the activation function often does not only output 1 or 0 but continuous numbers, this is the main idea. Either a neuron fires(activates), or it does not.

There exist many different groups of neural networks. They are characterized by which layers the neural network use. The layers are constructed of neurons. The different layers are characterized by how the neurons connect and the activation function. One type of layer structure is the dense layer. In a dense layer, all the neurons in one layer are fully connected to the neurons in the next layer. In Figure 2.1, all the layers are dense. The activation function differs depending on the problem at hand.

2.3 Convolutional Neural Networks and Recurrent Neural Networks in NLP

Convolutional Neural Networks (CNNs) are networks often used in processing visuals, such as images, and their architecture are inspired by the biological processes in how the neural connectivity is structured in the visual cortex of an animal (Goodfellow et al., 2016). The central concept of a convolutional neural network is that it learns hierarchical features, or feature maps, by convolving over the image with different filters. CNNs have also been used in NLP, for example, in character aware models (Kim et al., 2016) or machine translation with finite contextual windows (Gehring et al., 2017).

Recurrent Neural Networks deal with sequential data. They process one input at a time, keeping a state of history called hidden state, which enables the processing to store information about previous input (Lee et al., 2020). Since language, like text or speech, is sequential data, the network is popular in NLP. It has been used in a range of NLP tasks like machine translation (Wu & et al., 2016), text classification, and question answering from deep language representation (Peters et al., 2018). This thesis will not focus on RNNs and their contextual representation because we are more interested in another architecture, namely the transformer, which we will get to later in the chapter.

2.4 Text Processing Pipeline

To be able to process text in a neural network, the text requires to go through a pipeline of preparation steps. This section will describe three common steps that are present in almost every application that uses text modeling.

2.4.1 Formatting and Cleaning Text

The first step in making text ready for processing usually includes formatting and cleaning. It is not expected that text data come in a cleaned text file. In many cases, it might even be divided into multiple files that need to be merged into one, where the format is text. After the text is in the proper format, cleaning is standard procedure. The degree of how much text cleaning is necessary varies a lot, and it is becoming more and more usual to include the whole text as it is (Devlin et al., 2019). Cleaning can include removing stopwords, which are the most common words in a language, lemmatization, which is only keeping the lemma or root for each word, and removing special characters. The amount of cleaning all depends on the model and the text application. We do not detail all the cleaning steps, because in this thesis, we kept the text very close to how it is in the raw format, including special characters and stopwords and having the words with their complete forms.

2.4.2 Tokenization

Tokenization entails chopping the text into smaller units, such as words and characters. The units do not have to be words. There is also sentence tokenization and ngram tokenization. An ngram of words means n-words together, like bigram is two and to words, “of words,” “words means” and so on, trigrams is three and three words “of word means,” “words means n” and so on. While ngrams is a possibility and can prove helpful in several applications, when we talk about tokenization in this thesis, we are mainly referring to how the text is divided into words and characters or *wordpieces* (Wu & et al., 2016), as we will get to later in this chapter.

There exist many pre-defined tokenizers. Both Spacy[1] and NLTK[2] have their respective ones for word tokenization. In this project, we mainly used Spacy for Norwegian. However, we sometimes resorted to NLTK. The difference is often small between the two, and we do not think this affected our result in any significant meaning. In Table 2.1 we have found a sentence from BBC news (Schraer & Mwai, n.d.) and tokenized it with both Spacy and NLTK. In this example the sentence is tokenized the same way by both tokenizers.

[1] https://spacy.io/
[2] https://www.nltk.org/

Original	South Africa was where the new Omicron variant was first identified, and cases there have taken off rapidly.
Spacy	[“South”, “Africa”, “was”, “where”, “the”, “new”, “Omicron”, “variant”, “was”, “first”, “identified”, “,”, “and”, “cases”, “there”, “have”, “taken”, “off”, “rapidly”, “.”]
NLTK	[“South”, “Africa”, “was”, “where”, “the”, “new”, “Omicron”, “variant”, “was”, “first”, “identified”, “,”, “and”, “cases”, “there”, “have”, “taken”, “off”, “rapidly”, “.”]

Table 2.1: Example of tokenization. The table shows how two tokenizers split a sentence taken from BBC News in a set of smaller units. The tokenizers Spacy and NLTK uses word tokenization and splits the sentence exactly the same.

2.4.3 Bag-of-Words Representation

Since computers do logical operations, they are better at processing numbers, which are stronger related to logic, than raw text. Consequently, we need to translate our token into a numeric representation. BoW (Bag-of-Words) is one technique for transforming text into numbers. The idea behind BoW is that one has a “bag,” which is a vector in the size of the vocabulary, where the vocabulary is often set to all the unique words in an entire dataset after cleaning, and then one counts the number of each token in the relevant piece of text. In Table 2.2 we illustrate how part of this vector would look for the tokenized example in Table 2.1.

Vocabulary	Africa	was	second	...	last
Count	1	2	0	...	0

Table 2.2: Bag-of-Words example. The words in the BBC News sentence is counted, and the count for each word is set in the index of the tokens position. The word Africa stands once, so the index of Africa is populated with a 1.

BoW is a method that does not require many calculation steps to represent text as a numeric sequence. Therefore, it is mainly suited for applications where only the combination of specific words is essential in the application. However, there are multiple downsides to this technique. One of the downsides is no notion of how the words relate to each other. For example, the words “first” and “second” relate because they both are order numbers. However, when we represented the words as only an index in a vocabulary, it tells nothing about the relationship between them.

2.5 Language Representation

To find language representation that deals with the knowledge acquisition bottleneck one need to train a model on a vast corpus. Huge, labeled text datasets are very time-consuming and expensive to make (Ng et al., 1997). However, researchers found that using raw text data and finding patterns in how the words stood in context to each other could teach the machine useful features about language (Joulin et al., 2017; Mikolov, Chen, et al., 2013; Pennington et al., 2014). These features are represented as a sequence of numbers, where the relation between the numeric representations shows similarities and associations in language. The language representation learned from this process has proven to improve state-of-the-art results in many downstream applications of AI-complete problems such as machine translation, information extraction, and question answering (Brown et al., 2020; Radford et al., 2018; Wu & et al., 2016).

2.5.1 Pre-Training and Fine-Tuning

Today, computational language models divide their training into two phases: pre-training and fine-tuning. Pre-training involves finding patterns in raw data, aiming to represent the text with valuable features. Most downstream applications also require a fine-tuning process. The fine-tuning process trains the model on a specific task with labeled data. In

the fine-tuning process, the pre-trained language representation works as an initialization with general information about natural language. The unsupervised technique for language representation can be thought of as learning features about the text but not knowing which features relate to the task one want to solve. The model learns this information later in the fine-tuning period with supervised learning.

2.5.2 Word Embeddings

One way to represent language is to embed words into numeric vectors, where the distribution in the vector space would determine certain features, such as semantics, about the words. The idea of distributing words in vector space is rooted in the statement by Firth 1957): “a word is characterized by the company it keeps”. The technique has been around for a long time and started with information retrieval using vector space models (Salton, 1962; Salton et al., 1975).

Although the idea of representing words as numeric vectors has been around for a long time, recent advances in the field related in the last decade have indeed given popularity to word embeddings through models like Word2Vec (Mikolov, Chen, et al., 2013), GloVe (Pennington et al., 2014), and FastText (Joulin et al., 2017). These word embedding models have used neural architectures to find compressed word representations based on co-occurrence patterns in raw text. One of the papers explains how one can find mathematical relations between the word representations with the analogy example; vector(“King”)-vector(“Man”)+vector(“Woman”) should be the closest vector to vector(“Queen”) (Pennington et al., 2014). It has also been proven that these word embeddings have semantic features with other similar words. As an example, it would be expected that the most similar vector to the vector(“Woman”) would give results such as vector(“Girl”), vector(“Female”), and vector(“Lady”).

In Figure 2.2 we have created an imaginary example in 3D space to explain the

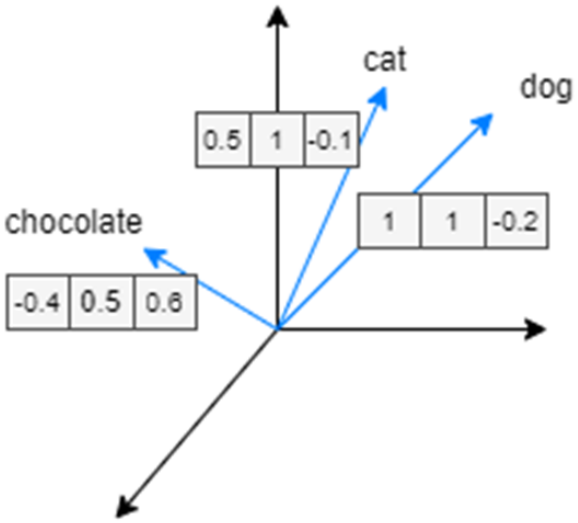


Figure 2.2: Imaginary illustration of word embeddings for “chocolate”, “cat” and “dog”. We used only 3 dimensions to show how a word embedding looks in latent space. The idea is that the two words “cat” and dog are closer to each other in the latent space than “chocolate” because the meaning of the two words are more similar.

concept of word embedding. For example, when looking at just the letters in the three words “chocolate”, “cat” and “dog” then “chocolate” and “cat” are more similar as they share the same letter. However, we are instead looking for semantic relation here (information about syntax is also often encoded into the embedding). Therefore the two house animals, “cat” and “dog” should be more similar.

We can distinguish between word embedding models by how they find co-occurrence patterns in text. The Word2Vec model (Mikolov, Chen, et al., 2013) use a local text window. It can only see the words surrounding that word when predicting the missing word in a piece of text. A window hyper-parameter can be five words. The word embeddings from this model were created from one out of two possible objectives, the Skip-gram model or the CBoW (Continuous Bag-of-Words) model. The Skip-gram model predicts the neighboring words by giving it one word as input, while the CBoW tries to predict the missing word by giving it the neighbor words. In Figure 2.3 we illustrate the process of

CBoW with one example where the window size is five from the tokenized sentence in Table 2.1. We choose the five first words, “was” is the middle word, which the model should predict with the surrounding four words as the input. Doing this process over a lot of data, where Word2Vec is a shallow neural network, creates a numeric projection for each word, which becomes its word embedding.

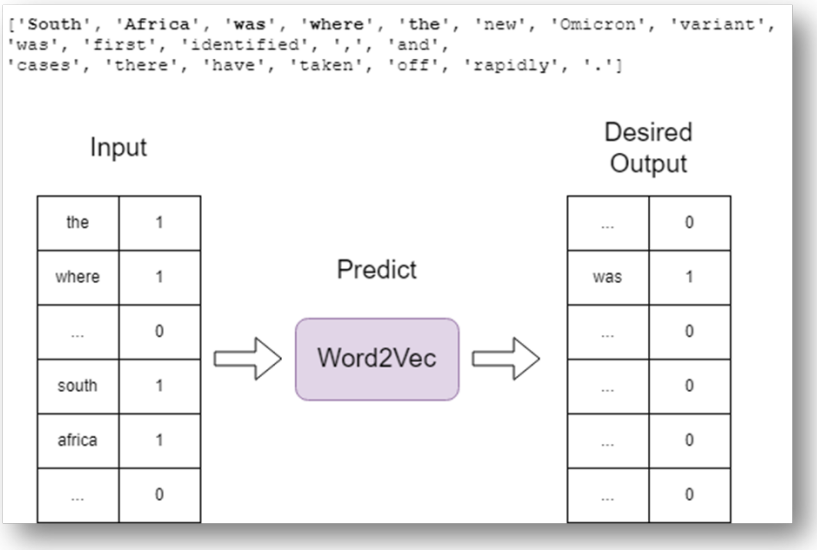


Figure 2.3: CBoW training example. The Word2Vec model takes as input the surrounding words to “was” and the model tries to predict the missing word.

The Global Vectors for Word Representations (GloVe) (Pennington et al., 2014) model uses a count-based method to find global statistics on how words co-occur. It is trained on the aggregated global word-word co-occurrence matrix, showcasing interesting linear substructures of the word vector space.

Word2Vec and GloVe might use different methods to create their word embeddings. However, they both treat each word with one specific word embedding and take no notion of the inner character structure of each word. It is exactly this inner character structure that fastText (Joulin et al., 2017) wanted to improve. They used similar methods as Word2Vec with CBoW and Skip-gram, but instead of representing each word in a word embedding,

they represent all n-gram characters that a word is made up of and then add them together. The 1-gram character of “player” is “p”, “l”, “a”, “y”, “e” and “r”, the 2-gram characters is “pl”, “la”, “ay”, “ye” and “er” and so on. In fastText they use all n-grams characters with n=3 to n=6 (Joulin et al., 2017). The n is also a hyper-parameter that can be changed. FastText has shown to be especially useful for morphologically rich languages, where the inner character structure is important for measuring similarity.

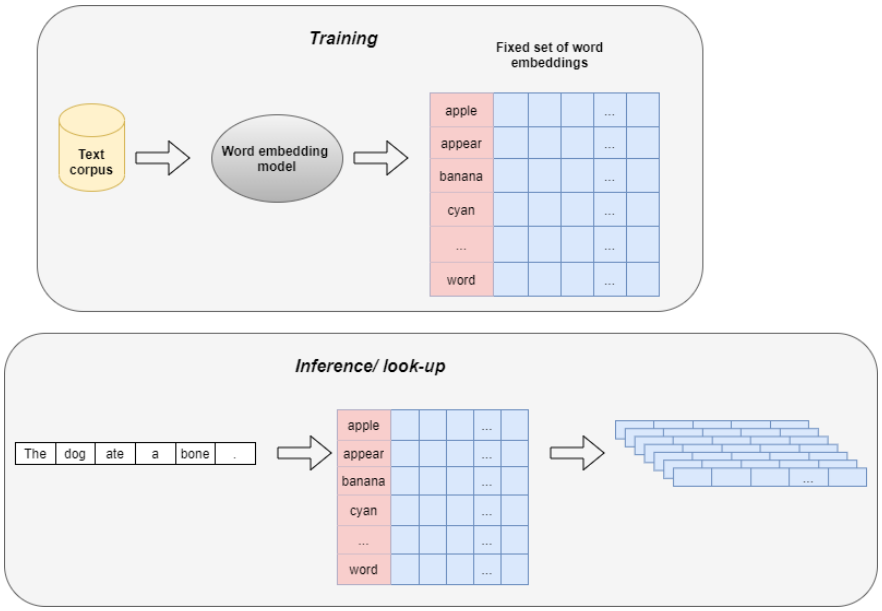


Figure 2.4: Illustrated flow of word embeddings. The model first learn a numeric projection for each of the words in the text corpus. After the model has learned a projection for each word we are left with only the word embeddings and do not use the word embedding model any more. It is then possible to use the word embeddings as input to a new model, which then can solve a specific NLP task.

In Figure 2.4 we illustrate the process and resources used to create word embeddings. After the training process, which learn the word embedding projections, they can be stored in a lookup table. At inference time, or just when one wants to evaluate the vector representation of a word, a sequence of words is embedded in vectors by finding them in a lookup table. We emphasize that models such as Word2Vec, GloVe, and fastText make lookup tables so that when the word embeddings are used in downstream applications,

the word embedding model is not used but input in another model.

The word embedding methods described in the previous paragraphs all suffer from one single embedding per word limitation. This limitation causes problems when multiple meanings of an ambiguous word get conflated into the same numeric representation (Navigli, 2009). When these embeddings are used as input or compared in vector format, they are directly translated from a static and fixed set of previously found vectors from the word embedding model. In other words, when this word-to-number translation happens, the embedding is not affected by the context in which the word appears.

2.6 Transformers

The transformer architecture has had tremendous development since the end of 2017 (Vaswani et al., 2017). Opposed to the word embedding models like Word2Vec, Glove, and FastText, a transformer model is deep with many stacked layers, and they create dynamic contextual representations from tokens. They can create dynamic contextual token representations because the input is not just one token but a piece of text with many tokens. Each of the tokens gets encoded into a representation, where the surrounding tokens also matter in the final representation.

It is important to emphasize that transformers are not only used to create language representation, such as the word embedding models described earlier. The transformers themselves can be adapted to solve NLP tasks directly in applications for end-users such as machine translation. The original architecture translated text between two natural languages (Vaswani et al., 2017). However, in this project, we are mainly concerned with how a transformer-based model represents words with context and how we can use this information directly.

2.6.1 Self-Attention in Multiple Heads

At the core of the transformer architecture is a method called self-attention. Self-attention allows the model to learn more direct relationships between the input words and therefore has proven to be efficient at finding long-range dependencies (Vaswani et al., 2017). For example, in the sentence “Chocolate is a type of candy and it tastes delicious.”; we want to enable the computer to see the relation between, e.g., “Chocolate”-“candy” and “Chocolate”-“delicious,” even though these words are far apart in word position.

The self-attention method consists of different vectors called Queries(Q), Keys(K), and Values(V). These vectors find direct relationships between data points (e.g., tokens or words). The vectors are a linear projection from the previous layer or the input embedding. By matching the Queries and the Keys through a compatibility function, the tokens attend to each other with different magnitude. The magnitude of the Query-Key pairs is then put into a distribution function to make the magnitude of each relation sum to one (Galassi et al., 2020). The new output vector for a given token is then the sum of all the Value vectors from each token, weighted by their distributed relating score. We used the sentence “Chocolate is a type of candy, and it tastes delicious.” to say how the word “Chocolate” can find a relation. In other words, attend to all the other words —the Query for “Chocolate” matched against the Keys for all the other words, even itself. Then, the new output vector for “Chocolate” is the sum of all the Value vectors from all the tokens, weighted by the Query-Key match score.

In Figure 2.5 we illustrate the self-attention function from (Vaswani et al., 2017). The Queries are matched with all the Keys before they are scaled. The scaling is the square root of the number of dimensions. Then they go through the Softmax function to distribute the magnitudes of the vectors. After this, the results are multiplied by the Values, and the output is then new vectors, which is a weighted sum of the Value vector input.

For the model to learn different token relations at once, the self-attention process

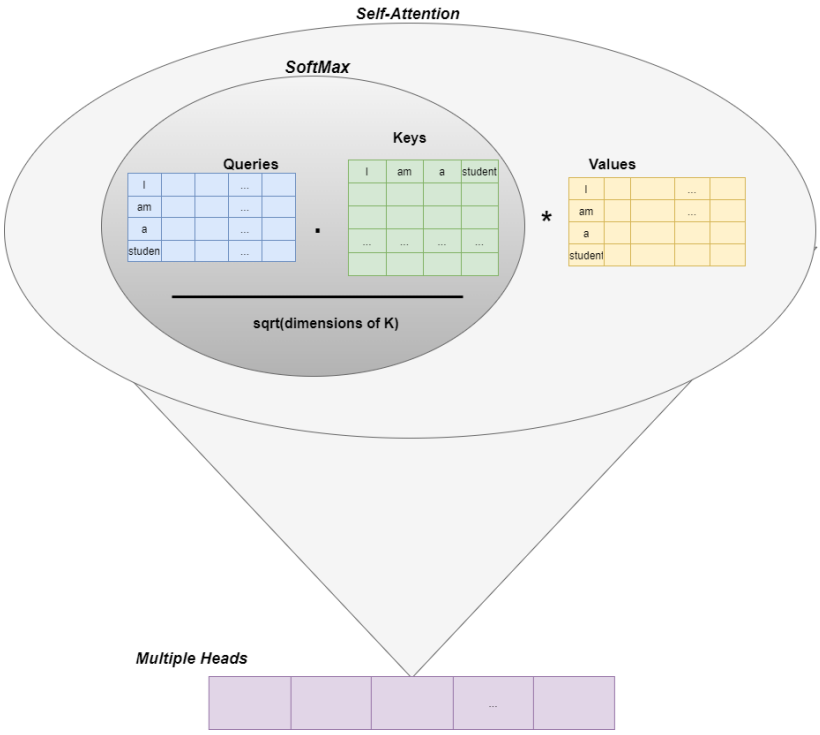


Figure 2.5: Illustrated self-attention concept with multiple heads. Inspiration from the attention function in the original transformer (Vaswani et al., 2017). Each attention layer has 8 heads and within each head the attention mechanism happens. The attention consist of a set of Keys, Queries and Values, which all can be represented as a matrix. Each vector in the matrix corresponds to one input token representation. It is created a dot product from the Keys and the Queries to give the magnitude of attentions. This magnitude is then normalized and used to decide how much of each value vector will create the new output representation for each token representation.

happens in multiple heads in parallel. One head is one self-attention process, where the output vector from the last layer, or input, is divided between each of the heads. For example, the original transformer model (Vaswani et al., 2017) uses eight self-attention heads in each layer.

2.6.2 Direct Global Patterns

Compared to recurrent neural networks and convolutional neural networks, the transformer architecture has achieved many state-of-the-art results in NLP tasks (Brown et al., 2020;

Devlin et al., 2019; Yang et al., 2019) with its self-attention mechanism. The advantage of the transformer architecture is suspected to come from the fact that all the data points in the model are directly matched and can therefore find more global patterns. In a convolutional neural network, the model is often efficient at finding local features within a filter. The filter strides over the input data, but as the features are extracted in a hierarchy, the model might overlook more direct relations between the input. Recurrent neural networks, on the other hand, use a temporal state. Therefore, altering word representation in sequence may cause problems in finding correct relations. For example, suppose there is a strong relationship between a word early in a sentence and another word later in that sentence. The recurrent layers might have difficulty finding this relationship because many intermediate words have processed the state.

But what about the position? If each data point, for example, a word, is directly tested towards all the other words, how can the model have information which reveal the word's position? The authors of the original transformer architecture figured out that they could give positional information to the model by adding a positional encoding to the input (Vaswani et al., 2017). The positional encoding was created so that a linear combination could tell which position the word was standing on (B. Wang et al., 2021).

2.6.3 Encoder with Wordpiece Input

Many transformers are encoder-decoder architectures that deal with *sequence-to-sequence* problems (Sutskever et al., 2014). The sequence-to-sequence problem is machine translation, where one has a sequence of words and characters in and a sequence of words and characters out. However, for the sake of a word representation, we do not need an output sequence of words. We only need the words to be embedded into useful semantic vectors. That is why we have only focus on the encoder part of a model, in other words, only the part that can make words into embeddings. BERT (Devlin et al., 2019) is an example of a

transformer model only based on the encoder part of the original transformer architecture (Vaswani et al., 2017).

In many transformers, the input tokens are wordpieces. A wordpiece can be a whole word, a part of the word, or even just a single character and is found based on a subword segmentation algorithm. The wordpiece method has improved the performance in several NLP problems (Devlin et al., 2019; Wu & et al., 2016). One of the significant advantages of using wordpieces instead of full-form words is the vocabulary length. The wordpiece vocabulary, for example, in the BERT Base model, is restricted to approximately 30,000 wordpieces (Devlin et al., 2019). A vocabulary with massive text corpora, typically a transformer model, can be trained on more than a hundred gigabytes of text, can include millions of unique words. A word can always be represented by wordpieces, where all characters are in the wordpiece-vocabulary. The wordpiece method avoids the *out-of-vocabulary* problem, where a word not present in the training set appears in the test set or at inference time.

The national library of Norway has created a Norwegian version of the BERT encoder model (Kummervold et al., 2021), and below, we give an example of a sentence in Norwegian, and how the sentence splits into wordpieces[3]. The tokens starting with "##" indicate that the wordpiece is not the beginning of a word.

- Hei, mitt navn er Pernille og jeg er student.
- "He" "##i" "," "mitt" "navn" "er" "Per" "##nil" "##le" "og" "jeg" "er" "en" "student" "."

2.6.4 Stacked Layers for Deep Learning

Deep language models such as ELMo (Peters et al., 2018), BERT (Devlin et al., 2019) and GPT-3 (Brown et al., 2020) all have multiple stacked layers. Each layer output an intermediate state from the input tokens, also named a hidden state. It has been proven

[3]https://github.com/NBAiLab/notram

that these internal, hidden states include more information about the structure of language, both semantic and syntactic features, than appreciated in the beginning (Peters et al., 2020). We mentioned earlier that all models try to adapt to an objective. The last layer in the model is, therefore, often very adapted to be able to predict the objective of the training process. However, after putting effort into analyzing more of the hidden internal states, one has found that the hidden states from the lowest layers in a deep language model often have syntactic features and that the upper hidden states carry more semantic features (Loureiro et al., 2021; Peters et al., 2020).

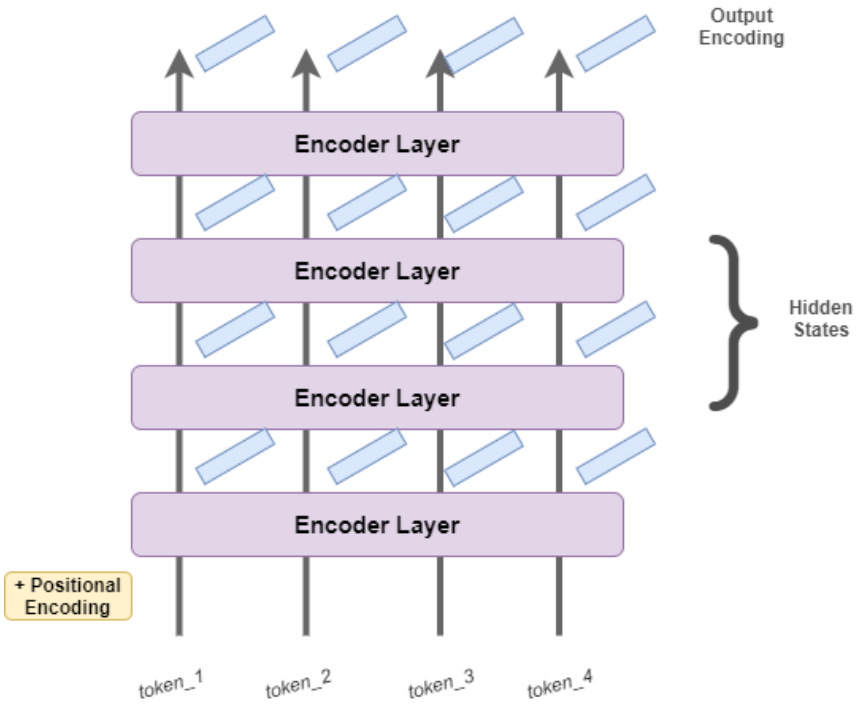


Figure 2.6: Token input to stacked encoder layers in deep language model with intermediate token embeddings. This model is a simplified version meant to illustrate the concept of how layers in a transformer encoder is stacked on top of each other and that after each layer it outputs an intermediate token embedding in the same format. The input tokens all propagate through their own path in the model but in the layers each token can attend to the other tokens. To know the sequence of the tokens a positional embedding is added to the input.

In Figure 2.6 we have created a figure to describe how input tokens is propagated

through encoder layers in a transformer and how it outputs contextual representations for each of the tokens in the output layer. In this figure, there are four stacked encoder layers. However, this number depends on the transformer architecture and is often more than four. The bottom one inputs a set of tokens, e.g., wordpieces, which are processed through each model layer and outputs intermediate states. The intermediate states can also be used as language representation, not just the last output state.

We have not mentioned that these layers include more than just the self-attention mechanism; they also include simple feed-forward layers for each token and residual connections between the layers. A residual connection includes the intermediate state from the last layer to affect the next state. In one sense, it prevents the states from changing too much by including some information from the last state. We will not go into any more detail about these features in this report, but it is essential to know that there is more to the transformer architecture than just self-attention.

2.6.5 Masked Language Modelling

One of, or essentially the main, training objective for BERT is called MLM(Masked Language Modelling). MLM entails masking 15% of the input tokens randomly and then letting the model predict what kind of tokens stands in the masked position. For example, in Figure 2.7 we illustrate the concept with our tokenized sentence and how a token becomes masked, and then the model needs to predict the masked token in the output layer.

From the original BERT paper, we found that the objective is a little more nuanced than masking 15% of the tokens to re-predict the token in the output layer. From the 15% of the masked tokens, only 80% is masked, while 10% is switched out with a random token, and 10% remains itself (Devlin et al., 2019). So in the example Figure 2.7, the token “taken” would be set to “[MASK]” 80% of the times, a random token like “snow” 10% of the times, and remain “take” 10% of the times. They do this to force the model to keep track

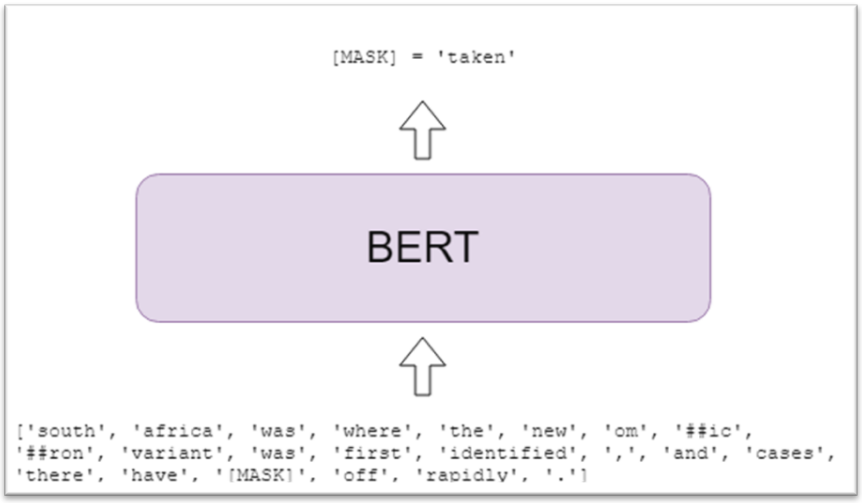


Figure 2.7: Masked Language Modelling in BERT. MLM is one pre-training objective in BERT where a token gets masked and the model tries to predict which token is missing. In the figure we illustrate how BERT masks the token "taken" and tries to predict that this is the missing word.

of relevant contextual embeddings for all the input tokens, not only the ones it needs to predict.

2.6.6 Next Sentence Prediction

NSP (Next Sentence Prediction) is the second training objective that BERT is pre-trained on (Devlin et al., 2019). BERT assigns the first token in the input sequence to be a "[CLS]" or classification token and the last to be a "[SEP]" or separation token. The "[SEP]" token can also be used between two text spans to show separation. In addition, the segment embedding is added to the input embedding of each word, just like the positional embedding, which tells the model which text segment the token belongs to. The model aims to say if two text spans are following each other or not, like in the sentence below the part before the first "[SEP]" token and the part after doing follow each other.

> [CLS] South Africa was where the new Omicron variant [SEP] was first identified, and cases there have [MASK] off rapidly. [SEP]

50% of the time, they create real examples where the answer for NSP is yes, and 50% of the time, they take two random text spans and put them together where the answer for NSP is no. The objective is meant to help the model understand relations between words at the sentence level.

2.6.7 mBERT, Multilingual BERT Trained on 104 Languages

The BERT architecture was originally trained on English monolingual corpora. However, more general models with multilingual abilities have also proved exciting, and one of these multilingual models is called mBERT [4] (Devlin et al., 2019). mBERT is trained on Wikipedia data sampled from 104 languages, where the data from languages with major Wikipedia languages are undersampled, and data from languages with fewer Wikipedia resources are upsampled. Since the model is trained in several different languages, the creators also made a shared vocabulary of approximately 120,000 wordpieces compared to the original English token vocabulary of 30,000 wordpieces.

2.6.8 Notram, a BERT Model Specialized in Norwegian

Notram is a model with BERT architecture specialized in Norwegian (Kummervold et al., 2021). It is trained on a vast amount of Norwegian corpus, including almost all the OCR-scanned books in the Norwegian Library and many other online text resources. Unlike the original base-bert, trained on English corpus and initialized on random weights, Notram is initialized from mBERT and share its wordpiece vocabulary.

For the Notram model, this means that the wordpiece vocabulary is not specialized for the Norwegian language but instead has more multilingual token splitting. In other words, this indicates that a word is more likely to be split into multiple pieces than it would with a Norwegian specialized tokenizer, even though the vocabulary is significantly larger.

[4] https://huggingface.co/bert-base-multilingual-cased

Notram may be mainly trained on Norwegian data. However, it is estimated that around 4% of the data is English corpus, and some are also other foreign languages like Danish and Swedish (Kummervold et al., 2021). In addition, because the model is initialized from a version trained in 104 languages, it has some information about other languages like Finnish and Spanish.

2.7 Related Work

BERTology study the BERT architecture and what patterns it has acquired through training (Rogers et al., 2020). Patterns in terms of language are often divided into two: syntax and semantics. The former describes features related to meaning, while syntax describes how a language is structured, such as grammatical rules. We introduce some research that shows what kind of patterns we are already aware that the model produces in terms of semantics and syntax.

In previous word by Tenney, Das, et al. (2019), they show that BERT acquires hierarchically information that corresponds to the traditional pipeline in NLP. The first layers show helpful information in local syntax structure, such as POS tagging and parsing, named entity recognition, semantic roles, and coreference are information encoded in the middle and later layers of the model. Similar discoveries can be found in other works as well (N. F. Liu et al., 2019; Tenney, Xia, et al., 2019).

Naturally, since BERT is a contextual model representing a word based on itself and the surrounding words, the question of whether one could distinguish different meanings of an ambiguous word arose. More than one paper find that ambiguous words divide different meanings into clusters from the contextual representation, although it is not always the same clusters as we might have expected (Loureiro et al., 2021; Wiedemann et al., 2019). Wiedemann et al. (2019) found that the best-hidden states from the BERT model to solve WSD (Word Sense Disambiguation) were layers 10 and 11.

To benefit from benchmarks like SimLex999 (Hill et al., 2015), WordSim353 (Agirre et al., 2009) and SimVerb3500 (Gerz et al., 2016) which evaluate semantics in static word embeddings, Bommasani et al. (2020) distilled a set of static word embeddings from contextual word embeddings. The resulting static embeddings could be compared to traditional static word embeddings (Joulin et al., 2017; Mikolov, Chen, et al., 2013; Pennington et al., 2014). Different aggregation and pooling strategies were used to create the static word embeddings. The strategies included element-wise mean, maximum or minimum values from the contextual vectors. They report that the mean-pooling strategy achieves the best results. Another important finding from this paper is that using more context examples for a word increases the quality of the word representation (Bommasani et al., 2020). Especially when using the late layers of a deep model, using more context examples significantly improves the quality of the representation. This phenomenon is suspected because more examples de-noise the context-sensitivity and create a more general word vector. They also find mean pooling over subtokens of a word in case a word consist of more than one token to be the best pooling strategy for subtokens. Even though one of the significant advantages of the model is that it is contextual, the distilled word embeddings still outperformed the traditional word embeddings (Joulin et al., 2017; Mikolov, Chen, et al., 2013; Pennington et al., 2014) on several benchmarks, proving that BERT is very good at encoding lexical semantics as well.

Chronis and Erk (2020) use K-means clustering to derive multi-prototype word embeddings from BERT contextual embeddings. They use a set of up to 100 sentences for each word and group the contextual embeddings into a previous set of K groups. The K is within {1, . . . , 10, 50}. In other words, they create many word embeddings for the same word with this prototyping. Still, they only use it to solve word similarity tasks and not WSD. The centroids of the K clusters for each word then become one of the word embedding prototypes. When comparing the similarity between words, they use either the maximum similarity between word groups or the average similarity between word groups. They

report maximum similarity to be the best-performing measure. The paper also proposes which layers are best for two different purposes. They suggest using layer 8 of BERT for semantic similarity and layer 12 for finding relatedness. In addition, they find that words with more abstract meaning, such as stop words, have a higher contextual variance than more concrete words (for example, nouns).

Chapter 3

Distilled BERT Embeddings for Norwegian

A static word embedding can be stored in a look-up table and used for direct comparison. Even though we used a contextual model that produces contextual word embeddings at inference time, we wanted to build this static look-up table so we could compare the representation of different words. To infer text through the model is computationally expensive given the deeply stacked layers with many parameters. Therefore, it can be advantageous to construct a static list before solving the task.

Following the work by Bommasani et al. (2020) we created a static set of word embeddings by taking the AOC (Average Over Contexts) of several contextual embeddings for a word t. The contextual embedding for word t is obtained from a context $c_t \in C_t$, where each c_t is two sentences from the relevant language corpus. The final static embedding then becomes the average of all the contextual embeddings.

$$s_t = \frac{1}{N_t} \sum_{n=1}^{N} w_{tn} \tag{3.1}$$

w_{tn} is the nth contextual embedding for the number of contexts $N_t = |C_t|$. For words that

constitutes of more than one wordpiece we used *mean pooling* to aggregate all of the token embeddings. *mean pooling* averages all the token embeddings in one word.

$$w_{tn} = \frac{1}{I_t} \sum_{i=1}^{I} p_{ti} \tag{3.2}$$

p_{ti} is the ith token in the word and I_t is the number of wordpieces in the word. We created static embeddings for all the 13 intermediate representations of BERT after all the 12 stacked layers and the input layer. Figure 3.1 illustrates the same process.

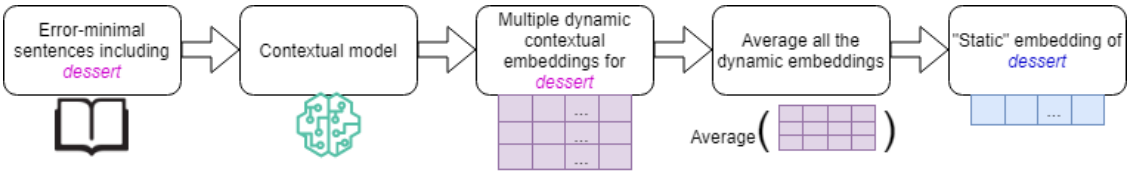


Figure 3.1: Process of how we produced our static word-embedding for the word *dessert*. We used a text corpus with several contexts for each word. We then found a set of contextual embeddings for each word and create the static embedding by taking the average of all these embeddings.

To produce our static word embeddings we used the transformer-based models Notram (Kummervold et al., 2021) and mBERT (Devlin et al., 2019). Both models can encode words in English and Norwegian, which became useful in our analysis.

3.1 The Norwegian News Corpus

We needed a corpus of raw Norwegian text to create our set of static word embedding. This section describe the data.

As a data corpus for raw Norwegian text, we have used the Norwegian News Corpus[1], the part in Norwegian bokmål (not nynorsk). The news articles in the dataset are from multiple different papers such as “VG”, “Aftenposten” and “Dagens næringsliv”, collected from the years 1998-to 2019.

[1] https://www.nb.no/sprakbanken/ressurskatalog/oai-nb-no-sbr-4/

The number of contexts N_t for each word is between 100 and 500. A context is defined as two sentences. The vocabulary is restricted to only include the 50,000 most frequent words from the Norwegian News Corpus. Additionally, we verified that the word is present in a Norwegian wordlist for Bokmål [2].

Initial Formatting

The news corpus is divided into many files from the raw source and not in a simple text format suitable for computational processing. Therefore, we first formatted and cleaned the data. The NBAiLab, the same organization that trained Notram, provides a pre-made script[3] we have utilized to clean and format the news corpus. This merged all the news articles (≈2,000,000) into the same file with double line shift to separate each article.

Text Splitting

To divide each article into sentences, we used the NLTK sentence tokenizer. To create our static embeddings, we constructed text blocks of two sentences to give the context of the words.

For word tokenization, we used the Spacy model for Norwegian Bokmål [4]. The Spacy tokenizer can separate punctuation from the words and is also familiar with certain abbreviations in the Norwegian language. Later in the thesis, we compare Norwegian and English parallel sentences. In this case, we used the NLTK word tokenizer.

The Spacy model is also able to detect language. We used this feature to remove text classified as English. Later we compared our embeddings with English word embeddings, and therefore we wished to keep our vocabulary of word embeddings to include only

[2] https://www.nb.no/sprakbanken/en/resource-catalogue/oai-nb-no-sbr-23/

[3] https://github.com/NBAiLab/notram/tree/fe0fd50b948096cc1af00e2215416a507dd99728/corpus_generation_scripts

[4] https://spacy.io/models/nb

Norwegian contexts. However, removing text in another language can be challenging, especially if only part of the sentence is in English.

Capitalization

Both Notram and mBERT have uppercase letters in their vocabularies. Therefore, we could have chosen to store the words in our vocabulary with uppercase information. However, to avoid complicated handling of uppercase letters, such as at the beginning of a sentence, we used only lowercase in our word embedding vocabulary. During inference time, we still retained the original text, uppercase or lowercase, but when we added the word embeddings to the static vocabulary, we only used the words in lowercase. For example, the contextual word embeddings for "Det" and "det" was averaged into a single word embedding for "det".

3.2 Word-to-Word Comparing

Instead of simply considering absolute similarity values, we ranked similarities for the entire vocabulary to find the closest match. The contextual representations are known to be highly anisotropic (Ethayarajh, 2019), especially in subsequent layers. So the similarities are not uniformly distributed, and we wanted to introduce this relative comparing system instead.

3.2.1 Similarity Measure

To compare word embeddings, we used the distance measure cosine similarity, defined as:

$$\cos(\mathbf{x1},\mathbf{x2}) = \frac{\mathbf{x1}\mathbf{x2}}{\|\mathbf{x1}\|\|\mathbf{x2}\|} = \frac{\sum_{i=1}^{n} \mathbf{x1}_i \mathbf{x2}_i}{\sqrt{\sum_{i=1}^{n} (\mathbf{x1}_i)^2}\sqrt{\sum_{i=1}^{n} (\mathbf{x2}_i)^2}} \tag{3.3}$$

where $x1$ and $x2$ represent two different word embeddings. Cosine similarity is the most commonly used similarity measure for word embeddings (Camacho-Collados & Pilehvar, 2018). It returns the similarity in angular direction between two different vectors, hence it does not depend on the magnitude of the vectors.

3.2.2 Top Matches

When comparing word-to-word, we first found the embeddings of one word, either from the static word embedding vocabulary, or we found the contextual word embedding for the word with context. Then we found the cosine similarity to all of the words in one vocabulary of distilled BERT embeddings, English or Norwegian depending on the task. We define a top matches with an @, so if we return only the most similar word embedding from the vocabulary this would be stated as @1, and if we wanted to return 10 matches then it would be stated as @10.

3.2.3 Comparing the Embeddings in Different Layers

The BERT architecture has 12 stacked layers. After each of these layers, it is possible to extract an intermediate representation. Since there is also the input layer, we can extract 13 intermediate representations. We were interested in analyzing not only the last of the intermediate states but all of them because it gave us some information about which layer representations perform best for different tasks. Additionally, since we were exploring how meaning is activated by context, it was fascinating to observe which layers of the context seemed to affect the contextual embedding of a word.

Chapter 4

Multilingual Representation Analysis

Multilingual language models, such as mBERT, are known to be able to solve tasks on datasets for different languages. However, there is also a new branch of research that investigates information shared and not shared among languages (Cao et al., 2020; Litschko et al., 2021; C.-L. Liu et al., 2020). This analysis have studied how the two languages, English and Norwegian, and their respective word representations relate. First, we attempted to discover similarities between the two languages by determining whether the word representation can relate to its translation. In the second part, we examined the differences by testing whether it was possible to distinguish a word in English from a word in Norwegian. Throughout the analysis, we compared the two models, mBERT and Notram, to evaluate whether there is any difference in performance.

4.1 Related Work on Multilingual Word Retrieval

Unsupervised multilingual word retrieval does not rely on parallel corpora as most traditional machine translation systems do. Mikolov, Le, et al. (2013) noticed that the distribution of word embeddings in latent space showed similar characteristics across different languages. Motivated by the similarity of distributions, they hypothesized that they could align two

distributions with word embeddings from two different languages to create a bilingual dictionary using word retrieval. Their technique relied on bilingual parallel corpora. Later, Conneau et al. (2017) showed that it was possible to align two word embedding distributions from different languages without any supervision. They utilized adversarial training to learn a linear mapping from source to the target language, alleviating the need for parallel corpora.

Conneau et al. (2017) relied on matrices to align two word embedding distributions in different languages. However, recent discoveries show that multilingual BERT, partially aligns the semantics in two languages automatically (Cao et al., 2020). Hence, word translation can be produced by simply looking for the most similar word embedding in the target language.

4.2 The Bilingual MUSE Benchmark

To evaluate the word retrieval from English to Norwegian, we used the English-Norwegian word benchmark from MUSE [1] (Conneau et al., 2017). MUSE is a benchmark that exists for several language pairs. Each language pair consist of an extensive set of word pairs like a bilingual dictionary. The benchmark has one training part, one development part, and one test part. However, since we only relied on unsupervised or techniques, we included the entire MUSE dataset when selecting the words to test our word retrieval. We only used the word pairs, where the Norwegian word is in the top 50,000 words from the vocabulary, and the English word is present in the Brown corpus[2]. Some English words have more than one Norwegian word translation. We define a *correct word retrieval* as at least one match from the possible translations.

[1] https://github.com/facebookresearch/MUSE
[2] https://www.nltk.org/nltk_data/

4.3 The English Brown Corpus

Contexts for English words are derived from the Brown corpus (Francis & Kucera, 1979), where N_t denotes the number of times a word occurs in the Brown corpus but a maximum of 500 times. The Brown corpus is much smaller than the Norwegian News Corpus, so the number of contexts is likely to be fewer than in the Norwegian contexts. In addition, the Brown corpus is pre-tokenized, so there was no need to use an additional tokenizer. Like with the Norwegian News Corpus, we also created a context for a word to be two sentences. As for the English vocabulary, we only obtained static word embeddings for the words in the MUSE benchmark, which resulted in roughly 12,000 words in the English source vocabulary.

4.4 Word Retrieval with Static Embeddings

To demonstrate that BERT aligns the semantics of the two languages English and Norwegian, without supervision, we conducted an experiment that attempts to retrieve a Norwegian target translation from an English source word. We did this using only the non-parametric method KNN from the static word embedding vocabularies. Figure 4.1 shows the two collections of static word embeddings. Each intermediate state was evaluated. By evaluating which layer performs best, we could test wherein the model cross-lingual semantics, which is information that two languages share, was most present.

4.4.1 Retrieving a Word Translation

To find the closest word embedding match, we ranked cosine similarities. For each English word in our benchmark, we used the static embedding of the English word and then matched that word with all the Norwegian word embeddings to rank the most similar. We

Figure 4.1: Two static word embedding vocabularies. The source vocabulary is English, while Norwegian is the target vocabulary. After creating static word embedding, this is what we were left with and used to search for embeddings. Each row corresponds to one word and its word embedding in numeric vector format.

then returned the top matches, to which we can say we used KNN with a K set to 1, 3, and 10. From the @1, @3, and @10 matches, we evaluated if one of the words was a valid Norwegian translation of the original English word.

$$\textit{K-Neighbours(i)} = \underset{j}{\operatorname{argmax}}\ sim(s_{i-en}, s_{j-no}) \tag{4.1}$$

$$y_i = \begin{cases} 1, & \text{if } K-Neighbours(i) \in translation(s_{no}) \\ 0, & \text{otherwise} \end{cases} \tag{4.2}$$

The accuracy of the word retrieval task is defined as all correct translations within K,divided by all the words tested (all the words in the English source vocabulary).

$$\textit{accuracy word retrieval} = \frac{1}{T}\sum_{i=1}^{T} y_i \tag{4.3}$$

T is the number of terms in the English vocabulary.

In Figure 4.2 we illustrate with a “cake”-word example. First, we have the English

word “cake” and its static embedding. Then we compare that embedding to all the 50,000 static embeddings from the Norwegian target vocabulary. In this case, the ideal scenario is if the Norwegian word embedding for “kake” is returned.

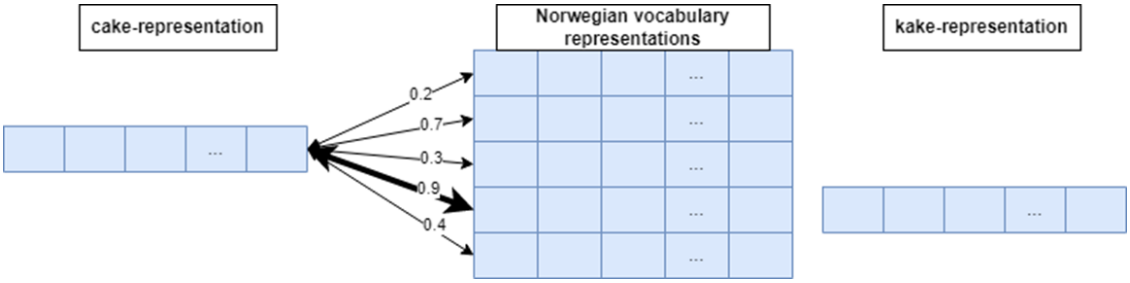
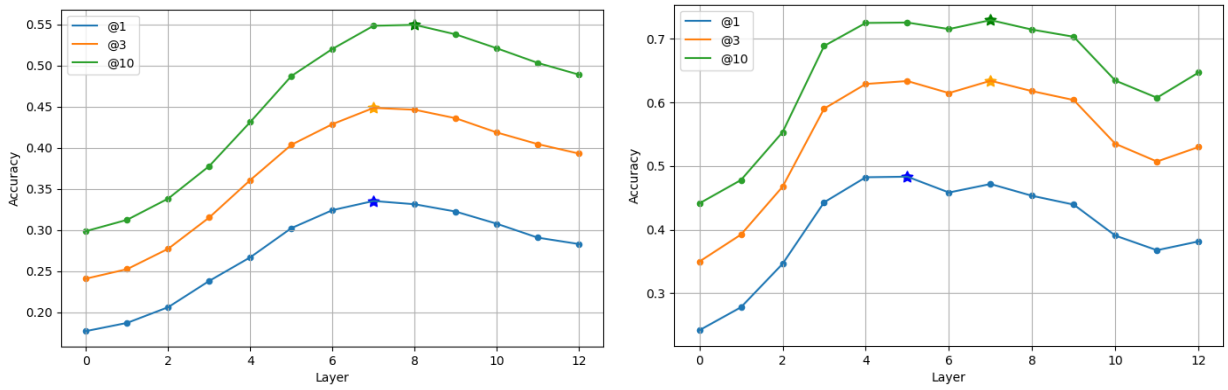


Figure 4.2: Word retrieval example of an ideal situation where the English word “cake” matched closest with the Norwegian word “kake”. The English word is matched with all the words from the Norwegian vocabulary using cosine similarity. The cosine similarities are then ranked by size, and the best match, in this case, “kake” is returned.

4.4.2 Results Static Word Retrieval



(a) mBERT **(b)** Notram

Figure 4.3: The figure shows static word retrieval performance from English to Norwegian with layer-wise performance accuracy. The bottom line represents the accuracy when only including the closest match, @1, while the middle includes @3 and the top line @10 matches—the star marker shows at which layer the performance peak.

In Figure 4.3 we report the result of the English to Norwegian word-retrieval using

KNN and cosine similarity. The performance comparison between mBERT and Notram show that Notram achieved better accuracy than mBERT in general. The middle layers seemed to perform best for both models, with Notram achieving around 50% at @1 match and more than 70% accuracy when using the @10 matches at layer 7. Another part to notice is the dip in performance in layer 11. Overall, we can at least argue that BERT does very well in aligning semantics across two languages without using any supervised dataset with parallel sentences.

4.5 Improving Static Word Retrieval with Mean Shift

C.-L. Liu et al. (2020) showed that words are represented with language-specific information and that applying a *mean shift* helps to improve the translation ability. Although proved for other languages our results show a comparison between the English and the Norwegian language.

4.5.1 Language Specific Embeddings and Mean Shift

We construct a "characteristic vector" for language by taking the mean of all the static word embeddings in each respective language vocabularies.

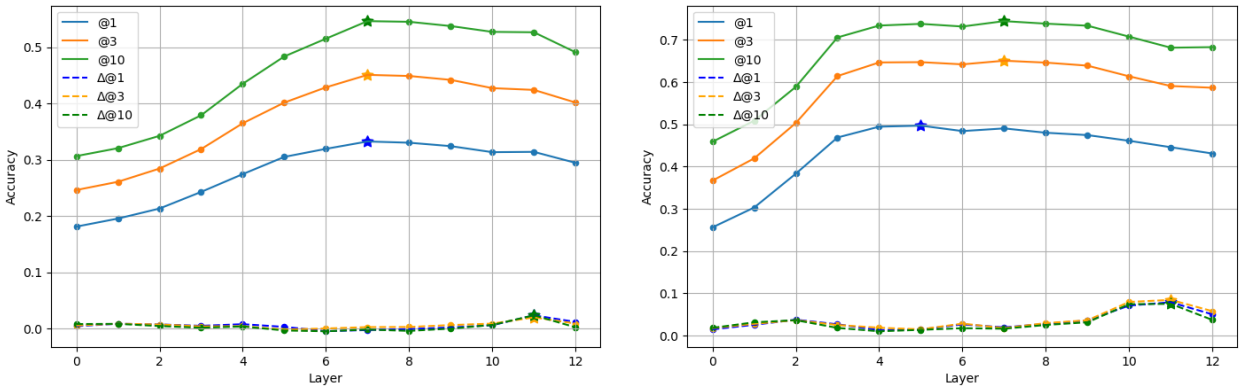
$$L_l = \frac{1}{T}\sum_{t=1}^{T} w_t \tag{4.4}$$

$l \in \{English, Norwegian\}$ and T is the number of words in each vocabulary. We try retrieving the Norwegian word for a English embedding by doing an additional mean shift (C.-L. Liu et al., 2020) with:

$$s_{t,en->no} = s_{t,en} - L_{en} + L_{no} \tag{4.5}$$

where the L_{en} and L_{no} are language characteristic vectors for English and Norwegian, respectively.

4.5.2 Results Word Retrieval with Mean Shift



(a) mBERT **(b)** Notram

Figure 4.4: The graph shows static word retrieval performance from English to Norwegian with layer-wise accuracy with mean shift. The lower dashed lines show the increased performance in word retrieval using the mean shift method. Again the stars show their peaking performance. For both models, the highest increase in performance for all @1, @3, and @10 happens in the 11th layer.

Figure 4.4 show the same type of graph as in Figure 4.3, however, this time with mean shift. The graph with mean shift drops less performance accuracy from the middle layers to the subsequent layers than the graph without mean shift. In layer 11, the word retrieval performance for the Notram model improves by 8% for K at @1, @3, and @10.

4.6 Cross-Lingual Sense Disambiguation

English and Norwegian share several words, some with the same meaning but others that do not share the meaning. To better understand the semantic power of the model, we inspected how a word both in the English and Norwegian dictionaries relates to different

words depending on the context in qualitative analysis. In Figure 4.5 we illustrate how the English and Norwegian vocabularies in some cases overlap by drawing a Venn diagram. The words we were interested in correspond to the lower part of the middle section, marked with bright purple color.

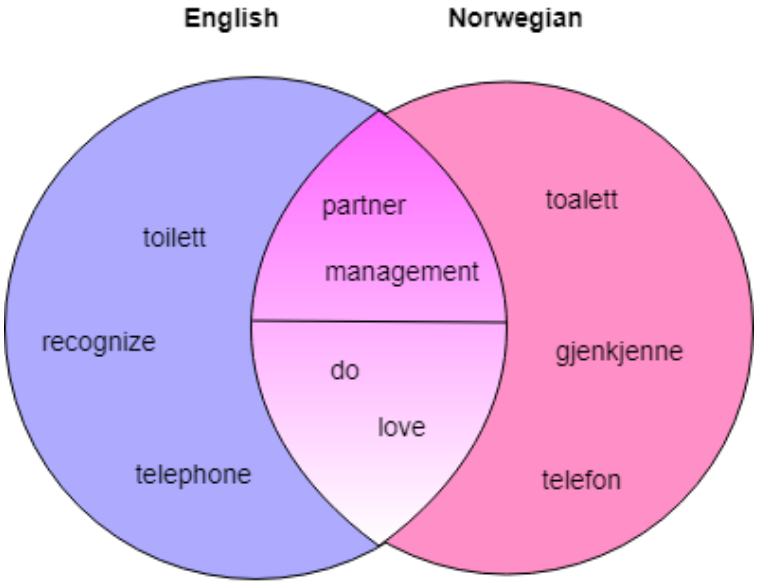


Figure 4.5: Venn diagram illustrating how the English and the Norwegian vocabulary partly overlap. Some words, like in the top part of the sharing part, marked with gradient purple, have words in both vocabularies that also share meaning. In the lower part of the purple gradient, we have words shared by both vocabularies but with a different meanings.

Table 4.1 gives two example words, “do” and “love” used in an English and a Norwegian sentence. The meaning of the word “do” can translate to the Norwegian word “gjøre.” However, the Norwegian meaning of “do” can translate to the English word “toilet”. The meaning of the word “love” when used as a verb in English means “elske” in Norwegian, while the Norwegian verb “love” can be translated to “promise” in English. We used one sentence in English and one in Norwegian with the two respective words and show what their closest matches from the Norwegian and the English vocabulary are. In all four examples, we see that the word relates to words with a similar meaning in their relevant context. More examples can be found in Appendix A.

Sentence	Language	@3 Norwegian	@3 English
Can you please **do** it?	En	gjøre, gjort, gjør	do, done, doing
Jeg må gå på **do** en tur.	No	do, toalettet, doen	toilet, bed, dock
Do you **love** me?	En	elske, elsker, elsket	love, loved, loves
Kan du **love** meg at vi ikke behøver å vente tre timer i kø?	No	garantere, lovet, lovte	promised, assure, assured

Table 4.1: Example sentences in English and Norwegian share a word but with different meanings. The word embeddings are fetched from layer 7. We see how each word related to words in each word embedding vocabulary for English and Norwegian, respectively. We see how the senses are disambiguated based on the meaning in each language.

4.7 Word Retrieval with Contextual Embeddings

Following previous work by Cao et al. (2020) we also tested if we could find the correct contextual embedding of a word, using aligned word pairs from parallel sentences[3] (Riksrevisjonen, 2018). The parallel sentences we use were created by the Norwegian organization “Riksrevisjonen”.

4.7.1 Visualizing Aligned Word Pairs from English and Norwegian

In Figure 4.6 we introduce a similar plot to the work by Cao et al. (2020), which originally compared English and German with the mBERT model. This figure compares contextual embeddings from layer 8 between English and Norwegian word pairs. The contextual embeddings are reduced to two dimensions with t-SNE (der Maaten & Hinton, 2008). The scatter color decides which language the contextual embeddings belong to, while the marker of the scatter decides which word the contextual embeddings belong to. It is clear from the plot that a word pair clusters together cross-language, though it does not always overlap. The exception is the word par “year-år,” marked with $+$, which overlap.

[3] https://www.elrc-share.eu/repository/browse/bilingual-english-norwegian-parallel-corpus-from-the-office-of-the-auditor-general-riksrevisjonen-website/a5d2470201e311e9b7d400155d0267060fffdc9258a741659ce9e52ef15a7c26/

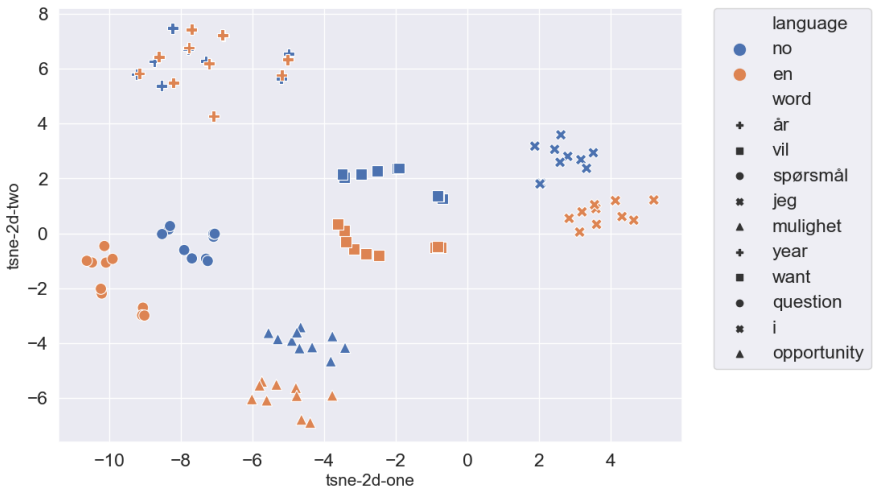


Figure 4.6: Visualizing contextual word embeddings for word pairs in English and Norwegian. Contextual embeddings taken from layer 8 of the Notram model and the English contextual embeddings have experienced a mean shift. The blue colored markers show contextual word embeddings for Norwegian while the orange color shows contextual embedding for English. Each word pair has its marker, so it is easy to spot which embeddings belong together.

4.7.2 Aligning Word Pairs for Comparison of Contextual Embeddings

We simplified the alignment of word pairs by using the MUSE benchmark as our valid translation. First, we tokenized the English and the Norwegian sentence with NLTK. Then, we checked if it existed in our MUSE benchmark for each English word in the sentence. If a valid Norwegian translation was found in the Norwegian counter sentence, we accepted the word pair. Finally, we removed every pair with more than one possible translation to ensure that the word pairs were the aligned versions. For example, it occurs if two of the same words are in the same sentence.

The process of retrieving a word becomes very similar to comparing static embeddings. We first created a vocabulary with contextual embeddings and no averaging with the word embeddings for all the words in the word pairs with English and Norwegian, respectively. Then for each of the English contextual word embeddings, we found the closest match in the Norwegian contextual vocabulary. We achieved a correct retrieval if we re-found the

corresponding word in the Norwegian vocabulary.

To find relevant word pairs, we sampled the first 5,000 sentences of the dataset and use MUSE to find relevant pairs. From this selection, we have listed the unique words and word pairs in Table 4.2.

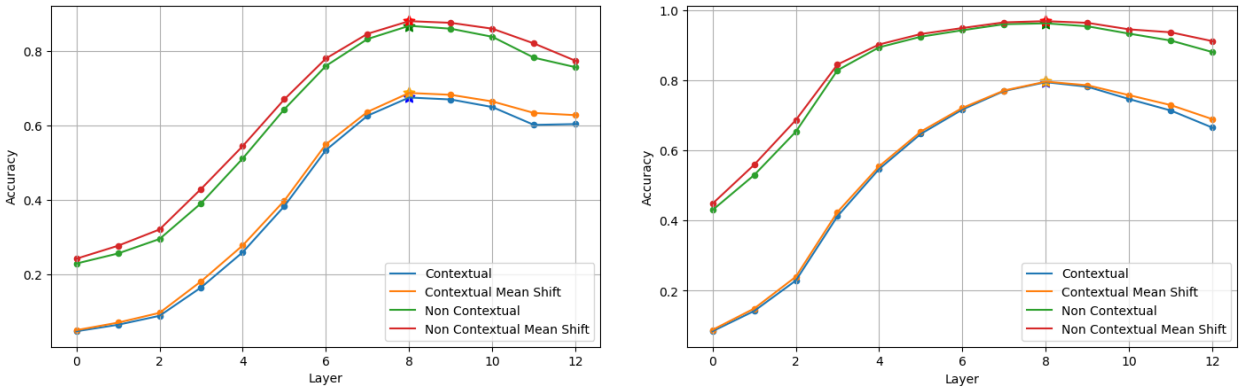
The number of unique words in this test was much lower than the number of word pairs, which means that certain words are represented many times and depend on contextual information to find the correct match. The contextual information is encoded into the word representation through the model. Therefore, we also conducted an experiment where we only allow a word pair to appear once, so if ("car", "bil") appears more than once, we removed all the pairs after the first.

Word pairs	English Words	Norwegian Words	Unique Word Pairs
23,711	2,266	2,561	2,854

Table 4.2: Word pair counts for contextual word retrieval.

4.7.3 Results Contextual Word Retrieval

Figure 4.7 shows the results from the contextual word retrieval. There is no surprise that the experiment with unique word pairs has better accuracy, shifting the graph upwards with approximately 20%. The source and target vocabulary here is limited to 2,854 pairs, while the experiment where word pairs can appear more than once has 23,711 pairs to search in. Like with the static word retrieval experiment, Notram scores better than mBERT. The difference is almost negligible when comparing with and without the mean shift. Especially at the best performing layer, layer 8, we experienced close to zero improvements in accuracy using the mean shift.



(a) mBERT **(b)** Notram

Figure 4.7: Layer-wise performance for word retrieval using contextual embeddings. The two highest lines only contain unique word pairs. In contrast, the two lower lines include all the relevant contextual embeddings from the collection of parallel sentences, even though one word can appear several times. We used one line for performance with and one without mean shift. The upper lines are with the mean shift. However, we see little to no difference in this experiment using the mean shift. The stars mark the peak-performing layer.

4.8 Language Detection using Language Embeddings

We created language-specific embeddings by taking the mean of many word embeddings Equation 4.4 in a single language and used this to demonstrate whether it was possible to improve the performance of word retrieval. In this section, we question ourselves: Can we detect the language a word belongs to using these language-specific embeddings?

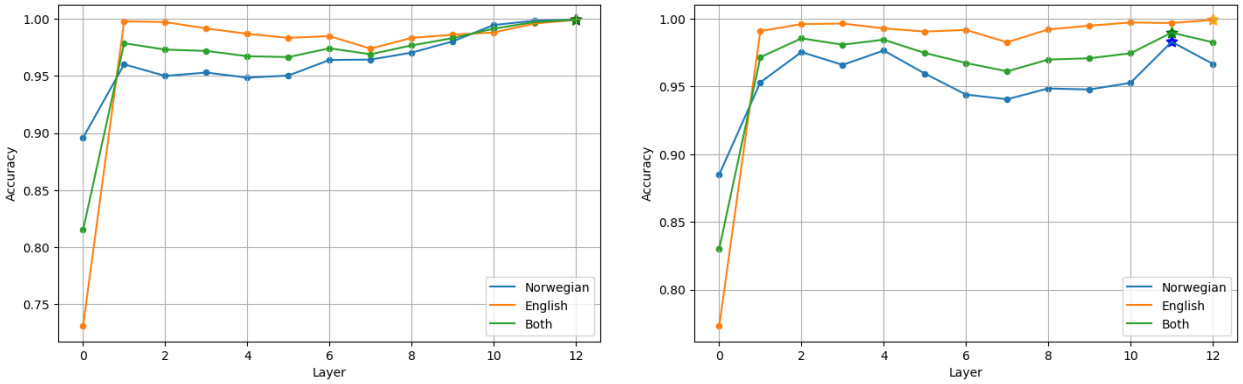
First, since the Norwegian vocabulary is more extensive than our English counterpart, we down-sampled the Norwegian vocabulary to be approximately the same size as the English. We did this by only including the words in the filtered MUSE benchmark, as the English words are. Then we split it into a test (20%) and a training (80%) part to avoid that the word embedding that we evaluated would be part of the mean language embedding. For every word in the test part belonging to English and Norwegian, we classified the language by ranking the cosine similarity of word embedding to English language embedding and Norwegian language embedding.

$$y_{i-l} = \begin{cases} 1, & \text{if} \quad sim(s_{i-l}, L_{en}) > sim(s_{i-en}, L_{no}) \quad and \quad l = en \\ 1, & \text{elif} \quad sim(s_{i-l}, L_{en}) < sim(s_{i-l}, L_{no}) \quad and \quad l = no \\ 0, & \text{otherwise} \end{cases} \tag{4.6}$$

Language detection accuracy was measured as the mean of the classification accuracy of the English words and the Norwegian words.

$$\textit{accuracy language detection} = \frac{1}{2T}\sum_{i=1}^{T} y_{i-en} + \frac{1}{2T}\sum_{i=1}^{T} y_{i-no} \tag{4.7}$$

4.8.1 Results Language Detection



(a) mBERT **(b)** Notram

Figure 4.8: Layer-wise language detection performance. The orange (best performing line from layer 1-) describes the prediction accuracy for the English vocabulary, and the blue line (lowest from layer 1-) describes the prediction accuracy for the Norwegian language. In contrast, the last green line describes the combined prediction accuracy of language detection—the stars mark in which layer the performance peak.

In Figure 4.8 we report our result from the non-parametric language detection between English and Norwegian. Both Notram and mBERT seem to be highly capable of classifying a word into its correct language with an accuracy more than 95% already after layer 2.

This result indicates that layer-specific information is encoded into the word representation from an early stage. Even the input layer scores high on language detection, way above the baseline of 50%.

4.8.2 English *Noise* in Norwegian Text Data

At layer 12 we were somewhat surprised that mBERT performed better than Notram in classifying the correct language of a word since Notram performed better at word retrieval, also with the mean shift. We noticed doing qualitative testing that there seemed to be traces of English in the Norwegian vocabulary. It is a fact that Norwegian borrows many words from English. However, sometimes we also borrow small phrases like English movie titles. Therefore, we in Table 4.3 listed a small sample of the words that were detected as English and not Norwegian from the Norwegian word embedding vocabulary.

Word	Sim Diff
lives	0.11
straight	0.08
hype	0.01
libre	0.05
lux	0.08
meles	0.02
halle	0.02
bowers	0.01
loves	0.07
lady	0.06
makes	0.18
stable	0.0

Table 4.3: Words from the Norwegian vocabulary are predicted to be English. The Sim Diff column quantifies the difference in similarity between English language embedding and Norwegian language embedding. For example, “makes” is more similar to the English language embeddings, while “stable” is almost precisely as similar to the English language embedding as the Norwegian language embedding.

For any English speaker, it is clear that most of the words also exist in the English

vocabulary. As a Norwegian and English speaker, some words seem more likely to occur in English texts since they are more frequent in this language than in the Norwegian language. In other words, it might not be good with 100% accuracy in the language detection because some of the data might have been in English. We mentioned earlier that we removed context entries detected as English using Spacy language detection. However, if English is only part of a sentence, this detection might not have been fully effective in removing all English phrases.

To provide a more in-depth qualitative analysis of the language detection, we found a few example sentences from the Norwegian News Corpus where English and Norwegian are mixed into the same sentence. In Table 4.4 we have listed three of these sentences and show what language Notram and mBERT detect each word.

From the qualitative analysis in Table 4.4 we found that Notram appears to be best at identifying Norwegian from English in the sentences, especially when we look at the last sentence. In the last sentence, mBERT classifies non of the English words as English. Because we have only provided three example sentences of contextual language detection, we can not conclude that Notram always performs better than mBERT. However, we argue that the results indicate that there is more to the results in Figure 4.8. A few more examples can be found in Appendix B.

4.9 Similarity Analysis from Monolingual and Multilingual Collections

To discover whether a distribution of word representations is uniformly distributed in latent space, Ethayarajh (2019) considered the mean cosine similarity of several random word pairs. He found that subsequent layers, the effect peaking at layer 11, were nowhere near being uniformly distributed. The average cosine similarity was roughly 0.6 in this layer. In

Word	Not	mB
Jeg	no	no
husker	no	no
at	no	no
jeg	no	no
spurte	no	no
“	**en**	**no**
who	en	en
are	en	en
you	en	en
afraid	en	en
of	en	en
”	en	en
,	no	no
forklarte	no	no
naboens	no	no
kone	no	no
.	no	no

Word	Not	mB
“	en	en
How	en	en
you	en	en
gon	en	en
na	en	en
drink	en	en
,	en	en
when	en	en
the	en	en
well	en	en
goes	en	en
dry	en	en
”	en	en
,	**no**	**en**
synger	**no**	**en**
hun	**no**	**en**
.	**no**	**en**

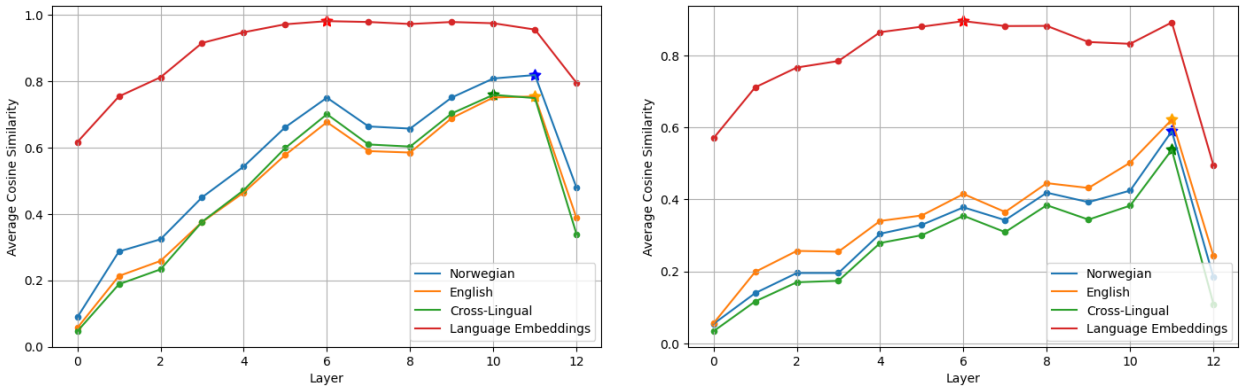
Word	Not	mB
Det	no	no
er	no	no
som	no	no
å	no	no
være	no	no
tilbake	no	no
på	no	no
Rødsberg-	no	no
diskoteket	no	no
i	no	no
1985	no	no
,	no	no
jeg	no	no
gikk	no	no
på	no	no
Strupe	no	no
-	no	no
make	**en**	**no**
no	**en**	**no**
mistake	**en**	**no**
!	**no**	**en**

Table 4.4: Language detection with language embeddings from Notram and mBERT layer 12. Not stands for Notram and mB for mBERT. Each word is detected with a language. In cases where the two models classify differently, the language code is marked in bold.

other words, the word representations only occupy a narrow cone of the latent space.

We were interested in discovering how this effect is when comparing the inter and intra similarities within and between English and Norwegian. Additionally, we compared the mBERT and Notram, to determine whether the amount of data affects the distribution of word representations from two different languages. To obtain an average similarity measure, we used 1,000 random word pairs drawn from the same collections as in section 4.8.

The resulting inter and intra similarity is shown in Figure 4.9. In addition to the inter and intra similarity, we also found the cosine similarity between the two respective



(a) mBERT **(b)** Notram

Figure 4.9: Average cosine similarity between random collection of words within each language. The top red line is the cosine similarity between the two language embeddings for English and Norwegian. In contrast, the three lower lines are average cosine similarities for the 1,000 random word embedding pairs.

language embeddings for English and Norwegian, marked with the top red line. The similarity between the two language embeddings is very high in the middle layers with close to 100% similarity in mBERT and 90% similarity in Notram, but it drops considerably to around 80% in mBERT and 50% in Notram in the last layer.

Another interesting result from Figure 4.9 is the difference in magnitude of average cosine similarity between mBERT and Notram. Generally, mBERT has a higher inter and intra similarity in all the layers except the input with a difference of around 20% in the layers where the difference is most significant. This result is suggesting that Notram occupies a bigger cone than mBERT in the latent space. This effect can be due to a set of reasons where we are not certain which affects the results most. The first is that mBERT is trained on text data from many languages, and hence it might find words within the same or two similar languages to be more similar. On the other hand, Notram is trained mainly on Norwegian and English text data (or other very similar languages) and, therefore can expand the differences between these two languages more. Another reason might be the amount of data the model is trained. Notram is initialized on mBERT, and then continued

trained on more data. We could hypothesize whether training on more data increases the variability of the word representations.

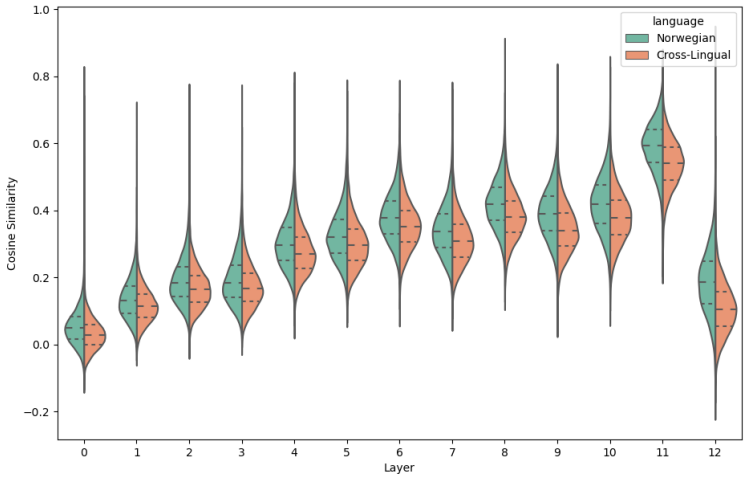


Figure 4.10: Cosine similarity distributions from the Notram model. The right side marked with green color is the distribution of intra cosine similarity for Norwegian word embedding pairs. We have taken 1,000 random pairs. The right and orange side is the inter cosine similarity for word embedding pairs where one random word is taken from Norwegian and one random word is taken from English. The dashed lines represent the 25%, 50%, and 75% percentile.

Comparing the average of cosine similarities does not give any information about the distribution of the similarities. To show how the distributions for cosine similarities look, we compare in Figure 4.10 a set of violin plots with the cosine similarity distribution for the intra Norwegian language and the inter Norwegian-English(cross-lingual).

Although the difference in the distribution between Norwegian and English (Cross-lingual) versus Norwegian alone is not considerably different Figure 4.10, it is enough to distinguish the two languages, especially in the last layers. We illustrate this again with a t-SNE plot in Figure 4.11 with 500 randomly chosen words from each respective vocabulary. The two languages separate into two clusters.

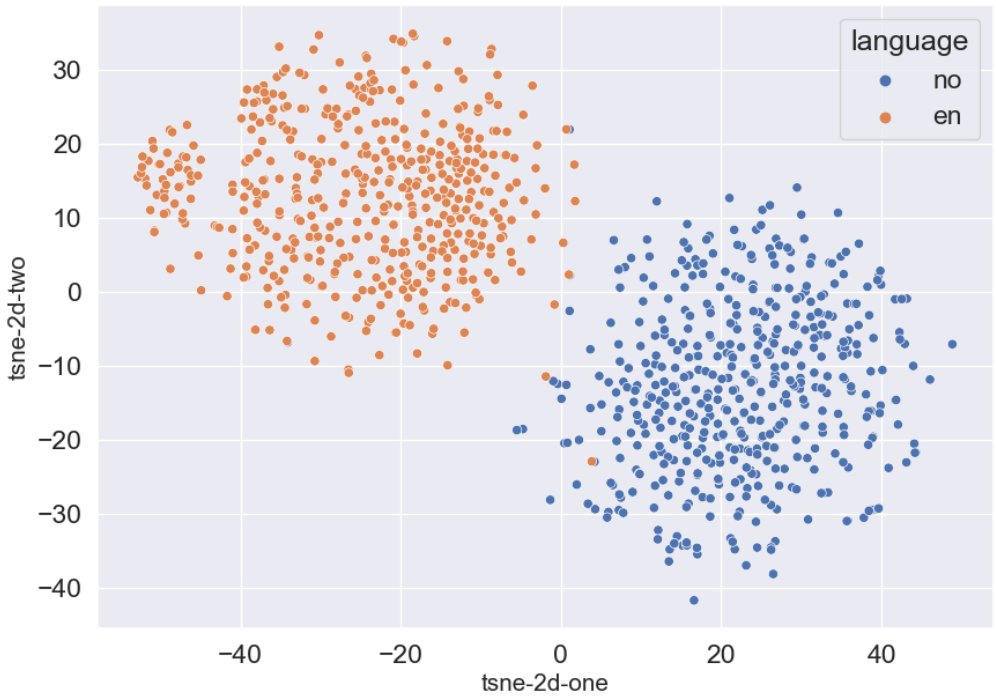


Figure 4.11: 500 random words from Norwegian and English vocabulary, respectively. Visualized with 2D plot from t-SNE dimension reduction. Embeddings from Notram layer 12. Orange points in the upper left corner correspond to English embeddings, and blue points in the lower right region correspond to Norwegian embeddings.

4.10 Discussion of Multilingual Representation Analysis

This exploratory analysis shows that BERT-based architectures such as mBERT and Notram tend to align semantics between English and Norwegian because they perform well on word retrieval tasks. We found that Notram outperforms mBERT in every word retrieval task. This difference is not so surprising, given that Notram is trained on much more data, especially for the Norwegian language. Previous work also demonstrates that Notram is better than mBERT at solving classification tasks for the Norwegian language, such as named entity recognition and POS-tagging (Kummervold et al., 2021).

For our word retrieval tasks, both static and contextual, the middle layers around layers 7 and 8 performed best. Previous evaluation of word embeddings on semantic similarity

benchmarks also indicates that the middle layers are more sensitive to semantics (Chronis & Erk, 2020). Unfortunately, good benchmarks for minor languages like Norwegian do not always exist. Therefore, we believe it helps to be able to evaluate a model's semantics based on alternative ways, such as cross-lingual word retrieval. This test does require the model to have a multilingual understanding.

The fact that English and Norwegian can be accurately aligned without any supervision, only relying on monolingual corpora, is exciting when considering the power of transferred knowledge form one language to another. Training datasets for minor languages can be sparse. However, it might be possible to use an English dataset for Norwegian classification tasks because if the important aspect in the classification set is semantics, we already know that English and Norwegian share this semantic information in encoding. Another option is combining both English and Norwegian data to fine-tune a specific task, which could improve a task since more data often helps achieve better results.

We also believe that multilingual alignment can help improve machine translation. Finding an appropriate translation is often more complicated than directly translating a word with a dictionary lookup. Word senses can often be nuanced and not always shared across languages. With our word retrieval, it is possible to obtain a list of suggestions for the target language that is not only based on direct translation but considers context.

Even though mBERT performs close to perfect at language detection at layer 12 Figure 4.8, we are skeptical of the results and still believe that Notram might performs better at detecting the correct language. When looking deeper into the problem, we started to suspect that there was some English noise in the Norwegian vocabulary of word embeddings, which might imply that a 100% accuracy is incorrect. For example, the misclassified words from the Norwegian vocabulary Table 4.3 seemed more likely to appear in English text than Norwegian text.

We created a more qualitative experiment because we suspected English noise in the Norwegian vocabulary Table 4.4. We found mixed Norwegian and English text sentences

to evaluate language detection performance. In this experiment, Notram showed a better ability to separate English and Norwegian words from the sentence.

Although using BERT for language detection is not the most computationally effective method, given that the model has 178 million parameters (Abdaoui et al., 2020), it is interesting at what level it can distinguish two languages. This method could, for example, be very useful in analysis where one is interested in investigating how the English words are being used in the Norwegian language.

Chapter 5

Contextual Property Analysis

Firth (1935) argued that:

> "The complete meaning of a word is always contextual, and no study of meaning apart from a complete context can be taken seriously"

This chapter will examine how this applies to word representations in a contextual language model. We created a word representation in either the wrong context or without context. What happens to a word representation if it does not fit into the context of the surrounding words? How is a word represented if only the word is processed through the model without surrounding words? We attempted to address these questions by exploring the contextual embeddings of different words from the Norwegian vocabulary.

5.1 Related Work on Real-Word Spelling Errors

For our experiment of collecting words in the wrong context, we have taken inspiration from the literature, including real-word spelling errors. A real-word spelling error is a correctly spelled word but in the wrong context. It typically occurs because the typer either does not know how to type a word or it misses a stroke on the keyboard. Some words are often

confused, for example, because they sound similar, such as “than”-“then”, “peace”-“piece”, “here”-“hear” etc. These are all examples of real-word spelling errors that often appear in the text.

When attempting to correct a spelling error, real word, or normal non-word spelling error, the proximity in letters or phonetics often matters when suggesting alternatives. The set of suggestions for a given spelling is called a confusion set because they are likely to be confused with the word (Carlson et al., 2001; Mays et al., 1991). In our analysis, we created a confusion set for each of the words in our vocabulary to make the wrong context more realistic than substituting with a random word.

5.2 Embeddings of Words in the Wrong Context

We constructed a confusion set to find a realistic wrong context. The confusion set for a candidate word could only include words from our word embedding vocabulary. In this section, we explain the steps to how we generated a wrong context and how this affected the embedding of a word through a similarity experiment.

5.2.1 Confusion Sets

Two commonly used edit distance metrics include Levenshtein (Levenshtein et al., 1966) and Damerau Levenshtein (Damerau, 1964). In this project, we used the Damerau Levenshtein distance to calculate character edit distance. The operations for 1 character edit are:

- Deletion (“cat” -> “ca”, “t”->“”)
- Insertion (“cat” -> “cats”, “”->“s”)
- Substitution (“cat” -> “cet”, “a”->“e”)

- Transposition (“cat” -> “act”, “ca”->“ac”)

To find relevant confusion candidates for a word in the vocabulary, we compared the phonetic encoding or the letter encoding, depending on which is the closest to the relevant word, for each word in the vocabulary of 50,000 words. The maximum edit distances was set to 3 edits, and there were more than 10 candidates with less than 3 edits, we only included the 10 closest.

Candidate Word	Confusion Set
og	[“å”, “os”, “g”, “o”, “ol”, “om”, “dog”, “ag”, “òg”, “tog”]
spille	[“spillet”, “snille”, “stille”, “pille”, “skille”, “spilte”, “spill”, “spiller”, “spilles”, “speilet”]
jul	[“hjul”, “kul”, “bul”, “jus”, “jol”, “jula”, “hul”, “juli”, “gul”, “pool”]

Table 5.1: 3 words with their confusion sets.

5.2.2 Randomly Inducing Wrong Words in Sentence

We used the Norwegian Treebank[1] to obtain contextual word embeddings for words in the wrong context. The Norwegian Treebank includes a set of tokenized sentences, and it also contains Part-of-Speech tags as well as tags for named entities (Jørgensen et al., 2019). For each sentence in the corpus, we first selected a random word from the tokenized sentence within our vocabulary. Then we switched it out with one of its confusion candidates, again random selecting among the confusion candidates. Table 5.2 shows 4 example sentences, and which of the words are substituted. When we obtained our word embedding, we only considered the embedding of the substituted word.

[1] https://huggingface.co/datasets/NbAiLab/norne

Sentence	Original -> Confusion
Litt forsinket , klokken 20.25 , satte bryllupsgjestene seg til bords for å nyte gallamiddagen på **slettet** .	slottet -> slettet
I galla og utsøkte festantrekk kom de til bryllupsmiddagen , kongelige , presidenter , familiemedlemmer , **verner** og representanter for regjering , storting og viktige norske institusjoner .	venner -> verner
Jagland skal **slede** utenrikskomitéen	lede -> slede
I **manne** tilfeller gis klare signaler om politiske kursendringer i den retning Fr.p. ønsker .	mange -> manne

Table 5.2: Examples of sentences with substitution. The word in bold is the word we induced in the sentence. In the right column, we state what was the original word in the sentence and what the new induced word is. We used the confusion set to find the induced word.

5.2.3 Possible Outcomes from Word Embedding Comparing

When we compared a contextual word embedding from a word in the wrong context, there were three possible outcomes for the closest match. The three possible outcomes were: the contextual word embedding was most similar to the static embedding of the word itself, most similar to the static embedding of the original word, or most similar to a static embedding of another word.

In Figure 5.1 we drew an imaginary scenario of how the embeddings of “desert” and “dessert” can look when comparing contextual and static embeddings. Every blue point represents static word embeddings, so in this example, we have a static word embedding for “dessert,” “candy” and “camel.” The pink dot represents the contextual embeddings of the word “desert”. However, in this scenario, the word is in the wrong context because it is confused with “dessert.” Therefore, in this case, “desert” is the word *itself*, “dessert” is the *original* word, and “candy” is some *other* word. With the proximity in two-dimensional latent

Figure 5.1: Illustration of imaginary scenario of static and contextual representation of the words desert and dessert in latent space. The blue points with corresponding figures represent a static word embedding, while the pink point represents a contextual word embedding from the adjacent sentence. In this case, the contextual word embedding for “desert” is most similar to the original word in the sentence “dessert.”

space, this example would predict the outcome as matching the original word “dessert.” Figure 5.2 describe the process of obtaining a match for a word embedding in the wrong context.

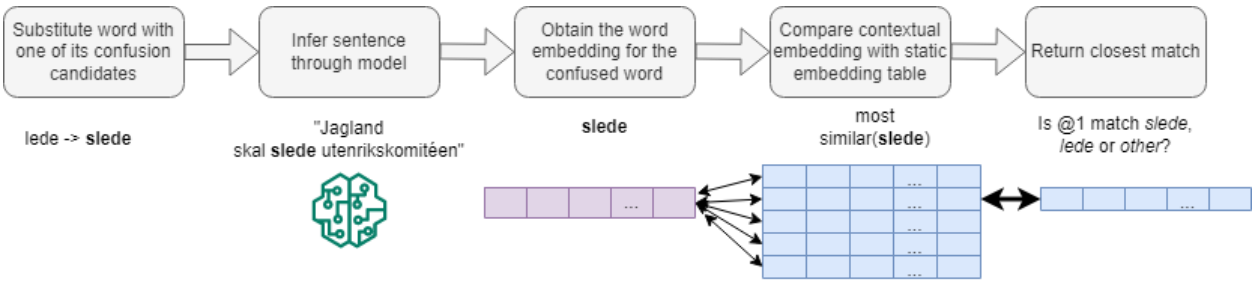
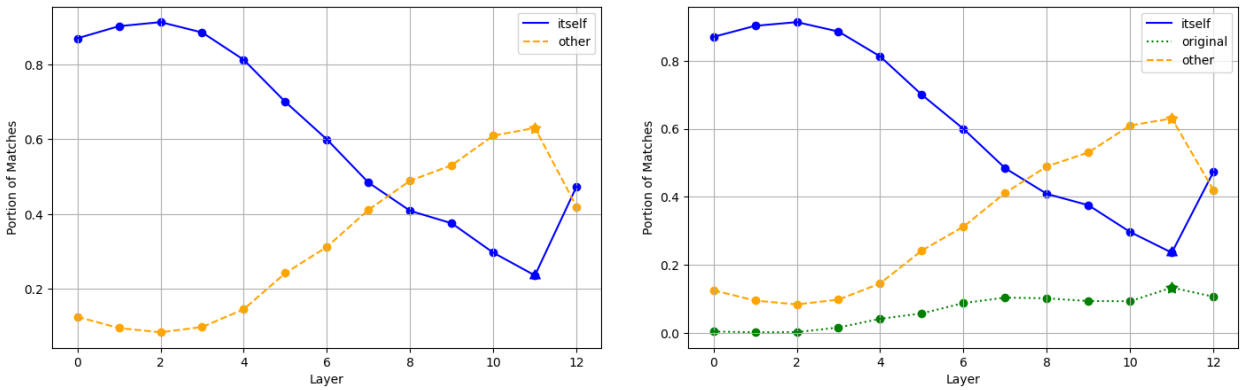


Figure 5.2: The process of obtaining a match for a word embedding in the wrong context. We use the example sentence “Jagland skal **slede** utenrikskomitéen” where “lede” is substituted with “slede”.

5.2.4 Results Word Embedding Comparing in Wrong Context

In Figure 5.3 we report the results from the analysis of the contextual embeddings of words in the wrong context. We have a baseline with evaluated words from real context to determine how big a portion deviates from its static word embedding. Here, the highest deviation can be found in the middle layers, with around 15% not matching itself, where layer 7 deviates the most.



(a) Baseline **(b)** Wrong context

Figure 5.3: Layer-wise match from contextual embedding to a static standard. The baseline shows how many words in the testing set match itself versus matching another word. The lower image with the wrong context shows the same, but this time we only checked the match for words induced into the wrong context. This difference makes it possible to add a green line (the lowest line), which counts how many times the word embedding matches the original word. A triangle marks the lowest-performing layer for both images, and a star marks the best performing layer. In the baseline, the highest deviation for matching itself is in the 7th layer, while in the wrong context example, it is the 11th layer.

Words in the wrong context do not match themselves very often in the subsequent layer Figure 5.3. Especially in the 11th layer, we have a significant portion of words that match a other word and a small portion that matches the original word. We tested several words qualitatively and discovered a surprising pattern. The “other” words that were the closest match did not seem so random. Although the words in the wrong context were different, many of the same words kept showing up as the closest match. This discovery

indicated that words in the wrong context did not get an arbitrary representation but rather shared similar properties. We tested which words were most frequently the closest match to investigate further. We used the 11th layer as the output embeddings because this was the layer where the least words matched themselves.

Word	Count
sogar	1569
ar	941
ala	426
poden	349
ev	253
nave	197
dett	166
prominent	144
ike	144
versus	143

Table 5.3: The top 10 most frequent matches for words in the wrong context where we have tested 15,467 words closest match to a vocabulary of 50,000 words. From 15,467 examples, 1,569 words all rank “sogar” as the closest match. Since “sogar” is not a common word in the Norwegian language, we found this strange.

In Table 5.3 we have listed the top 10 most frequent matches for words in the wrong context. For example, more than 10% of the tested words match the word “sogar” closest. 28% of the closest word matches are the same 10 words. Since the whole vocabulary comprises 50,000 word embeddings, matching the same 10 words in 28% of the scenarios is not random. It rather indicates that the embeddings in the wrong context share some property. Figure 5.4 illustrates how we believe embeddings in the wrong context cluster differently than embeddings in from real context. The yellow circles represent word embeddings from the real context, and the grey circles represent word embeddings from the wrong context.

To examine these patterns in more detail, we collected 1,000 words present in the wrong and real context from our evaluation corpus, the Norwegian Treebank. We have

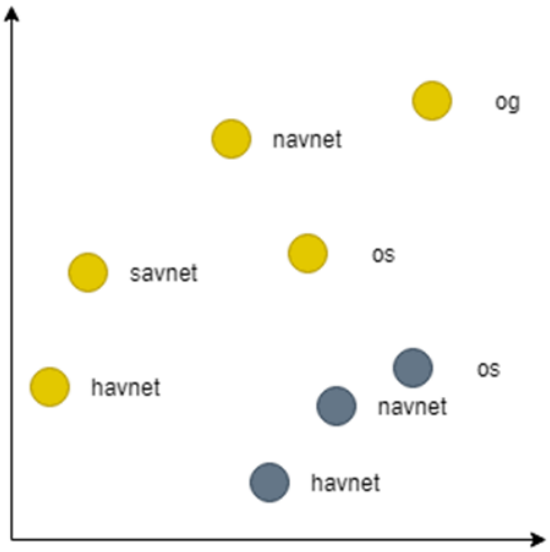


Figure 5.4: Illustration of how we hypothesized contextual embeddings from real context, marked with yellow color tending towards the upper left, and embeddings from the wrong context, marked with grey color tending towards the lower right, cluster in latent space.

one contextual word embedding from a real context and one from the wrong context for each word. In Figure 5.5 we used t-SNE to reduce the embeddings to 2 dimensions and plotted the results. The blue color corresponds to embeddings from the wrong context, while the orange corresponds to real context. Comparing this plot with our hypothesized figure in Figure 5.4, we observe that the plot does show a clustering between real context and wrong context.

5.3 Real and Wrong Versus Isolated Context

Static word embeddings can also be created without context. This kind of word embedding from BERT is often called ISO (isolated) isolated embeddings. ISO embeddings tend to perform worse than AOC embeddings on benchmarks like word similarity (Bommasani et al., 2020). How does an ISO word embedding relate to an embedding from real and an embedding from the wrong context? Can we find any similar properties?

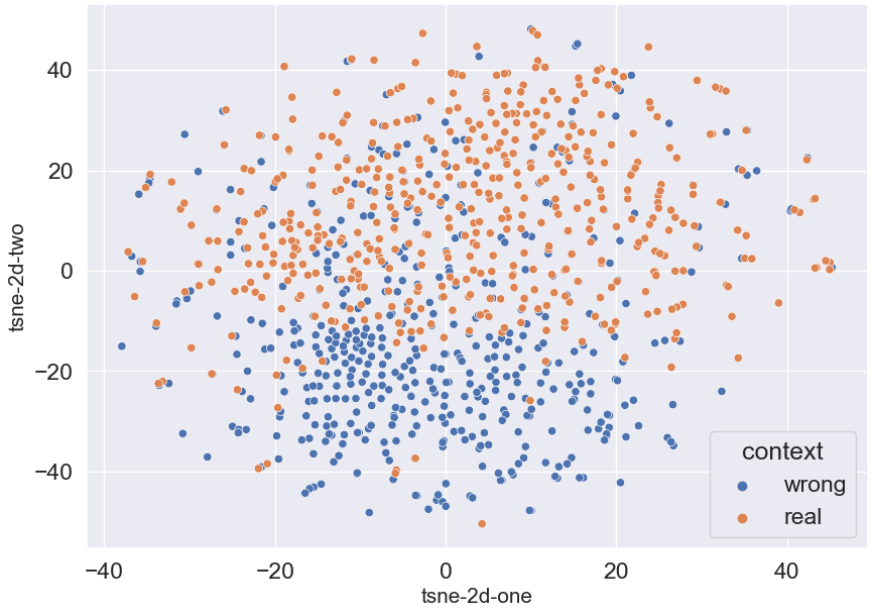
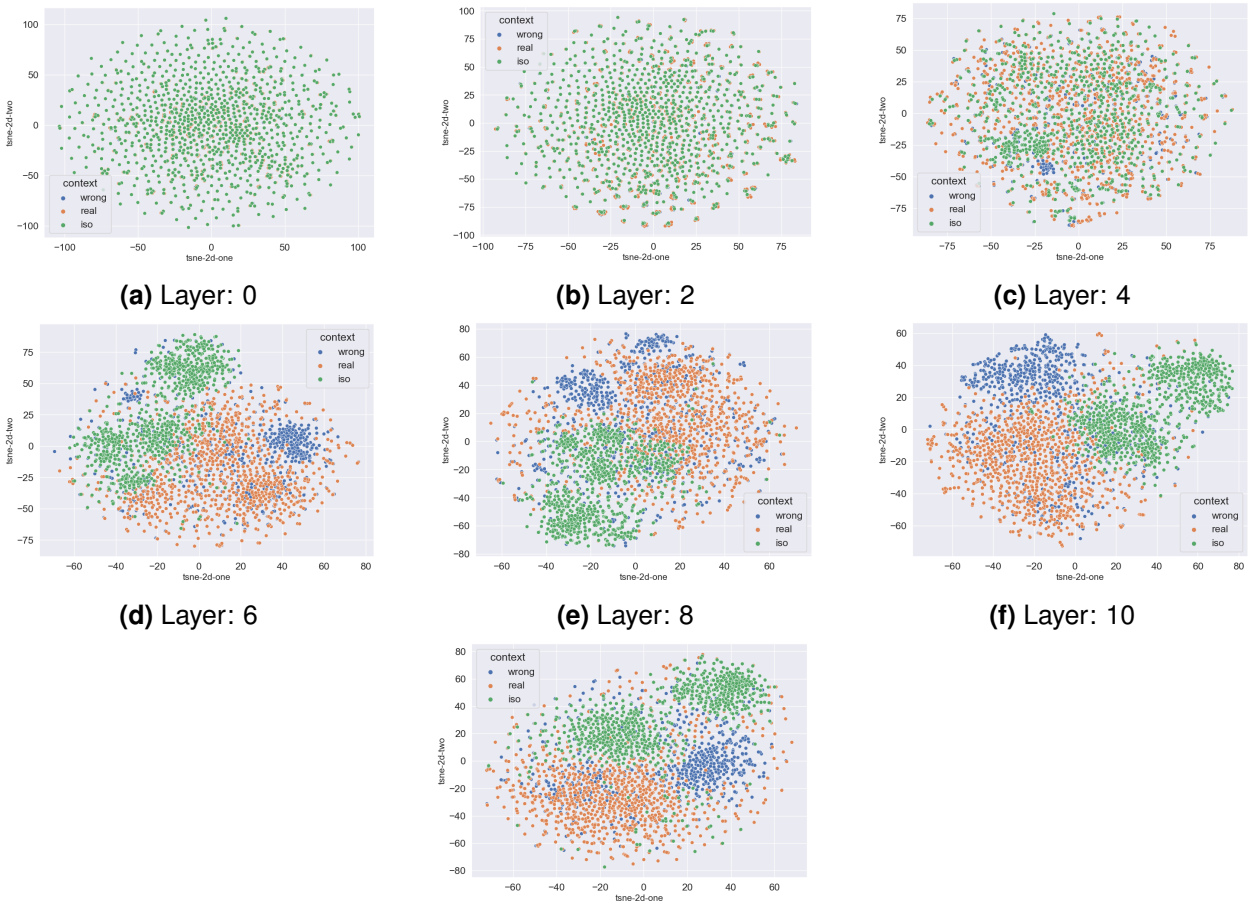


Figure 5.5: A sample of 1,000 random words from real and wrong contexts respectively reduced to 2D plot with t-SNE. The 1,000 words have both one real and one wrong context, so the word collection is the same for both types of contexts. The orange points, tending towards the upper part of the graph, are embeddings from real context, while the blue points, tending towards the lower part of the graph, are embeddings from the wrong context. The plot could indicate a separating feature between real and wrong context, but some of the blue points, so embeddings from the wrong context, are mixed with the real context cluster.

We first created a set of ISO embeddings from the same collection of words that we used in Figure 5.5. Thereby we obtained one word embedding from a real context, one from a wrong context, and one from an isolated context for the same collection of words. In this analysis, we were interested in exploring if there was any development from the first to the last layer. Figure 5.6 is a t-SNE reduced plot of the contextual embeddings from every second layer, including the first input layer and the last output layer. Interestingly, in the first layers, the embeddings overlap. However, we began to observe clustering of the embeddings from real, wrong and isolated context from the middle layers. Whether there is a more substantial relation between wrong and isolated context compared to real and isolated context is not possible to determine from this plot.



(a) Layer: 0 **(b)** Layer: 2 **(c)** Layer: 4

(d) Layer: 6 **(e)** Layer: 8 **(f)** Layer: 10

(g) Layer: 12

Figure 5.6: Scatter-plots from contextual embeddings reduced with t-SNE from real, wrong, and isolated contexts. The word collection consists of 1,000 words, all represented from the three types of context. Each word has 3 contextual embeddings in other words. We show the development from layer 0 to layer 12. The green, isolated embeddings in the input layer cover almost all the other embeddings. However, as the layers progress real, wrong, and isolated context starts to cluster. Real context is marked with orange color, wrong context with blue color, and isolated context with green color. From layer 6 we start to see a clear separation.

From the results in Figure 5.5 and Figure 5.6 we wanted to investigate whether there was a given direction from real to wrong, wrong to ISO. Finding one direction can be challenging since each vector has 768 dimensions. Therefore, we created a t-SNE reduced plot with a small collection of random words and their corresponding embedding

Figure 5.7: Scatter-plot of contextual embeddings from real, wrong and isolated context with word examples. The green color is the isolated context, the blue color is the wrong context, and the orange color is the real context. The suffix "_r", "_w" and "_i" correspond to real, wrong, and isolated, respectively. We do not see any common direction from the scatter-plot, e.g., real to wrong context or wrong to isolated context.

in real, wrong, and ISO embedding Figure 5.7. We observe no specific direction between real-wrong, real-ISO, and wrong-ISO embedding collections.

To elaborate on the results in Figure 5.7 we looked at the average cosine similarity between real and ISO, real and wrong, and wrong and ISO embeddings for the same word. The result is stated in Figure 5.8. We also tested the intra and average cosine similarity between random word pairs within the real, wrong, and ISO collections. Figure 5.8 states the resulting intra average cosine similarity for the random word pairs. ISO embeddings score highest on this test with a peaking similarity in layer 11 with more than 80% average similarity between the ISO embeddings, showing that words without context become similar in the model. In this example, we must also note that the ISO embeddings all share the same position (all the words are placed as the first word in a sentence, and hence

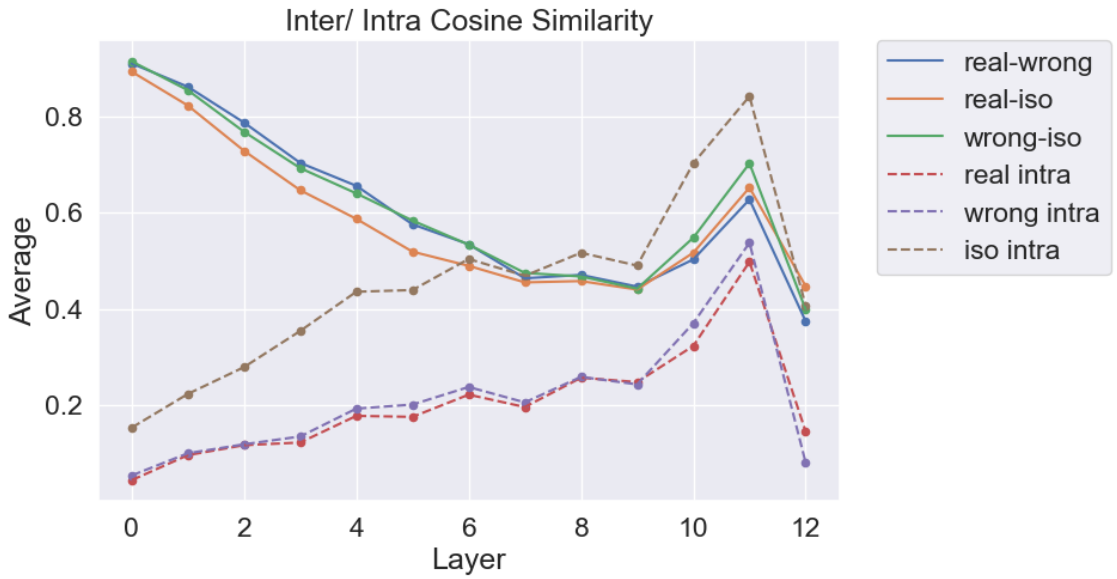
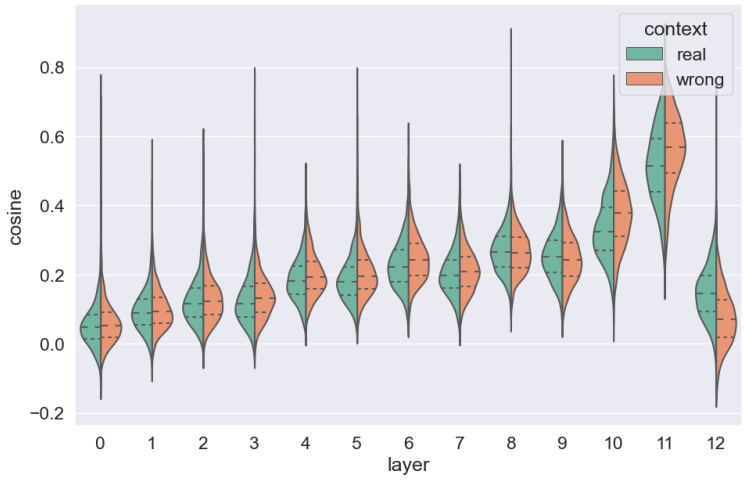


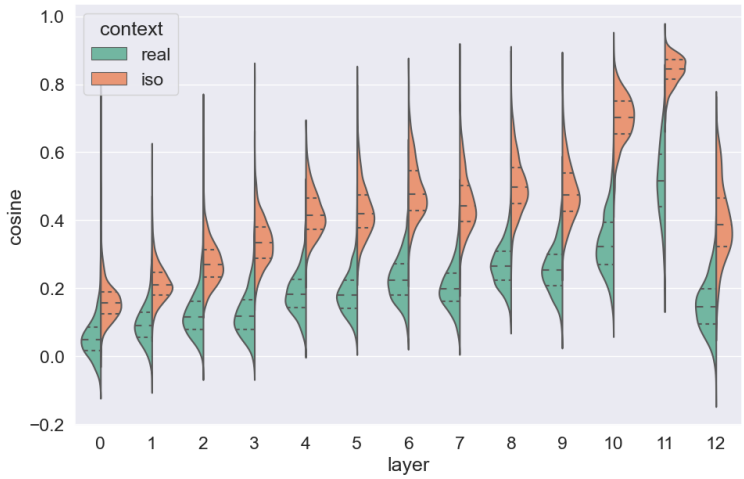
Figure 5.8: The average cosine similarity is measured from a collection of random word pairs. The solid lines represent the inter similarities, while the dashed lines represent the intra similarity. The solid lines compare the same word but from a different context type, while the dashed lines compare two random words within the same context type. The highest dashed line is intra similarity for isolated embeddings, which is much denser in representation in the later layers than the other intra similarities. This distribution is most likely because later layers are more contextual, and isolated embeddings do not have context.

given the same positional embedding in the input layer if they have the same number of wordpieces), making them more similar. However, the ISO embedding similarity is likely not due to the positional embedding because from Figure 5.6 we see that in the input layer the embeddings from real, wrong and isolated context are overlapping.

In Figure 5.9 we constructed distribution plots for the intra similarities. The green color (the left violin side) corresponds to real context in both plots, whereas the orange (the right violin side) distributions are wrong and isolated context, respectively. One thing to notice is the change between the 11th and 12th layers. In the 11th layer, the wrong context distribution shows higher cosine similarity than the real context, while in layer 12 the cosine similarity for the wrong context is suddenly lower. This shift does not happen in an isolated context, where the distribution of cosine similarity is always higher. One



(a) Real and wrong contextual embeddings.



(b) Real and isolated contextual embeddings.

Figure 5.9: The violin plots show the distribution of cosine similarities between randomly chosen word pairs within real, wrong, and isolated embeddings collections. The left green distribution is from the real contextual embedding collection in the upper and lower plot. The right orange distribution is from the wrong contextual embedding collection in the upper plot. The right orange distributions are from the isolated contextual embedding collection in the lower plot. Especially in layer 11, the difference in distribution between real and isolated embeddings is big, both in average value and the density of the distribution.

interesting result for the ISO embeddings is their spreading, however. Especially in layer 11, the isolated embeddings are more densely represented than embeddings from real context, which peak at spreading cosine similarity in this layer. We suspect that layer 11 is

highly contextual, and the embedding here is not much similar to itself but rather to the surroundings.

5.4 Discussion of Contextual Property Analysis

Firth argued that the complete meaning of a word is always contextual (Firth, 1935). According to our analysis, the word representations from BERT also need context to be able to represent the word's full characteristics because ISO embeddings are much denser in representation than embeddings from real context Figure 5.9.

Not all the words in the wrong context deviate from words in a real context. This deviation is to be expected given how we created the wrong context. Some words in a confusion set might fit into the same context, so even though we confused two words, it is still an option that the new infusion in the sentence fits. We suspect that this happened to some of the words and made the difference between real and wrong context less clear. Additionally, it has been pointed out that certain words, like stopwords, tend to be more contextual than e.g., nouns (Chronis & Erk, 2020). This effect may also have affected the results. We analyzed how the context of a single word affects that word. However, the word we infuse may affect the rest of the context, as it becomes the context for the rest of the words in the sentence.

Ambiguous words are likely to affect the results of our analysis. We mentioned in our chapter 2 Background and Related Work that one shortcoming when using static word embeddings is conflated meanings in the same word representation. Since we aggregated contextual embeddings over several contexts, it is expected that our static word embedding vocabulary also has conflated meanings. We believe that this affects the results, especially when looking at the baseline in Figure 5.3a. In the middle layers, the deviation from itself is around 15-20%. This deviation might be due to ambiguous words, where one contextual word embedding representing one sense may not match the static embedding in our

vocabulary.

ISO embeddings have a lot higher intra similarity than embeddings from real context and embeddings from the wrong context Figure 5.9b. One reason is may be that they share the same positional embedding. However, we also hypothesize that without context, they converge more to the same embedding because they do not have context to confirm all the normal properties. For example, without context how should the model encode an ambiguous word, which sense properties should it use then? We again point to the fact that AOC word embeddings from BERT have performed much better on similarity benchmarks than those created in isolated context (Bommasani et al., 2020; C.-L. Liu et al., 2020).

Chapter 6

Discussion

In the previous chapters, we have discussed the two experiments separately. Now we wish to compare and discuss the relevant parts of both experiments.

6.1 Semantic Property Most Apparent in Middle Layers

According to our cross-lingual word retrieval results, semantics appeared most present in the middle layers around layer 7 and 8 as noticed in Figure 4.4. This is in agreement with the results in previous work (Chronis & Erk, 2020), which also found the middle layers to perform best on English benchmarks for semantic similarity. Of course, it does not mean those semantics are not present in, e.g., later layers, but we believe that the semantics property is most apparent in the middle layers and that other properties are more apparent in other layers.

ISO embeddings evaluated on semantic benchmarks have been proven to perform worse than AOC embeddings (Bommasani et al., 2020; C.-L. Liu et al., 2020). In our distributional analysis of embeddings from isolated context, embeddings from the wrong context, and embeddings from real context, embeddings from isolated context are more densely distributed than embeddings with context Figure 5.9. Such results imply that they

do not capture all the characteristics they need.

6.2 Ambiguous Words

Ambiguous words are words with multiple senses. The relevant sense is dependent on the word's context. To best be able to represent a word, we believe one representation for each word is the best option. In our experiments, on the other hand, we have used a static vocabulary of word embeddings with only one representation per word. A static word embedding vocabulary keeps the problem of conflated meanings which we discussed in the Background chapter 2 *Background and Related Work*. This section discusses the influence ambiguous words may have had in our experiments.

The performance in the @1 match in the static word retrieval task achieves much lower accuracy with the highest score of around 50% Figure 4.4 compared to the contextual word retrieval task with a @1 accuracy at 80% Figure 4.7. Although these experiments differ in many ways (e.g., the number of target words to search in approx. 23,000 against 50,000, and that one experiment only has one embedding for a word while the other allows multiple) we suspect that the somewhat "poor" performance in the static word retrieval task may partially be due to ambiguous words. We believe this because the static embeddings are created from different corpora, where the senses of a word can be different depending on the context. In addition, since words can be highly ambiguous with many different nuanced senses, a benchmark such as MUSE with only direct non-contextual translations may not be able to capture the right relationships.

We also argue that our qualitative results from the cross-lingual sense disambiguation show an example of the effect of a conflated meaning representation. In the English and Norwegian sentences, including "love" Table 4.1, non of the two examples lists "love" as the highest match in the Norwegian vocabulary. In the example with "do" on the other hand, the word is matched highest to itself in the Norwegian vocabulary. Therefore, we suspect

that the static word embedding of “love” is a representation of the conflated meaning of both “to love” and “to promise,” each representing an English and Norwegian sense of the word. When we looked for the closest word embeddings of a contextual example, we got the suggestion “garantere” which is semantically similar to the contextual embedding of “love”. However, this word does not have its meaning conflated. Additional arguments suggest this effect is that the second and third match for “love” is two inflections of the word.

The baseline in Figure 5.3a show how words from real context deviate most from themselves in the middle layers. We believe ambiguous words are contributing to this deviation. For example, the Norwegian example “love” would have resulted in an “other word” match in the cross-sense disambiguation analysis. This match would have contributed to the deviation in the baseline.

A small portion of the closest match of words from the wrong context matches the original word that stood in the sentence Figure 5.3b. We hypothesize that if a spelling error is prevalent, it would also appear often in the actual data that a model is trained. Possibly the model might make a sense representation of spelling mistakes. One example is the word “vert” (host), which is often confused with “vært” (been) because they sound just the same. “Vært” is much more frequent, so it happens often that this word would be misspelled to “vert” but not so other the other way around If our hypothesis is true, then the word “vert” might have a sense representation that is closer to the word “vært” (been) than the real word “vert” (host). We emphasize that this is only a hypothesis, but it could be part of the explanation as to why the model can match the original word in approximately 15% of the time Figure 5.3b (layer 11).

6.3 Subsequent Layers are Highly Contextual

We have investigated the cosine similarity between many random word pairs from the same collection and different collections. All the distributional plots of cosine similarities Figure 4.10 Figure 5.9 show that the word representations are not uniformly distributed. These distributions agree with the findings in previous work (Ethayarajh, 2019). If one takes a random word embedding and compares it to many other random word embeddings from any collection, most of the embeddings would be similar in many dimensions. From the distributions, we also saw that most of them have a normal distribution because most random word pairs are similar. However, a small portion of the word embeddings was more similar, and a small portion was less similar.

The standard deviation increased in the 10th and 11th layer for the contextual embeddings in the real and the wrong context. We could see this because the distributional graphs become more “stretched” in these layers Figure 5.9. However, when we compared with the distributions from the static embeddings, we did not experience the same increase in standard deviation Figure 4.10. We add that the isolated embedding has a minimal standard deviation in layer 11 and is very similar in this layer. We argue that these layers show much more contextual information. When we took the average over several contexts in our AOC embeddings, we automatically removed contextual information and made the distribution denser for these layers.

6.4 An Embedding of Different Characteristics

We have studied word representations from the contextual language model BERT from different perspectives throughout this thesis. In analyzing the word representations from Norwegian and English, we found that semantics is part of the representation and proved this through a word retrieval test. In addition, we found a non-parametric method to detect

the language of a word representation, which tells us that language characteristics are also part of the vector representation. Previous work also has sound characteristics such as syntax and how words within the text relate (Tenney, Das, et al., 2019). Figure 6.1 illustrate how a contextual word embedding has much information about many characteristics in an embedding. We have also listed “other?” as there might be more exciting characteristics to discover within the embedded information.

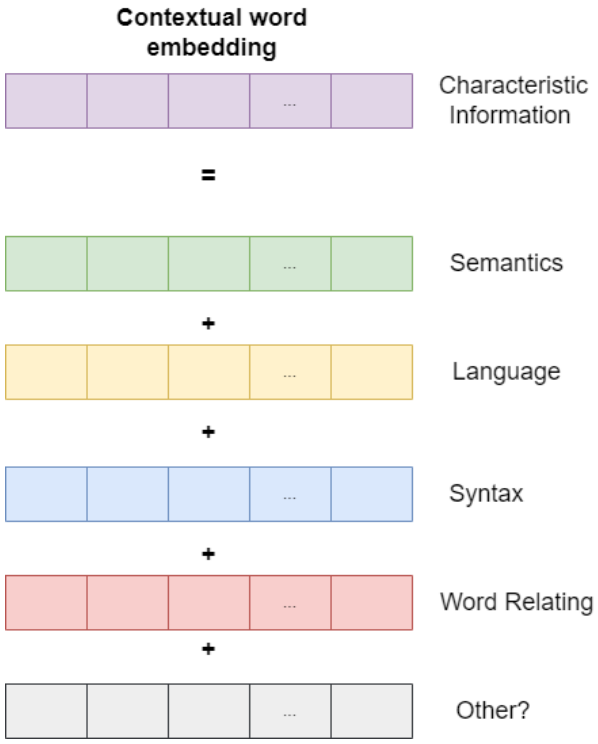


Figure 6.1: A contextual language embedding has information corresponding to several characteristics, some like language and semantics we proved in this thesis. Previous work has also proved other characteristics that are within the embeddings like syntax and word relating (Tenney, Das, et al., 2019).

Different NLP tasks can benefit from different features, but it might not be necessary to have all the features. We argue that mapping the features within a contextual embedding can help the community find where the more relevant information is. For example, if the task is to translate words, it might be more beneficial to use the middle layers as feature

extractors than the last layers. By cutting away a few layers, one can also reduce the computational processing time. The more we know about the model, the better we can benefit from it.

Chapter 7

Conclusion

Throughout this thesis, we have studied the properties of word representations from BERT. The first experiment confirmed that BERT could align semantics across English and Norwegian because we had a @1 word retrieval accuracy of more than 50%. In addition, we found that BERT can distinguish between the two languages with almost 100% accuracy through non-parametric language detection. The second experiment illustrated how the word representations clustered into groups depending on whether the context was real, wrong or isolated.

We contribute to the research community by extending the understanding of what information is encoded in the word representations. By confirming that the BERT architecture aligns semantics between English and Norwegian from a pre-trained stage, we explain why it is possible to transfer knowledge from English to Norwegian without parallel data. We also show that language property is encoded into the word representations and that this information can be used to perform language detection on each word in a text. By discovering that a word in a wrong context shares a similar property with other words in a wrong context, we explain why a BERT model can find a word error in a sentence, even though the word error was not part of a labeled training dataset. We also show that context is necessary for a word representation to gain all encoded properties of a word because

embeddings from isolated contexts are very similar in the last layers.

We hope that our analysis in the future motivates more use of multilingual models and new ways of using the encoded information, such as real-word error detection.

Bibliography

Abdaoui, A., Pradel, C., & Sigel, G. (2020). Load What You Need: Smaller Versions of Multilingual BERT. *arXiv preprint arXiv:2010.05609*.

Agirre, E., Alfonseca, E., Hall, K., Kravalova, J., Pasca, M., & Soroa, A. (2009). A study on similarity and relatedness using distributional and wordnet-based approaches.

Bommasani, R., Davis, K., & Cardie, C. (2020). Interpreting Pretrained Contextualized Representations via Reductions to Static Embeddings. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 4758–4781.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., . . . Amodei, D. (2020). Language Models are Few-Shot Learners. *arXiv preprint arXiv:2005.14165*.

Camacho-Collados, J., & Pilehvar, M. T. (2018). From word to sense embeddings: A survey on vector representations of meaning. *Journal of Artificial Intelligence Research*, *63*, 743–788.

Cao, S., Kitaev, N., & Klein, D. (2020). Multilingual alignment of contextual word representations. *arXiv preprint arXiv:2002.03518*.

Carlson, A. J., Rosen, J., & Roth, D. (2001). Scaling Up Context-Sensitive Text Correction. *IAAI*, 45–50.

Chronis, G., & Erk, K. (2020). When is a bishop not like a rook? When it's like a rabbi! Multi-prototype BERT embeddings for estimating semantic relationships. *Proceedings of the 24th Conference on Computational Natural Language Learning*, 227–244.

Conneau, A., Lample, G., Ranzato, M., Denoyer, L., & Jégou, H. (2017). Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Damerau, F. J. (1964). A technique for computer detection and correction of spelling errors. *Communications of the ACM*, *7*(3), 171–176.

der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of machine learning research*, *9*(11).

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, *1*(1950), 4171–4186.

Ethayarajh, K. (2019). How contextual are contextualized word representations? comparing the geometry of BERT, ELMo, and GPT-2 embeddings. *arXiv preprint arXiv:1909.00512*.

Firth, J. R. (1957). A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*.

Firth, J. R. (1935). The Technique of Semantics. *Transactions of the philological society*, *34*(1), 36–73.

Francis, W. N., & Kucera, H. (1979). Brown corpus manual. *Letters to the Editor*, *5*(2), 7.

Galassi, A., Lippi, M., & Torroni, P. (2020). Attention in Natural Language Processing. *IEEE Transactions on Neural Networks and Learning Systems*, 1–18. https://doi.org/10.1109/tnnls.2020.3019893

Gale, W. A., Church, K., & Yarowsky, D. (1992). Estimating upper and lower bounds on the performance of word-sense disambiguation programs. *30th Annual Meeting of the Association for Computational Linguistics*, 249–256.

Gehring, J., Auli, M., Grangier, D., Yarats, D., & Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. *International Conference on Machine Learning*, 1243–1252.

Gerz, D., Vulić, I., Hill, F., Reichart, R., & Korhonen, A. (2016). Simverb-3500: A large-scale evaluation set of verb similarity. *arXiv preprint arXiv:1608.00869*.

Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). *Deep learning* (Vol. 1). MIT press Cambridge.

Hill, F., Reichart, R., & Korhonen, A. (2015). Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, *41*(4), 665–695.

Jørgensen, F., Aasmoe, T., Husevåg, A.-S. R., Øvrelid, L., & Velldal, E. (2019). NorNE: Annotating named entities for Norwegian. *arXiv preprint arXiv:1911.12146*.

Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2017). Bag of tricks for efficient text classification. *15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017 - Proceedings of Conference*, *2*. https://doi.org/10.18653/v1/e17-2068

Kim, Y., Jernite, Y., Sontag, D., & Rush, A. (2016). Character-aware neural language models. *Proceedings of the AAAI conference on artificial intelligence*, *30*(1).

Kummervold, P. E., la Rosa, J., Wetjen, F., & Brygfjeld, S. A. (2021). Operationalizing a national digital library: The case for a norwegian transformer model. *arXiv preprint arXiv:2104.09617*.

Lee, J.-H., Kim, M., & Kwon, H.-C. (2020). Deep Learning-Based Context-Sensitive Spelling Typing Error Correction. *IEEE Access*, *8*, 152565–152578.

Levenshtein, V. I. et al. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics doklady*, *10*(8), 707–710.

Litschko, R., Vulić, I., Ponzetto, S. P., & Glavaš, G. (2021). Evaluating multilingual text encoders for unsupervised cross-lingual retrieval. *European Conference on Information Retrieval*, 342–358.

Liu, C.-L., Hsu, T.-Y., Chuang, Y.-S., & Lee, H.-Y. (2020). A study of cross-lingual ability and language-specific information in multilingual BERT. *arXiv preprint arXiv:2004.09205*.

Liu, N. F., Gardner, M., Belinkov, Y., Peters, M. E., & Smith, N. A. (2019). Linguistic knowledge and transferability of contextual representations. *arXiv preprint arXiv:1903.08855*.

Loureiro, D., Rezaee, K., Pilehvar, M. T., & Camacho-Collados, J. (2021). Analysis and Evaluation of Language Models for Word Sense Disambiguation. *Computational Linguistics*, 1–55.

Mallery, J. C. (1988). Thinking about foreign policy: Finding an appropriate role for artificially intelligent computers. *Master's thesis, MIT Political Science Department*.

Mays, E., Damerau, F. J., & Mercer, R. L. (1991). Context based spelling correction. *Information Processing \& Management*, *27*(5), 517–522.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*, *ICLR 2013*, 1–12. https://arxiv.org/abs/1301.3781

Mikolov, T., Le, Q. V., & Sutskever, I. (2013). Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Navigli, R. (2009). Word sense disambiguation: A survey. *ACM computing surveys (CSUR)*, *41*(2), 1–69.

Ng, H. T., Goh, W. B., & Low, K. L. (1997). Feature selection, perceptron learning, and a usability case study for text categorization. *Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval*, 67–73.

Otter, D. W., Medina, J. R., & Kalita, J. K. (2020). A survey of the usages of deep learning for natural language processing. *IEEE transactions on neural networks and learning systems*, *32*(2), 604–624.

Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global vectors for word representation. *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*. https://doi.org/10.3115/v1/d14-1162

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. *NAACL HLT 2018 - 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, *1*, 2227–2237. https://doi.org/10.18653/v1/n18-1202

Peters, M. E., Neumann, M., Zettlemoyer, L., & Yih, W. T. (2020). Dissecting contextual word embeddings: Architecture and representation. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018*, 1499–1509. https://doi.org/10.18653/v1/d18-1179

Pollard, C. J. (1987). Information-based syntax and semantics, vol. 1 Fundamentals. *CSLI Lecture Notes Series*, *13*.

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving Language Understanding by Generative Pre-Training.

Rajpurkar, P., Jia, R., & Liang, P. (2018). Know what you don't know: Unanswerable questions for SQuAD. *arXiv preprint arXiv:1806.03822*.

Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016). SQuad: 100,000+ questions for machine comprehension of text. *EMNLP 2016 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, (2), 2383–2392. https://doi.org/10.18653/v1/d16-1264

Riksrevisjonen. (2018). Bilingual English-Norwegian parallel corpus from the Office of the Auditor General (Riksrevisjonen) website – ELRC-SHARE. https://www.elrc-share.eu/repository/browse/bilingual-english-norwegian-parallel-corpus-from-the-office-of-the-auditor-general-riksrevisjonen-website/a5d2470201e311e9b7d400155d0267060fffdc9258a

Rogers, A., Kovaleva, O., & Rumshisky, A. (2020). A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, *8*, 842–866.

Salton, G. (1962). Some experiments in the generation of word and document associations. *Proceedings of the December 4-6, 1962, fall joint computer conference*, 234–250.

Salton, G., Wong, A., & Yang, C.-S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, *18*(11), 613–620.

Schank, R. C., & Abelson, R. P. (1975). Scripts, plans, and knowledge. *IJCAI*, *75*, 151–157.

Schank, R. C., & Colby, K. M. (1973). *Computer models of thought and language*. WH Freeman San Francisco.

Schraer, R., & Mwai, P. (n.d.). Omicron: What can we learn from South Africa's experience so far? - BBC News. https://www.bbc.com/news/59667268

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems*, *4*(January), 3104–3112.

Tenney, I., Das, D., & Pavlick, E. (2019). BERT rediscovers the classical NLP pipeline. *arXiv preprint arXiv:1905.05950*.

Tenney, I., Xia, P., Chen, B., Wang, A., Poliak, A., McCoy, R. T., Kim, N., Van Durme, B., Bowman, S. R., Das, D., et al. (2019). What do you learn from context? probing for sentence structure in contextualized word representations. *arXiv preprint arXiv:1905.06316*.

Turing, A. M. (2009). Computing machinery and intelligence. *Parsing the turing test* (pp. 23–65). Springer.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, *2017-Decem*(Nips), 5999–6009.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. R. (2018). GLUE: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Wang, B., Shang, L., Lioma, C., Jiang, X., Yang, H., Liu, Q., & Simonsen, J. G. (2021). On position embeddings in BERT. *International Conference on Learning Representations*, *2*, 12–13.

Weizenbaum, J. (1966). ELIZA—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, *9*(1), 36–45.

Wiedemann, G., Remus, S., Chawla, A., & Biemann, C. (2019). Does BERT make any sense? Interpretable word sense disambiguation with contextualized embeddings. *arXiv preprint arXiv:1909.10430*.

Williams, A., Nangia, N., & Bowman, S. R. (2017). A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Wu, Y., & et al. (2016). Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *CoRR*, 1–23. http://arxiv.org/abs/1609.08144

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., & Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Appendix A

Examples of Cross-Lingual Word Sense Disambiguation

Some sentences are made up, some taken from news corpus Bokmål or Brown corpus.

Sentence	Language	@3 Norwegian	@3 English
The two offices could **be** combined to achieve better efficiency and reduce the cost of administration.	En	bli, være, slås	be, been, was
Mr. Reama , **far** from really being retired , is engaged in industrial relations counseling.	En	langt, langtfra, milevis	far, farther, hardly
It was hard **to** do the job without any supervision.	En	å, vanskelig, slitsomt	to, can, could
Hans la igjen Batman kostyme hjemme for å gå og **be** om knask eller knep i noe litt med kosete.	No	be, ba, ber	ask, requests, request
Det å bli **far** til en jente, endret synet mitt på en del ting.	No	far, mor, forelder	father, parent, parents
Hun fikk **to** fingre delvis amputert.	No	to, tre, fire	three, four, six

Appendix B

Language Detection Mixed Sentence Examples

Detected language for each word. Top row is Notram detection and bottom row is mBERT detection. Sentences are drawn from the Norwegian News Corpus[1].

[1] https://www.nb.no/sprakbanken/ressurskatalog/oai-nb-no-sbr-4/

And	door-to-door	and	clothes	from	containers	are	mostly	the	same	,	svarer	Bartusis	ifølge	TV	2	hjelper	deg	.
en	en	en	en	en	en	en	en	en	en	no	no	no	no	no	no	no	no	no
en	en	en	en	en	en	en	en	en	en	no	no	no	no	no	no	no	no	no

Wahlberg	hadde	roller	som	skuespiller	i	flere	filmer	som	``	Planet	of	the	Apes	"	og	``	Rock	Star	"	,
no	no	no	no	no	no	no	no	no	no	en	no	no	en	no	no	no	en	en	no	no
no	no	no	no	no	no	no	no	no	no	en	en	en	en	no	no	no	en	en	no	no
før	han	på	begynnelsen	av	2000-tallet	ville	prøve	seg	som	produsent	-	med	stor	suksess	.					
no	no	no	no	no	no	no	no	no	no	no	no	no	no	no	no					
no	no	no	no	no	no	no	no	no	no	no	no	no	no	no	no					

Gustafssons	nedtakning	av	Jones	,	den	første	Jon	Jones	hadde	opplevd	i	sin	karriere	,	hadde	rystet	oss	,	
no	no	no	en	no	no	no	en	en	no	no	no	no	no	no	no	no	no	no	
no	no	no	no	no	no	no	no	no	no	no	no	no	no	no	no	no	no	no	
og	Joe	Rogans	kommentarer	hørtes	fortsatt	i	ekko	:	``	the	champ	is	cut	above	the	right	eye	"	.
no	en	no	no	no	no	no	no	no	en	en	en	en	en	en	en	en	en	en	no
no	en	no	no	no	no	no	no	no	en	en	en	en	en	en	en	en	en	en	en

Nå	trenger	vi	handling	,	tordnet	Juncker	da	han	holdt	den	såkalte	``	state	of	the	union	"	talen	for	EU
no	no	no	no	no	no	no	no	no	no	no	no	en	en	en	no	en	en	no	no	no
no	no	no	no	no	no	no	no	no	no	no	no	en	en	en	en	en	en	no	no	no
parlamentet	i	Strasbourg	.	``	state	of	the	union	"	talen	for	EU	parlamentet	i	Strasbourg	.				
no	no	en	no	en	en	en	no	en	en	no	no	no	no	no	en	no				
no	no	no	no	en	en	en	en	en	en	no	no	no	no	no	no	no				

Investorene	lukker	øynene	for	at	penger	som	går	til	bosetningene	,	blokkerer	fredsprosessen	,
no	no	no	no	no	no	no	no	no	no	no	no	no	no
no	no	no	no	no	no	no	no	no	no	no	no	no	no
sier	Jamal	Juma	'	a	,	ansvarlig	for	``	Stop	The	Wall	"	kampanjen
no	no	no	no	no	no	no	no	en	no	no	en	no	no
no	no	no	no	no	no	no	no	no	no	en	en	no	no

Appendix C

Examples Contextual Words Pairs

The contextual word pairs are taken from parallel sentences between English and Norwegian from Riksrevisjonen[1]. They are aligned with the MUSE bilingual list[2].

[1] https://www.elrc-share.eu/repository/browse/bilingual-english-norwegian-parallel-corpus-from-the-office-of-the-auditor-general-riksrevisjonen-website/a5d2470201e311e9b7d400155d0267060fffdc9258a741659ce9e52ef15a7c26/:2/22/2022

[2] https://github.com/facebookresearch/MUSE

English	Norwegian	Word pairs
A and B switch quotas in order to specialise their respective operations , for example for pelagic fish like mackerel or herring or demersal fish like cod , haddock and saithe .	A og B bytter kvoter innbyrdes for å spesialisere driften på for eksempel pelagisk fisk som makrell og sild , eller bunnfisk som torsk , hyse og sei .	[quotas , kvoter] [example , eksempel] [mackerel , makrell] [herring , sild] [cod , torsk] [haddock , hyse]
A broad information basis would enable the ministry to discuss the organisation and usefulness of relevant projects with the responsible parties .	Et bredt informasjonsgrunnlag vil gjøre departementet rustet til å dis- kutere innretning og hensiktsmessigheten av de aktuelle prosjektene med de ansvarlige aktørene .	[broad , bredt] [basis , grunnlag] [ministry , departementet] [to , til] [projects , prosjektene] [with , med]
A central approval system has been introduced for the parties responsible for pre-engineering for the removal of hazardous components and for the parties responsible for executing construction/demolition work for demolition and removal of hazardous components .	Det er innført sentral godkjenningsordning for prosjekterende for området miljøsanering og for utførende for området riving og miljøsanering .	[central , sentral] [introduced , innført] [demolition , riving]
A buyer is free to sell the credits directly to another buyer or to trade them on an exchange .	En kjøper kan fritt velge å videreselge kredittene direkte til en annen kjøper eller selge dem på børs .	[free , fritt] [sell , selge] [directly , direkte] [or , eller] [to , til] [them , dem]