THE UNIVERSITY OF CHICAGO

IDENTIFYING LINGUISTIC STRUCTURES IN CONTINUOUS SPACES

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE DIVISION OF THE HUMANITIES
IN CANDIDACY FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

DEPARTMENT OF LINGUISTICS

BY
DANIEL STEPHEN EDMISTON

CHICAGO, ILLINOIS
JUNE 2021

To Grandpa in Heaven.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

Thanks first and foremost to John Goldsmith, my dissertation advisor. John, you have been an excellent teacher and friend these past few years, both gracious and patient. I don't know of anyone else who would have given me the freedom to explore my interests and grow as a scholar quite like you. I especially appreciate you indulging my interest in—and actually discussing with me—algebraic topology and topological data analysis during my fourth and fifth year, in spite of their rather tenuous relation to linguistics. It was this imagination and open-mindedness which always made me look forward to our discussions.

Thanks to Greg Kobele, whom I consider my honorary dissertation advisor. Greg, in exposing me to a completely different way of doing linguistics, you also exposed me to the beauty of mathematics. This is a gift I will carry with me till my dying day. I count myself as immensely fortunate to have been able to study under you for two years in Chicago.

Thanks to Allyson Ettinger. Thank you for being so generous with your time, and for providing such thoughtful commentary. You always asked the most penetrating questions, but did so with such positive energy that your constructive criticism never really felt like criticism. Though we didn't get many opportunities to work together, I can tell that you (i) care about your students, and (ii) have great scientific instincts.

Thanks to Chris Kennedy. You, too, were more than generous with your time, both in my early years when I was studying semantics, and in my latter years as I shifted to computational linguistics. Thank you for your time, your feedback, and your open-mindedness.

Thanks also to Jason Riggle. We didn't talk too much about linguistics (twice, maybe three times?), but I'm okay with that since it was so much fun talking to you about everything else. I remember fondly you dancing at CLS, the yearly prospie parties at your house, and all the hilariously irreverent things you said over the years.

Before arriving in Chicago, I had the privilege of learning under many wonderful teachers. In particular, thank you to Eric Potsdam. I couldn't have asked for a better teacher early on in my career, as it was in your classes that I began to appreciate the beauty in language.

Thank you for allowing me to spend a year in Gainesville as a visiting scholar and for the many letters of recommendation over the years. It's not an exaggeration to say that I wouldn't be where I am without your help.

Thank you to the many professors at Seoul National University who have helped me along the way. Specifically, thank you to Heejeong Ko. You, too, have been instrumental in my early career's development. You taught me what is involved in doing research, and your uncompromising standards are an example to all your students. Thank you to Seungho Nam. I remember you went out of your way to make me and other foreign students feel welcome at SNU. A special thank you to Sang-goo Lee for allowing me to spend two summers in your lab. You didn't know me and had no reason to be so generous, but through your kindness I gained valuable experience and met many wonderful people.

Thank you also to Stefan Huber, my first linguistics teacher. I had so much fun in your class 13 years ago that I decided to change majors. One particularly memorable thing from that class was whenever a student had a question you weren't sure about, you'd tell them they should write a dissertation on the matter. Well, I did.

Without question, the best memories from my time at the University of Chicago were made with my friends. From the department, I wish to thank in particular Brandon, Jackie, Marina, and Orest. You guys were my core group, and it would have been a far more empty experience without all of you. I'd also like to thank Kartik and Woojae for being such great roommates, and Taeuk and Jihun for teaching me so much about NLP and sharing fun times in Seoul. Thanks also to Karl. I loved our chats in your TTIC office, and I look forward to many more such chats in the future. I also wish to thank those at the Hyde Park Korean United Methodist Church.

Finally, thank you to my family. Thank you to both of my loving parents, I can always count on your unwavering support. And thank you to my wife for being so patient, and to my daughter for being so lovely.

ABSTRACT

While most theories regarding the various aspects of human language are couched in the language of discrete mathematics, modern approaches to natural language processing (NLP) generally rely on continuous representations of linguistic units, and all but eschew linguistic theorizing. An emerging consensus on why these methods have been so successful in practical applications is that the flexibility of modern neural network techniques allows for the encoding of discrete linguistic structures in these models' continuous embedding spaces. This dissertation takes this view, and explores the continuous space representations of BERT (Devlin et al., 2019), a recent model which has shown to be particularly successful in NLP applications.

In experiments collected in three chapters, this dissertation uses multiple machine learning techniques to explore BERT's continuous space in search of evidence of assorted morpho-syntactic features, different aspects of lexical semantics (e.g. sentiment), as well as morphophonology and pragmatics. In all cases, the experiments provide evidence that the relevant linguistic information is encoded in BERT's embedding space.

Results of supervised classifiers provide strong evidence that BERT encodes words in a manner highly predictive of inflectional feature values. Further experiments show that it is possible to factor BERT's word embeddings via a generative neural network where the latent variables are inspired by generative linguistic theorizing. The result is so-called disentangled representations, and is a step towards more interpretable representations for NLP. Finally, assorted data-visualization techniques demonstrate that BERT's embedding space can be partitioned in a way which is highly predictive of both morphophonological and pragmatic features. In addition to contributing to the literature on interpretability of neural network models for NLP, this dissertation hopes to shed light on what aspects of language can be learnt by models like BERT, with no in-built linguistic structure and no supervision.

CHAPTER 1

INTRODUCTION

1.1 Approaches to Linguistic Representation

In theoretical linguistics, the study of human language at virtually every level of analysis is formalized in the language of discrete mathematics. For example, the study of phonology concerns itself with finite sets of elements (phonemes), the principles behind (il)licit orderings of these elements in strings (phonotactics), and transductions between strings of these elements. Morphology likewise is concerned with finite sets of elements (morphemes) and (il)licit orderings of these elements (morphotactics). At the level of syntax, the finite elements are often words, and a chief concern is the potential orderings and underlying hierarchical arrangements of these elements. Semantic theories often enrich syntax with additional structure in order to describe potential meanings of utterances. While these descriptions of different subfields of linguistics are necessarily simplified here, they convey the fact that the principal objects of study in linguistics are generally discrete, with the necessary mathematical machinery required for their description being sets, strings, tree-like structures (graphs), grammars/logics, and abstract algebra.

In the parallel field of natural language processing (NLP), the object of study is likewise language. However, modern approaches in NLP are based almost entirely on continuous mathematics, treating linguistic units (characters, words, sentences, etc) as points in continuous space, and relying on a modeling framework dependent on differentiable functions (neural networks). Furthermore, most modern methodologies in NLP eschew explicit importation, or even consideration, of linguistic theory when designing models. Recent successes of these methods on empirical tasks in NLP have led some to question whether this approach is capturing certain underlying truths concerning the nature of human language which are missing in mainstream linguistic thought (see e.g. Hannun et al. 2014).

However, there is reason to believe that the truth may be more nuanced, and there is

growing evidence that the continuous approach to language common in NLP is actually discovering and encoding discrete aspects of natural language in continuous space (see e.g. Coenen et al. 2019). The successes of such purely continuous methodologies would then be attributable, at least in part, to the function-approximating power of neural networks (Hornik et al., 1989; Hornik, 1991) rather than any inherent superiority in continuous representations.

Yet neural network-based NLP systems have consistently shown state-of-the-art performance on a host of varied tasks which presumably require knowledge of phonology (e.g. automatic speech recognition), morphology (e.g. feature tagging), syntax (e.g. constituency parsing), semantics (e.g. natural language inference), and even pragmatics (e.g. sentiment analysis), aspects of language which require mathematical objects of very different natures for their descriptions. This means that if it is indeed the case that these continuous models are encoding linguistic structures, the obvious question is, 'How?'. How are such varied linguistic structures being encoded in the high-dimensional Euclidean spaces within which most NLP systems work?

This thesis approaches this complex question by focusing on different aspects of language, mostly at the level of the word (Chapters 2 and 3), but also at the level of individual morphemes (Chapter 4). It does so by investigating embeddings produced by a recent state-of-the-art model known as BERT (Devlin et al., 2019), thus adding to a burgeoning literature colloquially known as *BERTology*. Through varied methodologies, experiments in this thesis analyze the complexity of BERT embeddings, identify evidence of linguistic structure in the form of morphological features in BERT embeddings, assess the role syncretism plays in "confusing" BERT, as well as factor BERT embeddings to arrive at more linguistically interpretable representations. Experiments touch on inflectional morphology, syntactic features, lexical semantics, morphophonology, and honorifics/polite speech. While far from an exhaustive study on the types and scope of linguistic information in BERT-style models, it is the hope that this thesis advances the state-of-the-art by exposing new forms of linguistic information hitherto undiscovered in these models, as well as by introducing multiple

methodologies not typically used in computational linguistics or NLP, but which are useful for examining high-dimensional continuous data more broadly.

1.2 Embeddings: Representing Linguistic Units in Continuous Space

The term embedding in the context of natural language processing refers to a representation of a linguistic unit by an element in a usually high-dimensional, usually continuous space, for example $\mathbb{R}^{300}$. While such representations are generally opaque to human understanding, they are useful as input to many machine learning algorithms, and serve as the foundation of modern natural language processing. Linguistic entities of all sorts have been *embedded* for use in natural language processing algorithms, including characters (Kim et al., 2016), n-grams (Bojanowski et al., 2017), morphemes (Qiu et al., 2014), words (Mikolov et al., 2013b), sentences (Wieting et al., 2015), paragraphs (Le and Mikolov, 2014), and documents (Lau and Baldwin, 2016). Embeddings are generally derived automatically based on unlabeled text, and can be generated in multiple ways.

1.2.1 Count-based Methods

An early example of linguistic embeddings are those as derived via the TF-IDF (term-frequency, inverse document frequency) algorithm (Jones, 1972), which simultaneously provides document and term (i.e. word) embeddings. Given a corpus D of documents constructed from a vocabulary V, the output of TF-IDF is a $|D| \times |V|$ matrix M. The representation (i.e. embedding) of the i^{th} document is the i^{th} row of M, and the representation of the j^{th} term is the j^{th} column of M. Entry $M_{i,j}$ is defined as the term frequency, how many times term j appears in document i divided by the length of the document, multiplied by the inverse document frequency, which is $|D|$ divided by the number of documents i' that

term j appears in.[1]

As the values of matrix M are determined by term frequencies, TF-IDF is an example of a count-based approach to deriving embeddings. While representations provided by TF-IDF are intuitively simple and interpretable, a significant drawback in practice is the large dimensionality required due to the size of the vocabulary $|V|$, which increases as more documents are added. In addition, the document representations filling the space are sparse, and become exponentially more sparse as the vocabulary grows.

A solution to the problems associated with dimensionality come in the form of matrix decomposition methods, principal among which are Latent Semantic Analysis (LSA) (Deerwester et al., 1990). LSA is likewise a count-based method producing term and document embeddings, and proceeds by performing truncated singular value decomposition (SVD) on a term-by-document matrix $M \in \mathbb{R}^{|D|\times|V|}$ to produce $M' \in \mathbb{R}^{|D|\times P}$, where P is a hyperparameter. It is often the case that P can be chosen such that the resultant matrix M' contains a large proportion of the variance originally in M and yet $|V| \gg P$.[2] Furthermore, projection of count vectors into the lower dimensional space as entailed in SVD has as a by-product the ability to make apparent latent relationships between words and documents. However, while representations as produced by LSA are generally more efficient and more useful in tasks such as information retrieval, the move from sparse to dense representations resulting from the application of SVD often results in a loss of interpretability of individual dimensions.

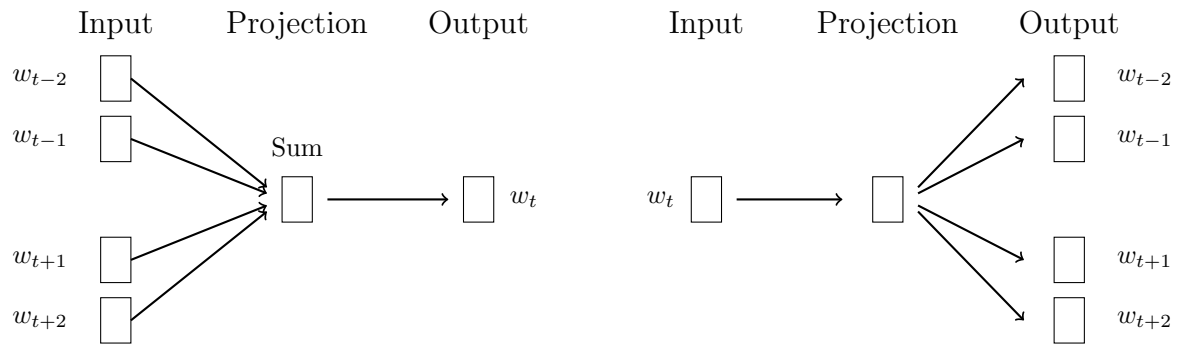
1. There exist multiple alternatives for calculating the exact values in the TF-IDF matrix.

2. While proportion of variance maintained is generally an important consideration when choosing dimensionality for algorithms such as truncated SVD, in the particular case of LSA considering variance is generally discouraged in favor of empircal measures such as performance on downstream tasks (Landauer et al., 1998). Nevertheless the point remains that LSA is able to significantly reduce dimensionality as compared to non-decomposed term-by-document matrices.

1.2.2 Prediction-based Methods

Opposed to count-based methods of learning embeddings are prediction-based methods. In such methods, embeddings are generally randomly initialized in a continuous space of pre-determined dimensionality, and are recursively updated to optimize a prediction-based objective. An early example of such methods was as in Bengio et al. (2003), who learn "distributed representation(s) for words" (i.e. word embeddings) via the task of training of a language model by iteratively predicting the next word in a sequence.

Another influential prediction-based word embedding model of the recent past is the Word2Vec algorithm of Mikolov et al. (2013a,b). Each of the two variants of this algorithm, the *continuous-bag-of-words* (CBOW) variant and the *Skip-gram* variant, are representative of how prediction-based tasks can be used to train word embeddings. Word2Vec takes as input a sequence of words from a corpus, where the sequence consists of a center word w_t, and words on either side of w_t within some fixe-length window, e.g. $w_{t-2}, w_{t-1}, w_t, w_{t+1}, w_{t+2}$. The CBOW variant of Word2Vec then uses the representations of the context words to predict the center word, and the Skip-gram variant uses the center word's representation to predict the surrounding context words. These two variants are schematized as in Figure 1.1.



(a) CBOW (b) Skip-gram

Figure 1.1: Schematics for Word2Vec variants. Visual adapted form Mikolov et al. (2013a).

When first introduced, Word2Vec provided a leap forward for word embeddings, both in

terms of training efficiency and in terms of performance, and Baroni et al. (2014) show that in general such prediction-based methods are superior to count-based methods. Word2Vec inspired a host of similar algorithms used for training embeddings for both longer text sequences (e.g. documents; Le and Mikolov 2014) and subword units (e.g. n-grams; Bojanowski et al. 2017).

1.2.3 State-of-the-Art: Contextual Embeddings

While embeddings such as those provided by algorithms like Word2Vec proved to be a significant improvement over previous methods, they are limited in their ability to model meanings of words in (at least) one crucial respect: their inability to account for context. The representations assigned to words by an algorithm such as Word2Vec are static. This is in obvious contrast with natural languages such as English, where a word can assume different meanings—even different parts-of-speech—depending on the context in which it appears. For example, consider the word *bank* in the following sentences.

(1) The fighter pilot nearly lost consciousness while attempting to *bank* at high speeds.

(2) The robbers entered the *bank* through the ventilation above the vault.

As Word2Vec is simply a mapping from words to embeddings in $\mathbb{R}^n$, it is unable to reflect the fact that word meanings may change depending on context. Any model which seeks to more faithfully represent meaning in natural language must reflect the dynamism inherent in natural language usage. To account for this aspect of natural language, more recent models construct *contextualized* embeddings, mappings from words to embeddings which are sensitive to context.

One of the first models to produce contextualized embeddings was the CoVe (**Co**ntextualized Word **Ve**ctors) model of McCann et al. (2017), who given a sequence use a pre-trained LSTM to encode that sequence, and append the resultant vectors to static embeddings. Specifically, given a sequence of words $w_1 \dots w_n$, the CoVe algorithm first converts the sequence of words

to a sequence of GloVe (Pennington et al., 2014) embeddings $\text{GloVe}(w_1) \dots \text{GloVe}(w_n) = g_1 \dots g_n$, and encodes this sequence using a pre-trained LSTM to create a sequence of hidden states $LSTM(g_1 \dots g_n) = h_1 \dots h_n$. The final contextualized embeddings are the concatenation of each hidden state with each GloVe embedding, i.e. $[g_1; h_1] \dots [g_n; h_n]$, with $[g_i, h_i]$ being the contextualized embedding for the i^{th} word in the sequence.

Returning to sentences (1) and (2) above, it is then clear how CoVe's representation of *bank* in each of the examples would differ. In (1), *bank*'s representation would include information from words such as *fighter* and *pilot*, while in (2), it would include information from words such as *robbers* and *vault*. The introduction of such contextualized information into word embeddings led to improved results on a host of downstream tasks when compared against models using only static embeddings, and subsequent models such as Peters et al. (2018)'s ELMo (**E**mbeddings from **L**anguage **Mo**dels) have further improved upon CoVe.

Currently, the state-of-the-art in virtually every NLP task is held by a model based on the transformer architecture of Vaswani et al. (2017). Particularly, the BERT model (Devlin et al., 2019) has been highly influential. As embeddings produced by different BERT models are the source of data used in all experiments in this dissertation, a technical description of the transformer architecture, and BERT in particular, follow.

1.3 The Transformer Architecture and BERT

1.3.1 The Transformer

The transformer (Vaswani et al., 2017) is a deep neural network architecture originally introduced to perform sequence transduction tasks such as machine translation. Like many modern sequence transduction models, the transformer consists of two separate modules, an **Encoder** and a **Decoder** (Bahdanau et al., 2014). The **Encoder** is a function from sequences of entities from some vocabulary to sequences of vectors in some high-dimensional continuous space. The **Decoder** is likewise a function, this time going from sequences of

vectors (the output of the **Encoder**) to sequences of entities in some vocabulary, which may or may not be the same vocabulary from the **Encoder**.

Here, the **Encoder** and **Decoder** functions are implemented as deep neural networks, each making use of the subroutines **Multihead-Attention** and **Layer-Normalization**. After exploring **Multihead-Attention** and **Layer-Normalization**, we examine **Encoder** and **Decoder** in turn.[3]

Multihead-Attention

Multihead-Attention is a function so-named as it relies on multiple instances of attention (Bahdanau et al., 2014), or mulitple "attention heads." The input to **Multihead-Attention** is two sequences $x_1 \dots x_n$ and $y_1 \dots y_m$, and its output, $\mu_1 \dots \mu_n$, is a transformation of sequence $x_1 \dots x_n$ which is sensitive to sequence $y_1 \dots y_m$.

Assuming the number of attention heads is H, then for each head $h = 1 \dots H$ the function assumes four weight matrices, $W_{h,Q}$, $W_{h,K}$, $W_{h,V}$, and U_h. Here, Q, K, and V are mnemonic for *query*, *key*, and *value* respectively. Each of $W_{h,Q}$, $W_{h,K}$, and $W_{h,V}$ is in $\mathbb{R}^{p \times d}$, where d is the dimensionality of input embeddings, and p is generally significantly smaller than d, e.g. all weight matrices are in $\mathbb{R}^{64 \times 512}$ in Vaswani et al. (2017). Much of the computation in **Multihead-Attention** takes place in the smaller $\mathbb{R}^p$, leading to significant computational savings such that the total cost of all heads combined is roughly equal to a single attention head in the original dimensionality. Each matrix U_h is in $\mathbb{R}^{d \times p}$, and the purpose of these matrices is to project back to the original space $\mathbb{R}^d$ for the final output. A forward pass of **Multihead-Attention** is as in Algorithm 1.

In words, **Multihead-Attention** transforms the first input sequence such that its entries become the sums of convex combinations of values determined by the second input sequence. In this way, it is similar to standard attention mechanisms (Bahdanau et al., 2014), but with

3. The presentation here is inspired by the technical notes found at `http://karlstratos.com/notes/transformer17.pdf`.

Algorithm 1 Multihead-Attention

Input: sequence1 $x_1 \dots x_n$, $x_i \in \mathbb{R}^d$, sequence2 $y_1 \dots y_m, y_i \in \mathbb{R}^d$
for $h \in 1 \dots H$ **do** ▷ For each head h
  for $i \in 1 \dots n$ **do** ▷ Calculate *query* representations for $x_i \in \mathbb{R}^p$
    $x_i^{(h,q)} = W_{h,Q} x_i$
  for $i \in 1 \dots m$ **do** ▷ Calculate *key*, *value* representations for $y_i \in \mathbb{R}^p$
    $y_i^{(h,k)} = W_{h,K} y_i$
    $y_i^{(h,v)} = W_{h,V} y_i$
  for $i \in 1 \dots n$ **do** ▷ sequence1 *queries* sequence2 *keys*
    for $j \in 1 \dots m$ **do**
      $\alpha_{i,j}^{(h)} = \frac{x_i^{(h,q)} \cdot y_j^{(h,k)}}{\sqrt{p}}$ ▷ p as in $\mathbb{R}^p$
    $\beta_{i,1}^{(h)} \dots \beta_{i,m}^{(h)} = Softmax(\alpha_{i,1}^{(h)} \dots \alpha_{i,m}^{(h)})$ ▷ Calculate attention weights
    $\mu_i^{(h)} = \sum_{j=1}^{m} \beta_{i,j}^{(h)} y_j^{(h,v)}$ ▷ Convex combination of sequence2 *values*
for $i \in 1 \dots n$ **do**
  $\mu_i = \sum_{h=1}^{H} U_h \mu_i^{(h)}$ ▷ Project back to $\mathbb{R}^d$ and combine attention heads
Output: $\mu_1 \dots \mu_n, \mu_i \in \mathbb{R}^d$

multiple heads. The term *self-attention* refers to an instance of **Multihead-Attention** in which both input sequences are the same sequence.

Layer-Normalization

Layer-Normalization (Ba et al., 2016) is an additional type of layer used in neural networks, generally added between linear layers and non-linear activation layers. The purpose of **Layer-Normalization** is to speed up neural network training. The high-level intuition is that as neural networks learn, particularly in the early stages of learning, the shifts in network weights can be dramatic and lead to large shifts in the distributions of the outputs of individual layers as training progresses. These shifting distributions result in "moving targets" which weights must learn to optimize. To ease the burden on individual layers, normalization layers can be added which revert outputs to predictable and consistent distributions. These normalization layers can be easily integrated with ordinary neural network

layers, as they are differentiable and amenable to back-propagation.

Given the output of some previous layer, **Layer-Normalization** proceeds by calculating the mean and standard deviation of the output's neuron activations, subtracting the mean from the output and dividing by the standard deviation plus some small constant c. **Layer-Normalization** then proceeds with pointwise multiplication of the resulting vector and vector g, a *gain* hyperparamter, finally adding a vector b, a *bias*. A forward pass is as in Algorithm 2.

Algorithm 2 Layer-Normalization

1: **Input:** vector $x \in \mathbb{R}^d$
2: $\mu = mean(x_1 \dots x_d)$ ▷ Calculate mean of individual activations
3: $\sigma = StdDev(x_1 \dots x_d)$ ▷ Calculate standard deviation of individual activations
4: $\bar{x} = g \odot \frac{x-\mu}{\sigma+c} + b$
5: **Output:** vector $\bar{x} \in \mathbb{R}^d$

Encoder

Like many deep neural networks used in NLP, the **Encoder** function can be described as the composition of an initial embedding function and multiple *layer* functions.

$$\mathbf{Encoder} = \mathbf{Layer}^{(l_{Enc})} \cdots \circ \dots \mathbf{Layer}^{(1)} \circ \mathbf{Embedding} \tag{1.1}$$

Embedding here is a function which takes sequences of elements over some vocabulary and returns sequences of vectors, *embedding* each element in the sequence. Given vocabulary V and some sequence from that vocabulary, $x_1 \dots x_n$, $x_i \in V$, **Embedding** is defined here as:

$$\mathbf{Embedding}(x_1 \dots x_n) = \mathbf{Drop}(\mathbf{lookup}(x_1) + \pi_1) \dots \mathbf{Drop}(\mathbf{lookup}(x_n) + \pi_n) \tag{1.2}$$

where **Drop** is dropout (Srivastava, 2013), **lookup** is a function $V \to \mathbb{R}^d$, taking elements

in V to embeddings in $\mathbb{R}^d$, and $\pi_i \in \mathbb{R}^d$ is the i^{th} position embedding. The output of **Embedding** is then a sequence of vectors in $\mathbb{R}^d$. Given this definition of **Embedding**, Algorithm 3 then describes **Encoder**.

Algorithm 3 Encoder

1: **Input:** sequence $x_1 \dots x_n, x_i \in V$
2: $z_1 \dots z_n = \textbf{Embedding}(x_1 \dots x_n)$
3: **for** $l = 1, \dots, l_{Enc}$ **do** ▷ l_{Enc} total layers
4: $\quad \tilde{z}_1 \dots \tilde{z}_n = \textbf{Multihead-Attention}(z_1 \dots z_n, z_1 \dots z_n)$ ▷ Self-attention
5: $\quad$ **for** $i = 1, \dots, n$ **do**
6: $\quad\quad \bar{z}_i = \textbf{Layer-Normalization}(z_i + \textbf{Drop}(\tilde{z}_i))$
7: $\quad\quad z_i = \textbf{Layer-Normalization}(\bar{z}_i + \textbf{Drop}(\textbf{Linear}(\bar{z}_i)))$
8: **Output:** sequence $z_1 \dots z_n, z_i \in \mathbb{R}^d$

Here, **Linear** is a 2-layer feed-forward neural network with ReLU activation in the hidden layer. Note that the **Multihead-Attention** function in line 4 is being run with the same sequence twice as input, amounting to self-attention.

Decoder

Given **Encoder** output $z_1 \dots z_n$, the **Decoder** module generates sequence output $y_0 \dots y_m$, where y_0 is pre-determined to be a unique BOS (beginning-of-string) token. Each token y_j in the output sequence is generated auto-regressively, meaning at each step $j = 1 \dots$, **Decoder** has access to generated symbols $y_0 \dots y_{j-1}$. At each generation step $j = 1 \dots$, **Decoder** can then be given a similar high-level treatment as **Encoder**, that is as the composition of functions.

$$\textbf{Decoder} = \textbf{Layer}^{(l_{Dec})} \cdots \circ \dots \textbf{Layer}^{(1)} \circ \textbf{Embedding} \tag{1.3}$$

The inputs for **Decoder** at step j are then the output of **Encoder**, $z_1 \dots z_n$, along with the hitherto generated output sequence $y_0 \dots y_{j-1}$. $\textbf{Embedding}(y_0 \dots y_{j-1})$ is treated as in the **Encoder**, though potentially with a different **lookup** mapping. Algorithm 4 then describes **Decoder**.

Algorithm 4 Decoder

Input: Encoder output $z_1 \dots z_n$
$y = [\,]$ ▷ Initialize output sequence y
$prediction = \text{BOS}$
$y.append(prediction)$ ▷ y_0 predetermined to BOS token
while $prediction \neq \text{EOS}$ **do**
  $o_0 \dots o_{j-1} = \textbf{Embedding}(y_0 \dots y_{j-1})$ ▷ Embed j-length sequence in y
  for $l = 1, \dots, l_{Dec}$ **do** ▷ l_{Dec} total layers
    $\tilde{o}_0 \dots \tilde{o}_{j-1} = \textbf{Multihead-Attention}(o_0 \dots o_{j-1}, o_0 \dots o_{j-1})$ ▷ Self-attention
    for $i = 0, \dots, j-1$ **do**
      $\bar{o}_i = \textbf{Layer-Normalization}(o_i + \textbf{Drop}(\tilde{o}_i))$
    $\underline{o}_0 \dots \underline{o}_{j-1} = \textbf{Multihead-Attention}(\bar{o}_0 \dots \bar{o}_{j-1}, z_1 \dots z_n)$ ▷ Attend to encoding
    for $i = 0, \dots, j-1$ **do**
      $\hat{o}_i = \textbf{Layer-Normalization}(\bar{o}_i + \textbf{Drop}(\underline{o}_i))$
      $o_i = \textbf{Layer-Normalization}(\hat{o}_i + \textbf{Drop}(\textbf{Linear}(\hat{o}_i))$
  $prediction \sim Softmax((E^O)^\top o_{j-1})$ ▷ E^O is output embedding matrix
  $y.append(prediction)$
Output: sequence y

1.3.2 BERT: Pretraining Transformer Encoders

The BERT (**B**idirectional **E**ncoder **R**epresentations from **T**ransformers) model (Devlin et al., 2019)—the model from which all embeddings are drawn for this dissertation—is a deep neural network model based on the transformer architecture (Vaswani et al., 2017) discussed above, using only the Encoder portion of the architecture. As such, BERT serves as a sequence encoder, going from sequences of text to sequences of embeddings as described in Algorithm 3. A schematic of BERT encoding a single sentence is as in Figure 1.2.

In this figure, we see that the encoding of a sentence is a two-step process, with tokenization occurring before application of the BERT encoder. In tokenization, sentence of length m consisting of words $w_1 \dots w_m$ is tokenized into sequence of tokens of length n, $t_1 \dots t_n$, via the wordpiece algorithm (Wu et al., 2016); as this is a subword tokenizer, $n \geq m$. The input to BERT is then the sequence of tokens $t_1 \dots t_n$, plus two additional tokens, a [CLS] token and a [SEP] token, added at the beginning and end of the sequence respectively.[4] An

4. The [CLS] token, mnemonic for classification, is used for fine-tuning of BERT models, while the [SEP] token, mnemonic for separation, is used to separate sentences when more than one sentence is encoded; the

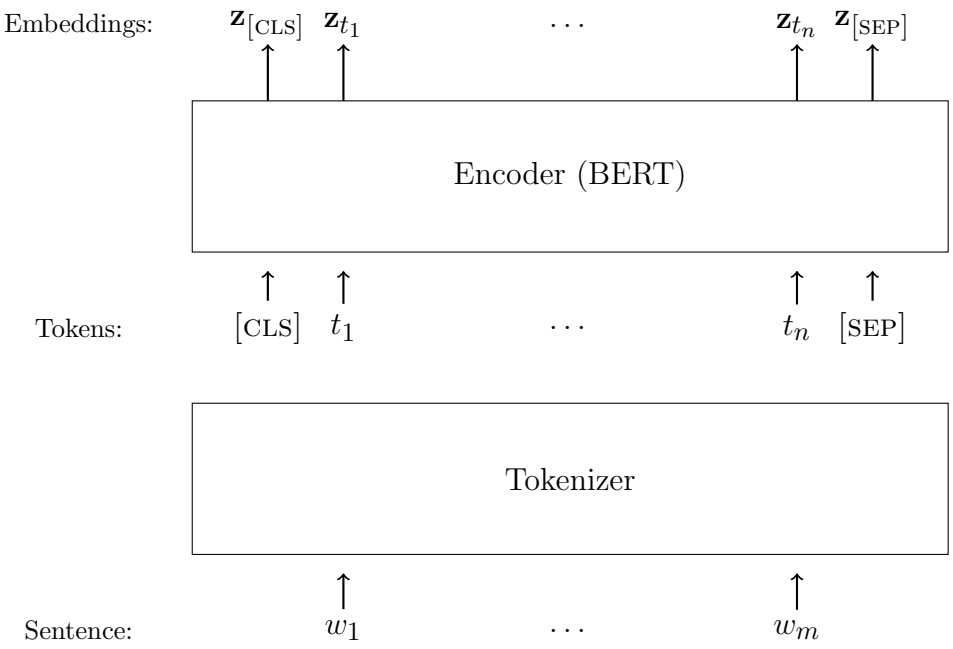


Figure 1.2: Schematic for BERT encoding of a sentence.

example of wordpiece tokenization for sentence "Karlos likes pork best" is as in Figure 1.3.

BERT can encode sequences of up to 512 tokens,[5] and is pre-trained on unlabeled text via the self-supervised tasks of masked language modeling and next sentence prediction; each of these tasks, in addition to fine-tuning pre-trained models for downstream tasks, is discussed below.

Masked Language Modeling

The traditional way of approaching language modeling with neural networks is to do so auto-regressively, which was demonstrated above in the description of the **Decoder** module of the transformer. Auto-regressive language modeling is the generative process in which the probability of a word(/character) at time step t is dependent upon words(/characters)

significance of these tokens will become apparent when discussing fine-tuning and next-sentence prediction below.

5. There is a subliterature on BERT-style models which seeks to extend models to longer sequences so as to enable the encoding of e.g. short documents. Among these long-form BERT-style models are Longformer (Beltagy et al., 2020) and BlockBERT (Qiu et al., 2019).

Input:		Karlos	likes	pork	best	
Output:	[CLS]	_Karl os	_likes	_pork	_best	[SEP]

Figure 1.3: Example of wordpiece tokenization. Four-word sentence results in token sequence of length seven. Underscore is a special character denoting the beginning of a word.

generated up to time step t. The probability of a sequence $x_1 \dots x_t$ is then $p(x_1 \dots x_t) = p(x_1)p(x_2|x_1) \dots p(x_t|x_1, \dots, x_{t-1})$. Typically, autoregressive neural network language modeling has been done using recurrent neural networks (see e.g. Mikolov et al., 2010), though more recent influential work (e.g. GPT models; Radford et al., 2018) has used transformer decoders.

However, since transformer encoders have access to the entirety of a sequence during encoding, they cannot be used auto-regressively for language modeling. As this is the case, Devlin et al. (2019) propose the masked language modeling task as an alternative to auto-regressive modeling. In the masked language modeling task, a fixed percentage of tokens in a sequence (15% in the original task) are masked out and targeted for prediction. A schematic of this is as in Figure 1.4.

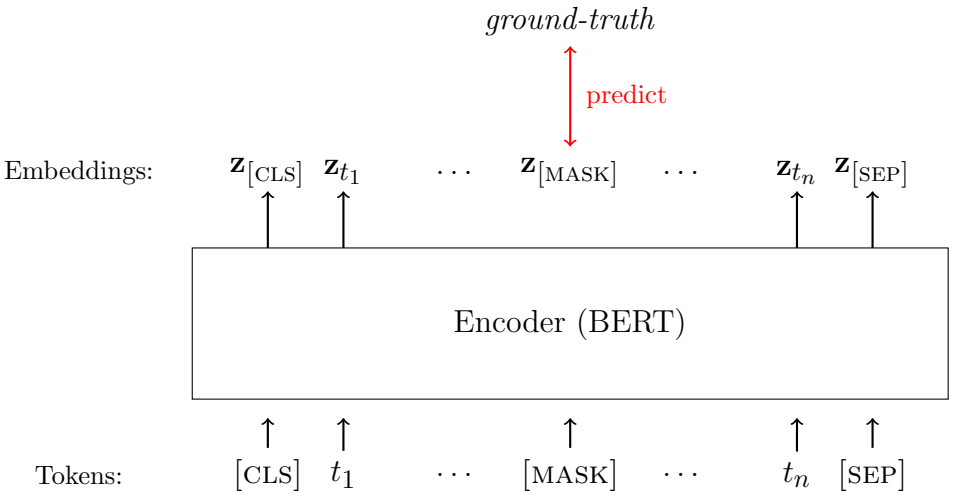


Figure 1.4: Schematic for masked language modeling task.

Next Sentence Prediction

A number of downstream NLP tasks, such as natural language inference or question answering, concern relationships between sentences. Next sentence prediction is a task which is meant serve as a viable means of pre-training for the type of sentence-level information useful for such downstream tasks. Training instances are constructed from pairs of sentences, A and B, where B may or may not be the actual sentence following A in some corpus. The sentence pair is then jointly encoded via BERT—separated by the special [SEP] token mentioned above—and a binary classifier is trained to predict either IsNext or NotNext given the output embedding of the [CLS] token. In such cases, the output encoding of the [CLS] token can be used as an encoding of the entire sequence.

Fine-tuning BERT

The tasks of masked language modeling and next sentence prediction are of little use in and of themselves, but prove very useful for what is known as *pre-training*. Pre-training refers to the training of a model's parameters on an auxiliary task before *fine-tuning*, training for the actual downstream tasks. In the case of BERT, it has been shown that pre-training via masked language modeling and next sentence prediction before fine-tuning results in great improvements on such varied tasks as predicting grammaticality judgments (Warstadt et al., 2019), sentiment analysis, and natural language inference (Williams et al., 2018), tasks encompassing both syntactic and semantic aspects of natural language.

Pre-training BERT-style encoders via the tasks of masked language modeling and next sentence prediction is thought to be useful for downstream tasks because the sort of linguistic knowledge necessary for these tasks is also useful for predicting masked words in context and whether one sentence follows another. The pre-training tasks are then thought to prime BERT's parameters with general knowledge about language. If this is true, BERT models bear on the question of what aspects of natural language can be learnt by domain-neutral learning algorithms with no linguistic priors, and it is worth investigating the linguistic

information in such models.

1.3.3 BERTology: Analyzing BERT Models

A large subliterature—which has taken on the title *BERTology* (Rogers et al., 2021)—has emerged in the NLP literature which investigates BERT and its peers for linguistic information. Work in BERTology has used multiple approaches to investigate multiple linguistic aspects, with a heavy focus on syntax. A host of works (Clark et al., 2019; Htut et al., 2019; Vig and Belinkov, 2019) investigate syntactic dependencies via the attention heads in BERT, other works (Hewitt and Manning, 2019; Coenen et al., 2019; Kim et al., 2020) have shown evidence that syntactic trees may be embedded in BERT's high-dimensional embedding space, and Goldberg (2019) provides evidence that BERT captures subject-verb agreement in English via a masked prediction task. Work in BERTology has also investigated morphological knowledge (Edmiston, 2020; Hofmann et al., 2020) and lexical semantics (Vulić et al., 2020), and Ettinger (2020) runs BERT through experiments from the psycholinguistic literature to show that BERT makes predictions which somewhat—though not entirely—resemble the predictions of human participants.

Interestingly, multiple works have investigated individual layers within BERT, showing that BERT appears to encode different aspects of language at different depths (Tenney et al., 2019). For instance, Jawahar et al. (2019) show that BERT's internal representations tend to focus on surface features in lower layers, syntactic information in middle layers, and semantic features in the later layers. In Chapter 2, I likewise run layer-specific experiments.

Work exploring BERT's representations thus addresses many facets of language, and does so in myriad ways, with works generally divisible into those using probing classifiers (e.g. Hewitt and Manning 2019) vs. those taking a more explicitly geometric approach (e.g. Coenen et al. 2019). See Rogers et al. (2021) for a recent survey on BERTology. Finally, it is worth noting that analysis of continuous high-dimensional embeddings as used in NLP precedes BERTology by some years; see Belinkov and Glass (2019) for a survey of analysis

methods for neural network models in NLP not specific to BERT.

1.4 Contributions

The contributions of this dissertation can be best situated in the context of *BERTology* in that it explores continuous representations produced by BERT models for evidence of linguistic structure. However, in the bigger picture the true point of this endeavor is not necessarily to better understand the BERT model itself, but to address the question of what aspects of language can be learnt by models such as BERT, with BERT(-style) models currently serving as the best proxy for showing what can be learnt from purely distributional information with no inherent linguistic structure built-in. In other words, investigating BERT here touches on the question of how much of language can be learnt from domain-general learning mechanisms. The dissertation serves to advance the field in exploring multiple types of linguistic information not yet investigated in the literature, and also by introducing methodologies from outside fields to explore the continuous embedding spaces in novel ways.

Chapter 2 investigates BERT for inflectional morphology, building on experiments from Edmiston (2020). Specific contributions of this chapter include showing that BERT models in general can reliably predict values for multiple morphological features given a word embedding, as well as comparison between how well BERT tags different features. In addition, Chapter 2 examines the effect that syncretism has on classifiers' ability to correctly identify feature values given BERT embeddings. Finally, Chapter 2 runs intrinsic dimensionality experiments on BERT embeddings which suggest that the intrinsic dimensionality of BERT's representations are far lower than its 768 embedding space. Such experiments shine light on the underlying shape of the hypothetical manifold from which BERT embeddings are drawn, and are the first of their kind to the author's knowledge.

Chapter 3 applies (semi-supervised) variational autoencoders (VAEs;Kingma et al., 2014; Kingma and Welling, 2013) to multi-lingual BERT (MBERT) embeddings so as to perform a form of disentangled representation learning (Higgins et al., 2016; Siddharth et al., 2017;

Dupont, 2018), a method which seeks to factor continuous embeddings into a more interpretable form. To the author's knowledge, this is the first instance of disentangled representation learning applied to word embeddings. In order to perform disentangled representation learning, I provide a linguistically principled means of factoring word embeddings playing on an analogy between generative neural networks and generative grammar.

Chapter 4 uses data visualization techniques from the unsupervised learning literature in order to show that BERT models are sensitive to both morphophonological information as well as certain aspects of pragmatic information. To the author's knowledge, this is the first instance of BERTology which examines this type of linguistic information, and excepting Edmiston and Kim (2019), the first work of any kind examining this information in continuous embedding spaces.

Each chapter in this dissertation then seeks to advance BERTology in offering methodological novelties—methods not common in NLP—to either reveal hitherto undiscovered linguistic structure in BERT embeddings or provide means of factoring BERT embeddings such that they're more readily interpretable via disentangled representation learning.

1.5 Outline

Chapter 2 investigates inflectional morphological information contained in the BERT models of five Indo-European languages by running a series experiments which include classifiers and intrinsic dimensionality estimation of the embeddings. Chapter 3 then proceeds to apply variational autoencoders in the semi-supervised setting. This chapter investigates the extent to which BERT-style models' contextual embeddings can be factored into separate morphological, syntactic, and semantic representations, resulting in so-called disentangled representations. The resulting disentangled representations are then tested on downstream morphological, syntactic, and semantic tasks. Chapter 4 shifts gears and investigates affix embeddings from the KoBERT model, a BERT model trained for the Korean language. Affix embeddings in this language are investigated for morphophonological and pragmatic infor-

mation using entirely unsupervised data-visualization methods. Chapter 5 then concludes this dissertation, reflecting on the results of the previous chapters and suggesting future work.

CHAPTER 2

EXAMINING INFLECTIONAL MORPHOLOGY IN BERT MODELS

2.1 Introduction

This chapter describes and reports on experiments designed to probe the contextualized embeddings of BERT models trained on different Indo-European languages. Three experimental methods are run on models to try to answer questions such as (i) what the intrinsic dimensionality of BERT embeddings for different subsets of language is, (ii) how much information in the form of inflectional morphological features can be recovered from BERT's high-dimensional vectors, and (iii) where and how that morphological information is reflected.

The chapter is structured as follows. Section 2.2 introduces the languages, models, and datasets used in the experiments for this chapter. Section 2.3 details experiments which seek to estimate the intrinsic dimensionality for the different point-clouds used. These experiments provide insight into the differing complexities of different subsets of embeddings for words marked for different morphological features, as well as give an idea of the large scale shape of the embeddings as they move through BERT's layers. Section 2.4 recounts experiments wherein two different classifiers are trained on embeddings produced by BERT models to predict different feature values for different inflectional features (e.g. case), examining questions like classifiers' ability to distinguish syncretic forms. Finally, Section 2.5 describes experiments which address similar questions as Section 2.4, but instead of using supervised classifiers, experiments are run using an unsupervised clustering task. Section 2.6 concludes the chapter.

Feature \ Language	English	French	German	Russian	Spanish
Case			$\{Nom, Acc, Dat, Gen\}$	$\{Nom, Acc, Dat, Gen, Loc, Ins\}$	
Gender		$\{Masc, Fem\}$	$\{Masc, Fem, Neut\}$	$\{Masc, Fem, Neut\}$	$\{Masc, Fem\}$
Mood	$\{Ind, Imp\}$	$\{Ind, Sub, Cnd\ Imp\}$	$\{Ind, Sub, Imp\}$	$\{Ind, Cnd, Imp\}$	$\{Ind, Sub, Cnd, Imp\}$
Number	$\{Sing, Plur\}$	$\{Sing, Plur\}$	$\{Sing, Plur\}$	$\{Sing, Plur\}$	$\{Sing, Plur\}$
Person	$\{1, 2, 3\}$	$\{1, 2, 3\}$	$\{1, 2, 3\}$	$\{1, 2, 3\}$	$\{1, 2, 3\}$
Tense	$\{Past, Pres\}$	$\{Past, Pres, Impr, Fut\}$	$\{Past, Pres\}$	$\{Past, Pres, Fut\}$	$\{Past, Pres, Impr, Fut\}$
Verb Form	$\{Fin, Inf, Ger, Part\}$	$\{Fin, Inf, Part\}$	$\{Fin, Inf, Part\}$	$\{Fin, Inf, Part\ Conv\}$	$\{Fin, Inf, Part, Ger\}$
Model	Base-Cased	CamemBERT	DBMDZ	RuBERT	BETO

Table 2.1: Languages and associated features/ values, along with models used. All models are BERT-base models, with 12 hidden layers, 12 attention heads, and 768 dimenional vectors.

2.2 Experiment Preliminaries

2.2.1 Languages and Models

The experiments of this chapter investigate five languages of the Indo-European language family: English, French, German, Russian, and Spanish, each of which inflects for some set of morphological features. Specifically, we investigate the morpohlogical features of Case, Gender, Mood, Number, Person, Tense, and VerbForm (which is related to what is traditionally called finiteness).[1] For each of these languages, we investigate a BERT-base model (Devlin et al., 2019) pre-trained on a large corpus for that language. The languages, models, features, and each feature's values are organized in Table 2.1.[2]

As BERT-base models are a 12-layer transformer architecture (Vaswani et al., 2017), for each word in a sentence the model produces 13 vectors, including the input vector, in 768 dimensions.[3] The experiments of this chapter investigate not only performance across

1. Feature names are written as they appear in standardized Universal Dependency datasets (Nivre et al., 2016), meaning they are capitalized, and VerbForm is written with no intervening space.

2. Abbreviations used in Table 2.1: Nom=Nominative, Acc=Accusative, Dat=Dative, Gen=Genitive, Loc=Locative, Ins=Instrumental, Masc=Masculine, Fem=Feminine, Neut=Neuter, Ind=Indicative, Imp=Imperative, Sub=Subjunctive, Cnd=Conditional, Sing=Singular, Plur=Plural, Pres=Present, Impr=Imperfect, Fut=Future, Fin=Finite, Inf=Infinitive, Ger=Gerund, Part=Participle, Conv=Converb.

3. Strictly speaking, the model produces vectors for each token in a sentence. Word embeddings are derived by taking the average of each word's constituent token embeddings.

Confound\Language	English	French	German	Russian	Spanish
Pct. of ambiguous forms	17.8%	10.0%	26.1%	14.1%	6.0%
Avg. feature-length	2.6	3.0	2.86	3.43	3.17

Table 2.2: Percentage of ambiguous examples across all features for language, as well as average feature length for relevant features.

languages and features, but also performance across layers.

In this chapter, I will make reference to certain aspects of the morphology of the languages investigated here, as they are of particular salience when performing tasks like classification. The two particular aspects are (i) what I will call *feature-length*, defined to be number of possible feature values a feature in a certain language may take, e.g. the feature-length of Case in German is 4, possible values being nominative, accusative, dative, and genitive, and (ii) the amount of ambiguity which exists in the language, as measured by the percentage of surface forms which may take more than one feature value for a particular feature, e.g. *der* in German, which is three-way ambiguous for Case (nominative, dative, genitive). Statistics with regard to these two confounds are in Table 2.2, where the percantage of ambiguous forms is calculated from the datasets used for all experiments in this chapter.

2.2.2 Datasets

All data used for the experiments described in this chapter were collected from Universal Dependency (UD) Treebanks (Nivre et al., 2016), and ambiguity was calculated using UD-compatible lexicons (Sagot, 2018). See Appendix A for specifics on the treebanks used. A dataset is constructed for each language-feature combination in Table 2.1. This entails extracting from treebanks words which are marked for the relevant morphological feature, along with the sentence (i.e. context) in which the word is contained, and the relevant feature value. This results in examples taking the form of triples: (*word*, *sentence*, *value*). For example, a hypothetical entry in the English-Tense dataset would be, (*ran*, "Timmy *ran* in the woods .", Tense: Past). BERT can then be used to embed the word in the context of the sentence, and the resulting embeddings can be used as input to a classifier to predict

the feature value.

For each dataset, 750 examples are drawn for each feature value a feature can take.[4] Thirteen such datasets are created for each language-feature combination, one for each layer of BERT. For the supervised classifiers, I use a .85/.15 train-test split, and reported scores are F1-score on the test set.

2.3 Experiment 1: Intrinsic Dimensionality of BERT Embeddings

2.3.1 Experiment Overview

The first experiment of this chapter deals with the intrinsic dimensionality of BERT-produced embeddings, operating under the assumptions of the manifold hypothesis. The manifold hypothesis has it that much real-world, high-dimensional data is actually situated on relatively low-dimensional manifolds embedded in the high-dimensional space. In the context of the current experiment, the manifold hypothesis would predict that the hidden representations of BERT, while embedded in $\mathbb{R}^{768}$, actually lie on (or near) a low-dimensional manifold. Insights into the shape/dimensionality of this manifold can serve as a first step to better understanding the BERT model. Some requisite definitions follow.

Definition 1. *d*-Dimensional Manifold (Croom, 2016): A topological d-dimensional manifold, or d-manifold, is a second countable Hausdorff space[5] in which each point has a neighborhood homeomorphic to an open set in Euclidean d-space $\mathbb{R}^d$.

The crucial part of this definition for our purposes is that each point has a neighborhood which resembles an open set in $\mathbb{R}^d$, which means we can treat it locally as we would $\mathbb{R}^d$.

4. The only exceptions to this were the Mood feature in French and Spanish, for which there was insufficient data to extract 750 examples of the imperative. As such, the French Mood dataset consisted only of 249 examples for each value, and the Spanish Mood dataset only 381 examples for each value.

5. This means that topological manifolds have a countable dense subset, akin to $\mathbb{Q} \subset \mathbb{R}$, and that any two distinct points can be separated by two non-intersecting open sets containing those points. These are two properties shared with $\mathbb{R}^n$ equipped with the usual topology.

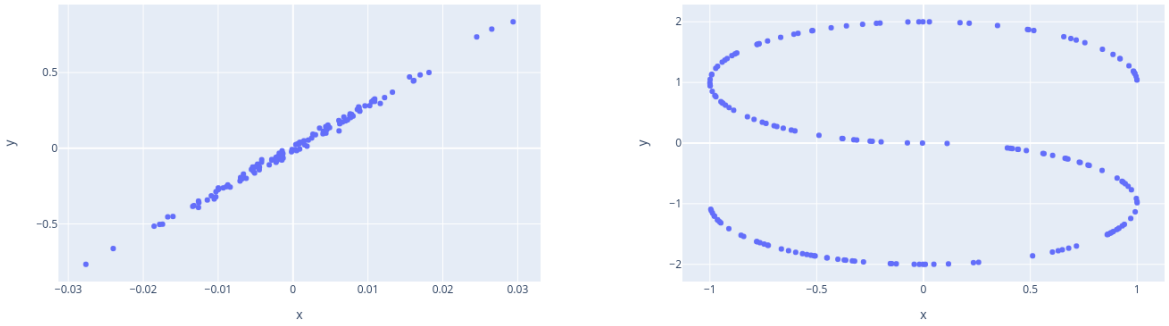
Definition 2. Embedding (Croom, 2016): If X is a topological space which is homeomorphic to subspace $A \subseteq Y$, then X can be embedded in Y. The homeomorphism $f : X \to A$ is called an embedding of X in Y.

Following the manifold hypothesis, we assume that there exists an underlying d-manifold of BERT embeddings, call it $\mathcal{M}_{BERT}$, and the observed contextualized embeddings produced by BERT models in $\mathbb{R}^{768}$ can be described as an embedding from $\mathcal{M}_{BERT}$ into $\mathbb{R}^{768}$. The goal of intrinsic dimensionality estimation is then to recover (or at least approximate) dimensionality d of d-dimensional manifold $\mathcal{M}_{BERT}$. Here, two experiments are run to estimate the intrinsic dimensionality of different point clouds produced by BERT models. These experiments serve two purposes. First, they will give us an idea of the complexity of BERT's embeddings. Second, they will allow us to ask certain linguistic questions related to morphology. For example, consider the respect in which the German definite article '*der*' contains more information than does English's '*the*', in that in addition to the syntactic and semantic information both seem to share, the German variant contains additional morphological information related to gender, case, etc.. I hypothesize that such additional morphological complexity manifests itself as higher intrinsic dimensionality for contextualized embeddings. Results below examine this hypothesis.

Often, estimation of intrinsic dimensionality is done via methods from dimensionality reduction. It is the purpose of dimensionality reduction algorithms to represent points in one space in a relatively low dimensional space, while preserving as much information as possible. We can then consider a dimensionality reduction technique as a function $f : \mathbb{R}^p \to \mathbb{R}^d, d < p$. These functions are learnt by either minimizing a loss function, or alternatively by maximizing variance. The values of loss and variance are then useful for estimating intrinsic dimensionality, as will be made precise below.

Dimensionality reduction techniques can be bifurcated into *linear* methods and *non-linear* methods (Wang, 2012). A linear dimensionality reduction method constrains function $f(\cdot)$ to be a linear projection, while a non-linear dimensionality reduction method has no

such restriction. Linear methods are most appropriate when the underlying manifold upon which the data lie is a linear manifold, or in other words for $f : \mathbb{R}^p \to \mathbb{R}^d$, when the data lie on a d-dimensional affine subspace of $\mathbb{R}^p$. Non-linear methods are appropriate otherwise (Lee and Verleysen, 2007). For intuition, Figure 2.1 shows examples of data embedded in $\mathbb{R}^2$ drawn from underlyingly 1-dimensional manifolds; Figure 2.1a shows data from a linear manifold, and Figure 2.1b shows data from a non-linear manifold.



(a) Data from a linear manifold

(b) Data from a non-linear manifold

Figure 2.1: Examples of data drawn from one-dimensional linear and non-linear manfiolds embedded in $\mathbb{R}^2$.

Interestingly, highly non-linear manifolds[6] can appear as linear manifolds of higher dimensionality. As an example, consider the construction of space-filling curves as in Figure 2.2. Successive iterations of the Hilbert construction (Hilbert, 1935) increasingly fill a two-dimensional space, yet the underlying manifold is one-dimensional. Below, I will hypothesize that the disparity in results between the linear and non-linear dimensionality estimation methods suggests that BERT's underlying manifold is akin to a higher dimensional version of the Hilbert curve, a highly non-linear manifold filling an affine subspace of higher dimension.

6. By *highly non-linear manifold*, I mean manifolds where pairwise distances as defined on the manifold are poorly correlated with the pairwise Euclidean distances in the surrounding space.

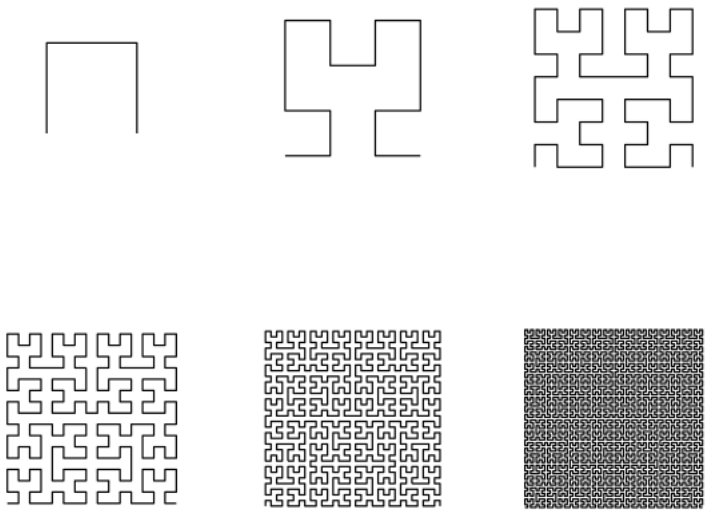


Figure 2.2: Six one-dimensional manifolds (with boundary) embedded in $\mathbb{R}^2$. Images produced by six successive iterations of the Hilbert curve construction (Hilbert, 1935). Image from `https://en.wikipedia.org/wiki/Space-filling_curve`

2.3.2 Method 1: Intrinsic Dimensionality Estimation via PCA

Principal Component Analysis (PCA) (Hotelling, 1933) is a popular method in multivariate data analysis, and is often used for linear dimensionality reduction. Given dataset $X = \{x_1, ..., x_n\}, x_i \in \mathbb{R}^p$, PCA calculates the subspace spanned by d pairwise orthogonal vectors, $d \leq p$, such that projection onto that subspace maximizes variance. The orthogonal vectors which PCA returns are the eigenvectors associated with the d largest eigenvalues of the sample covariance matrix. Projection onto this subspace is then often an effective method of dimensionality reduction, assuming the data are drawn from a linear manifold.

As the eigenvalues of the sample covariance matrix correspond to the variances of the latent variables (see Lee and Verleysen (2007) for a derivation of this), PCA provides a built-in means of determining the goodness-of-fit of the dimensionality-reduced datapoints for dimension $d < p$. If we assume that the true underlying latent variables of BERT embeddings are responsible for, e.g. 90% of the observed variance,[7] then we can select dimension d as the minimum dimension for which 90% of the observed variance is accounted

7. The remaining 10% would then, presumably, be attributable to noise.

for (i.e. where the variance of the projected data is 90% of the observed variance of X). Table 2.3 shows the results of such an experiment for each language-feature combination; estimated dimensions are averaged over all layers of the BERT models, and experiments were run only for multiples of 10.

Feat.\Lang.	English	French	German	Russian	Spanish	Average
Case			300.77	386.15		343.46
Gender		270.0	325.38	360.0	320.0	318.85
Mood	253.85	172.31	200.0	224.62	264.62	223.08
Number	333.08	273.08	310.0	336.15	330.0	316.46
Person	190.0	210.0	270.0	224.62	289.23	236.77
Tense	277.69	247.69	216.15	290.0	297.69	265.85
VerbForm	332.31	262.31	270.77	313.85	333.08	302.46
Average	277.38	239.23	270.44	305.05	305.77	

Table 2.3: Intrinsic dimensionality estimation as calculated by PCA for language-feature combinations, averaged over all 13 layers of BERT base models.

There are two immediate take-aways from the results in Table 2.3. The first is that in each case, BERT-produced point-clouds lie mostly on relatively low-dimensional subspaces of $\mathbb{R}^{768}$. The second take-away is that there is variation between both features and languages. With regard to features, the Mood feature on average requires the fewest dimensions, while the Case feature requires the highest. With regard to language, Russian and Spanish point-clouds require relatively many dimensions, while the French data appear to lie on the lowest dimensional subspace.

Recalling now the intuition that one might expect morphological complexity to be correlated with complexity of point-clouds as measured by intrinsic dimensionality, we are now in a position to test whether this is the case. Here, we measure morphological complexity with feature-length, recalling that this is defined to be the number of feature values a feature can take. Figure 2.3 is a box-plot which organizes the estimates of intrinsic dimensionality by feature length. Though the data is limited in that there are no features of length 5, and only one of length six (Case in Russian), morphological complexity as measured here does not appear to correlate with intrinsic dimensionality estimation.

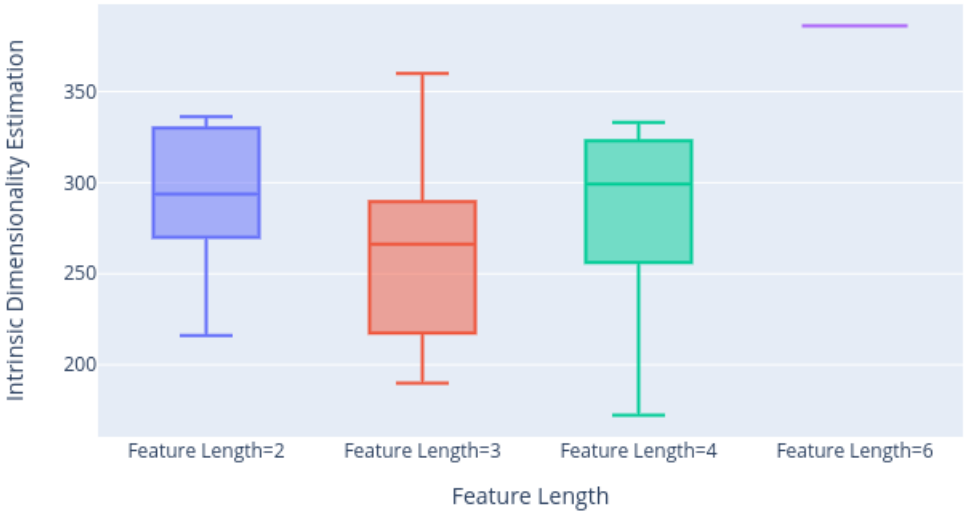


Figure 2.3: Box-plot showing IDE statistics as calculated by PCA for morphological features of different feature-length.

The average intrinsic dimensionality estimation for the different layers of the point-clouds (averaged over all feature sets) can be found in Table 2.4, and the results are visualized in Figure 2.4. Here, the general trend is that intrinsic dimensionality increases from input to roughly layer 9, and then begins to decrease through to the output layer. Russian is an exception, which shows a more variable path through the layers.

2.3.3 Method 2: Intrinsic Dimensionality Estimation via Topology-preserving Autoencoders

PCA above provided estimates for intrinsic dimensionality under the stringent assumptions that (i) the underlying latent variables are uncorrelated, and (ii) the underlying space is linear. These assumptions, however, are unlikely to hold for BERT embeddings, as the generating mechanism is a non-linear function parameterized by a deep neural network. One

Layer\Lang.	English	French	German	Russian	Spanish	Average
Input	200.0	203.33	211.43	318.57	255.0	237.67
1	208.0	231.67	222.86	302.86	261.67	245.41
2	236.0	238.33	222.86	285.71	276.67	251.91
3	258.0	245.0	234.29	285.71	286.67	261.93
4	264.0	250.0	247.14	281.43	301.67	268.85
5	276.0	253.33	265.71	288.57	311.67	279.06
6	290.0	260.0	291.43	307.14	316.67	293.05
7	302.0	266.67	297.14	320.0	326.67	302.5
8	314.0	266.67	307.14	332.86	331.67	310.47
9	324.0	256.67	308.57	330.0	336.67	311.18
10	330.0	245.0	302.86	320.0	335.0	306.57
11	326.0	230.0	307.14	311.43	321.67	299.25
12	278.0	163.33	297.14	280.0	313.33	266.36
Average	277.38	239.23	270.44	304.95	305.77	

Table 2.4: Intrinsic dimensionality estimation over layers as calculated by PCA, scores averaged over each feature.

can therefore take the figures in Tables 2.3 and 2.4 as providing soft upper-bounds, and we expect the true dimensionality of our hypothesized manifolds to be lower.

We therefore need a less restrictive means of estimating intrinsic dimensionality which drops the restriction of underlying latent variables being uncorrelated and the underlying manifold being linear. In other words, we need a function which projects onto the new lower-dimensional coordinate system which is not constrained to be linear and allows for correlations between variables. Neural networks are an appropriate framework for doing this, as they are capable of modeling highly non-linear functions. We thus assume nothing about the underlying manifold except that local pairwise geodesic distances are in correspondence with Euclidean distances, allowing us to model distance along the manifold with the standard Euclidean distance metric for sufficiently local points.

The method I use here is in the spirit of the SAMANN, or Sammon Artificial Neural Network (De Ridder and Duin, 1997), adopting the loss function of Potapov and Ali (2002). This means that given observed BERT embeddings $X = \{x_1, \ldots, x_n\}, x_i \in \mathbb{R}^p$, the goal is to learn a mapping $f : \mathbb{R}^p \to \mathbb{R}^d, d < p$, such that $d(x_i, x_j) \approx d(r(x_i), r(x_j))$ for points

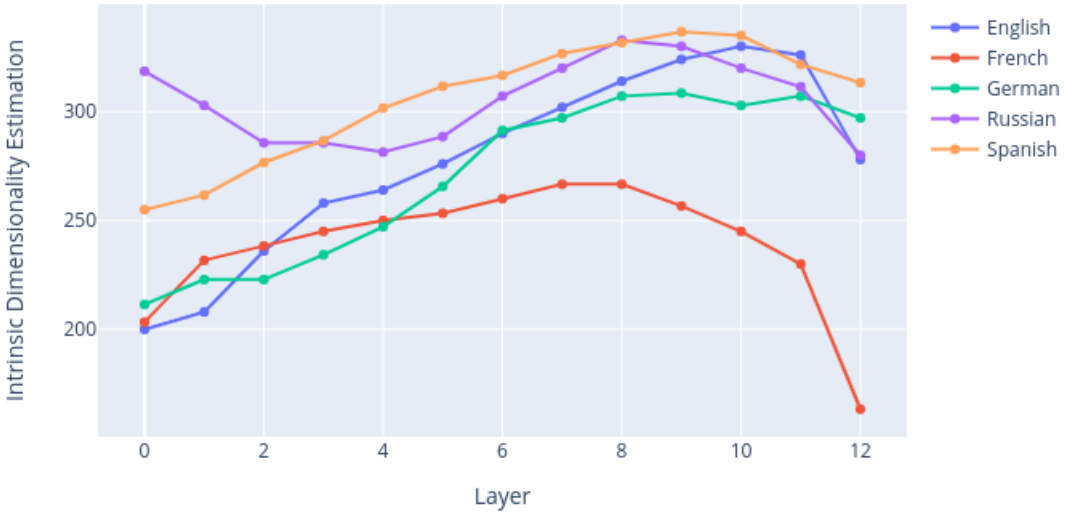


Figure 2.4: Intrinsic dimensionality estimates as calculated by PCA for layers (averaged over feature point-clouds) of different BERT models.

$x_i, x_j \in X$ which are sufficiently close. Here, x_i and x_j are defined to be suffiently close when x_j is one of the k-nearest neighbors of x_i, where $k = 5$.

In this framework, dimensionality reducing function $f(\cdot)$ is similar to the encoder portion of an autoencoder architecture, however the objective is different. In a standard autoencoder, the encoder is encouraged to compress datapoints such that a minimum of information is lost, thus allowing a decoder to then recreate the original datapoints. Here, the objective seeks to preserve pairwise distances from the original space in a non-linear projection into the lower-dimensional space. Call a neural network trained to preserve distances in this way a Topology-preserving autoencoder (TPAE). Its objective function is as defined in Equation 2.1 (Potapov and Ali, 2002), where $\hat{x_i} = f(x_i)$, or the encoding of x_i by the TPAE.

$$J(X;\theta) = \frac{1}{k \cdot |X|} \sum_{(x_i,x_j) \in X^{k_{NN}}} \left(\frac{\|x_i - x_j\|}{\|\hat{x}_i - \hat{x}_j\|} + \frac{\|\hat{x}_i - \hat{x}_j\|}{\|x_i - x_j\|} \right) \tag{2.1}$$

Here, $X^{k_{NN}}$ is defined as the dataset $\{(x_i, x_j) \mid x_i \in X, x_j \in k_{NN}(x_i)\}$. That is, $X^{k_{NN}}$ is the set of tuples such that x_i is a datapoint in X, and x_j is one of the k-nearest neighbors of x_i. This satisfies the aforementioned restriction that only sufficiently local distances are being considered.

The definition of $f(\cdot)$ as used in these experiments is given in Equation 2.2. That is, $f(\cdot)$ is a two-layer neural network, where the first layer is an affine transformation with parameters $W_1 \in \mathbb{R}^{960\times768}, b_1 \in \mathbb{R}^{960}$, followed by a *tanh* activation function.[8] The second layer is likewise an affine transformation, with parameters $W_2 \in \mathbb{R}^{d\times960}, b_2 \in \mathbb{R}^{d}$. There is no activation function following this second affine layer.

$$f(x) = W_2\big(tanh(W_1(x) + b_1)\big) + b_2 \tag{2.2}$$

For PCA, the amount of variance captured was used to measure the goodness-of-fit when reducing dimensionality. Here, I use the "elbow" heuristic, which uses the value of the loss function at differing dimensions to measure goodness-of-fit. Intrinsic dimensionality can then be estimated when goodness-of-fit ceases to increase significantly with dimensionality, or equivalently, when loss ceases to significantly decrease with dimensionality. Consider Figure 2.5. Here, loss is monotonically decreasing for dimensions 10 through 50, however between dimension=50 and dimension=60 it stops decreasing, and even begins to increase.[9] Dimension=50 is then considered an elbow, and is our estimate for intrinsic dimensionality. The intuition is that by affording the autoencoder more dimensions, it continues to better represent the original data in the lower dimensional space. However, at some point the autoencoder no longer improves, presumably because the low-dimensional space is sufficiently large to fully encode the data, making this elbow-point a good estimate for the intrinsic

8. *tanh* is a non-linear function defined as $tanh(x) = \frac{exp(x)-exp(-x)}{exp(x)+exp(-x)}$. When performed on a vector as in a neural network, the function is performed pointwise.

9. While it's counterintuitive that the loss could actually increase with increasing dimensionality, it is possible here due to the stochasticity of initializing neural networks. Stochasticity in this case cannot be controlled for as the TPAE will have a differing architecture for each dimensionality.

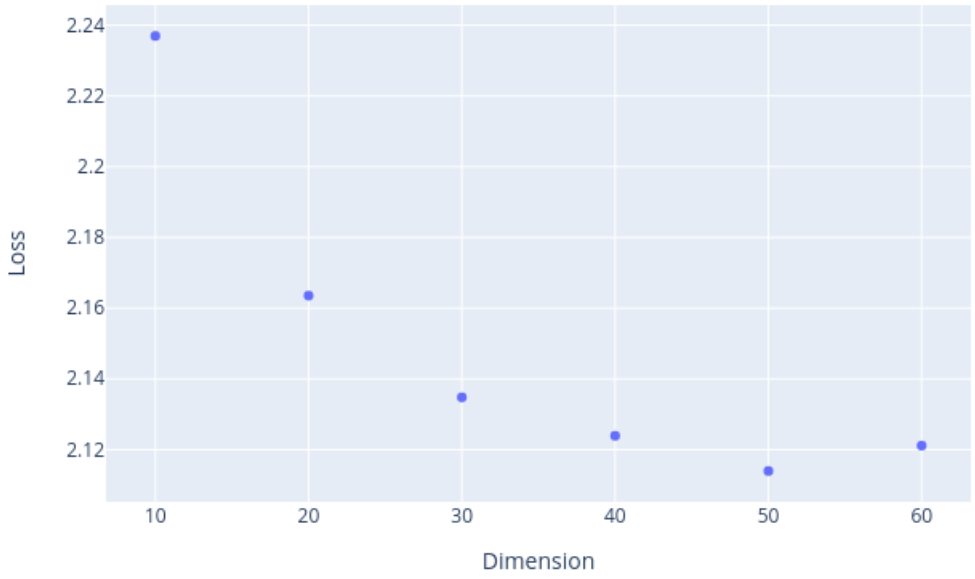


Figure 2.5: Plot of loss against dimension using topology-preserving autoencoder. Elbow at Dimension=50. Point-cloud taken from layer 12 of the English-Mood dataset.

dimension of the underlying manifold. The results for an experiment in which intrinsic dimensionality is estimated in such a way, incrementing by intervals of 10, is in Table 2.5.

Feat.\Lang.	English	French	German	Russian	Spanish	Average
Case			43.85	40.77		42.31
Gender		66.15	50.77	55.38	50.77	55.77
Mood	47.69	59.23	44.17	43.33	51.54	49.19
Number	58.46	76.15	49.23	56.15	49.23	57.85
Person	43.33	48.46	39.23	48.46	45.38	44.97
Tense	56.15	46.15	50.77	50.77	44.62	49.69
VerbForm	40.77	54.62	43.08	43.08	44.62	45.23
Average	49.28	58.46	45.87	48.28	47.69	

Table 2.5: Intrinsic dimensionality estimation for language-feature combinations as measured by a TPAE, averaged over all 13 layers of BERT base models.

Here again, we see intrinsic dimensionality as estimated to be far lower than the embedding space's dimensionality of 768, with intrinsic dimensionality in no case even approaching 100 dimensions. The manifold hypothesis, i.e. that high-dimensional data are drawn from

a relatively low-dimensional manifold, appears to hold in this case and the original high-dimensional data points can be faithfully represented in a lower dimensional space. Furthermore, note that the estimates for dimensionality as produced by the non-linear TPAE are far lower than the estimates produced by PCA (cf. Table 2.3). I hypothesize that this result indicates that $\mathbb{M}_{BERT}$ appears as a high-dimensional analogue to the space-filling curve visualized in Figure 2.2, with $\mathbb{M}_{BERT}$ being a non-linear manifold of dimension d which fills an affine space of dimension $e > d$.

Figure 2.6, to be compared with Figure 2.3, shows again that the number of values a feature can take does not seem to be (positively) correlated with the intrinsic dimensionality of point-clouds of word-embeddings marked for that feature. The layer-wise results are amassed in Table 2.6, with visualization in Figure 2.7. Contrary to the layer-wise estimates produced by PCA, which showed a general upward trend moving through the layers before again descending, the estimates produced by TPAE show a general downward trend from the early to mid-layers before eventually stabilizing.

Layer\Lang.	English	French	German	Russian	Spanish	Average
Input	65.0	81.67	61.67	58.33	60.0	70.83
1	60.0	53.33	52.86	50.0	53.33	53.9
2	50.0	66.67	52.86	48.57	51.67	53.95
3	48.0	58.33	52.86	48.57	48.33	51.22
4	48.0	58.33	47.14	45.71	48.33	49.5
5	44.0	55.0	42.86	38.57	45.0	45.09
6	46.0	58.33	40.0	47.14	43.33	46.96
7	46.0	53.33	41.43	40.0	45.0	45.15
8	40.0	56.67	47.14	40.0	41.67	45.1
9	42.0	51.67	40.0	41.43	43.33	43.69
10	44.0	58.33	45.71	48.57	45.0	48.32
11	40.0	56.67	37.14	47.14	43.33	44.86
12	48.0	60.0	42.86	45.71	40.0	47.31
Average	47.77	59.1	46.5	46.13	46.79	

Table 2.6: Intrinsic dimensionality estimation over layers as calcuated by TPAE, scores averaged over each feature.

This first experiment into the point-clouds associated with different morphological fea-

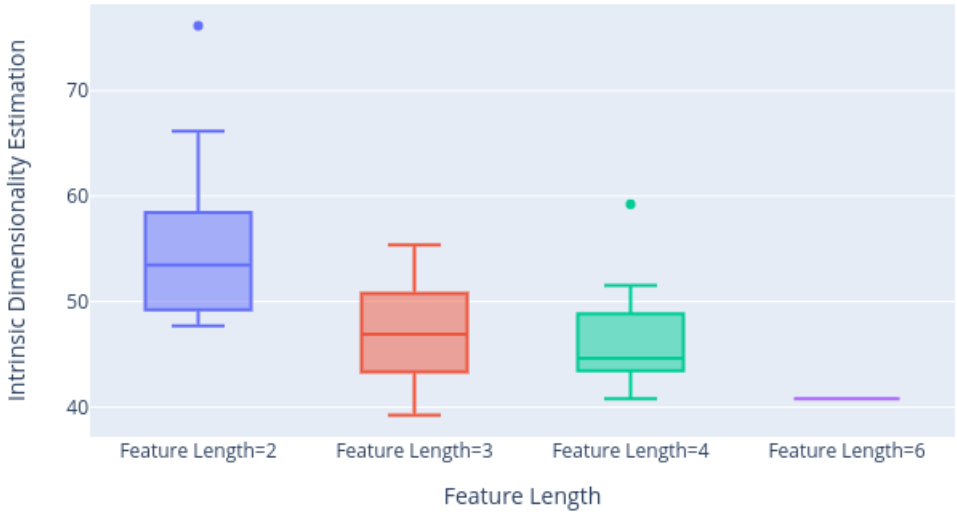


Figure 2.6: Box-plot showing IDE statistics as calculated by TPAE for morphological features of different feature-length.

tures then provides some insight into the embeddings investigated in the rest of this chapter. First, the embeddings in all cases appear to lie on low-dimensional manifolds relative to the 768 Euclidean space they are embedded in. Furthermore, it does not appear to be the case that the intrinsic dimensionality of these word embedding point-clouds is positively correlated with the complexity of the morphological features the words are marked for.

2.4 Experiment 2: Classifiers on Embeddings

2.4.1 Experiment Overview

Given the contextualized word embedding point-clouds described in Section 2.2.2, this experiment seeks to test the amount of morphological information in the point-clouds with classifiers.[10] The experiments described in this section amount to n-way classification tasks

10. The experiments described in this section were taken from Edmiston (2020)

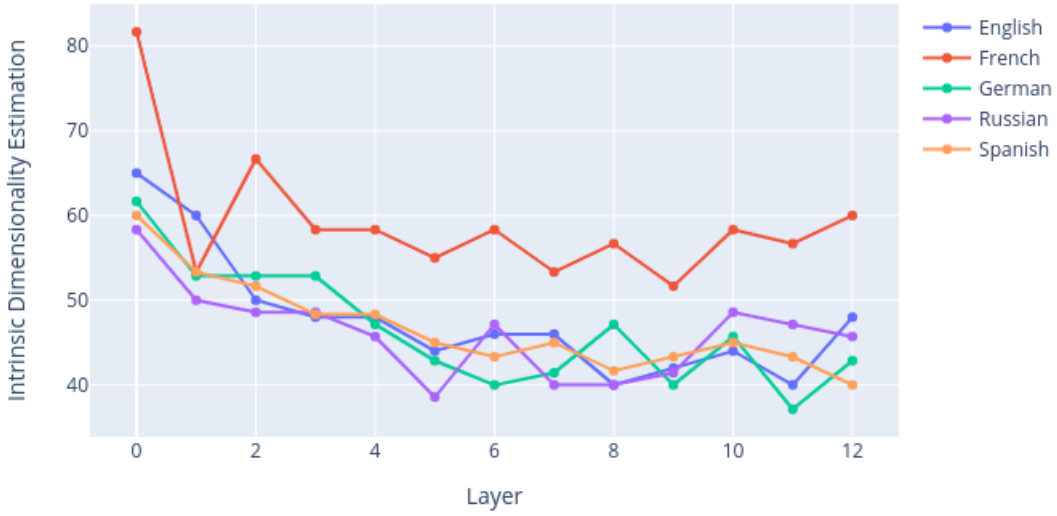


Figure 2.7: Intrinsic dimensionality estimates for layers as calculated by TPAE (averaged over feature point-clouds) of different BERT models.

for each language-feature pair, where n is the number of feature values that the relevant feature can take (e.g. $n = 3$ for Mood in German, values being indicative, imperative, and subjunctive).

The classification tasks are done with a linear classifier and a non-linear classifier, the latter of which consists of a 3-layer neural network with ReLU-activations in hidden layers. The linear classifier allows us to judge the extent to which the embeddings are linearly separable according to morphological feature value. In addition to answering the question of how much morphological information is reflected in BERT's embeddings, the linear restriction goes some way to answering how that information is represented. Specifically, assuming success of a linear classifier on delineating feature value, one can conclude that BERT's embedding space can be partitioned into convex subregions according to feature value. The non-linear neural network classifier does little to answer the question of the shape of the data, as such models are often understood as blackboxes. Nevertheless, neural network clas-

sifiers provide more discriminative capacity than linear classifiers, and are in many cases more successful than their linear counterparts.

In order to get a sense of how much BERT's pre-training on large corpora is aiding in its apparent acquisition of linguistic information, many of the experiments are also run with a random baseline, using a randomly initialized BERT model with no pre-training to embed the same datasets.

As in Section 2.3, results are presented from two different perspectives. The first explores the performance of classifiers for each language relative to feature. This should provide insight into how well certain aspects of morphological information are captured relative to one another. The second explores the performance of classifiers for each language relative to depth of embedding. Do embeddings from relatively deep layers disambiguate features better than those from relatively shallow layers? This goes some way to answering the question of how much context aids in differentiating between different morphological features, especically in cases where syncretism is involved. As above, we test the effect that feature-length has on performance, and in addition test the effect that syncretism has.

2.4.2 Classifier 1: Linear Classifier

For the first classifier, I use the LogisticRegression class avaiable in the *SKLearn* toolkit (Pedregosa et al., 2011), which provides a standard linear model with which to run n-way classification tasks. All scores are related as F1-scores.

The results in Table 2.7 are highly suggestive that BERT models trained with their usual Masked Language Modeling task are sensitive to inflectional morphology, that is, contain information in their embeddings which is useful for determining feature value.[11] Furthermore, the fact that such strong results are coming from a linear classifier means that BERT's embedding spaces can be partitioned into convex sub-regions in a manner which highly correlates with morphological feature value.

11. See Table B.1 in Appendix B for results of this experiment using a random baseline.

Feat.\Lang.	English	French	German	Russian	Spanish	Average
Case			0.9	0.84		0.87
Gender		0.96	0.87	0.84	0.99	0.92
Mood	0.98	0.95	0.91	0.99	0.89	0.94
Number	0.97	0.99	0.92	0.93	0.99	0.96
Person	0.98	0.98	0.95	1.0	0.94	0.97
Tense	0.99	0.99	0.95	0.96	0.98	0.97
VerbForm	0.88	1.0	0.98	0.99	0.98	0.97
Average	0.96	0.98	0.93	0.94	0.96	

Table 2.7: F1 scores for features using linear classifier.

Recalling the two issues of feature-length and syncretism, one would expect both to be negatively correlated with classification performance. In the case of feature-length, clearly n-way classification becomes a more difficult task as n grows larger. Syncretism also poses a problem for classification, forcing the classifier to rely on the surrounding contextual information to disambiguate any forms. Table 2.8 shows the correlations between weighted F1 scores of the linear classifier and the feature-length and percent of syncretic forms.

Language	English		French		German		Russian		Spanish	
Variable	Perc.	FL	Perc.	FL	Perc.	FL	Perc.	FL	Perc.	FL
Correlation	-0.87	-0.88	-0.94	-0.12	-0.64	-0.30	-0.81	-0.42	0.13	-0.44

Table 2.8: Pearson correlation scores between performance on feature (as measured by the linear classifier in Table 2.7) and percentage of ambiguous entries/number of possible values for feature. Light blue indicates statistically significant with p-value below 0.1, Dark blue indicates statistically significant with p-value below 0.05.

As would be expected, with the exception of Spanish syncretic forms (for which there are very few; see Table 2.2), all weighted F1-scores reported for the linear classifier are negatively correlated with the confounds of feature-length and percentage of ambiguous forms. In four cases, this correlation is statistically significant.

Figure 2.8: Visualization of layer-wise performance of the linear classifier on different subsets of the German Case point-cloud.

As mentioned before, classification of ambiguous forms must rely on contextual information in the embeddings. As it is thought that the sort of syntactic context necessary for disambiguation is present in the middle-to-late layers in BERT-style models (Tenney et al., 2019), one would expect performance on ambiguous forms to increase through the layers. This is indeed what we find in the case of German Case and Russian Case, where certain forms are up to 4-way ambiguous and 5-way ambiguous respectively. The results of F1 score on different subsets of the German Case point-cloud and Russian Case point-cloud are visualized in Figures 2.8-2.9.

In both cases, while unambiguous forms (i.e. Ambiguity=1 forms) see strong performance throughout the layers, ambiguous forms all witness marked improvement through the layers. This suggests that BERT does host the necessary contextual information in its embeddings for the disambiguation of syncretic forms, and the classifier is able to recognize and leverage this information. Note however, that while native-level Russian and German

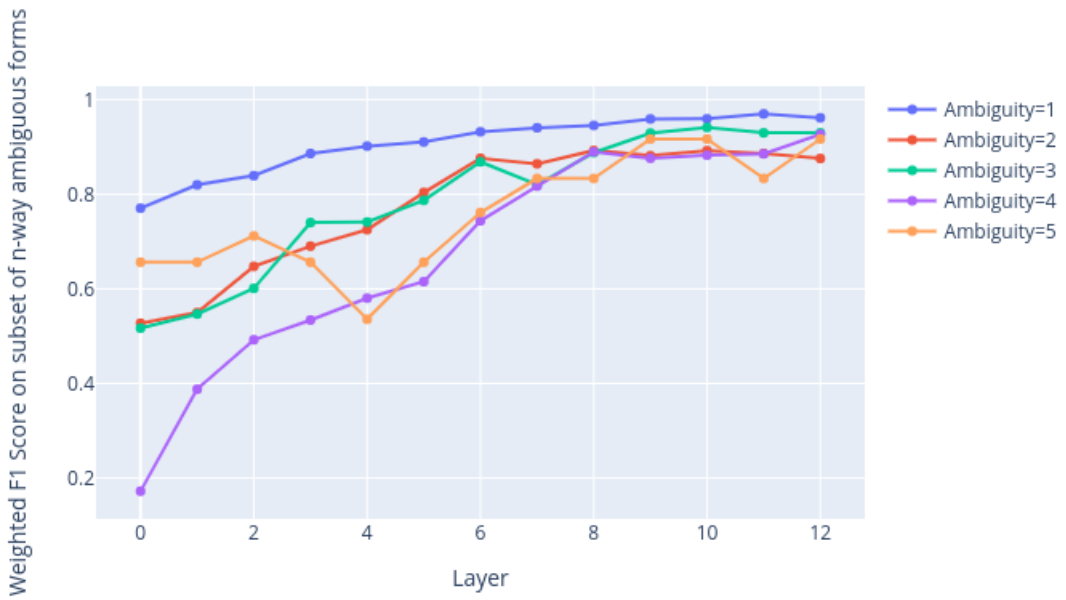


Figure 2.9: Visualization of layer-wise performance of the linear classifier on different subsets of the Russian Case point-cloud.

speakers disambiguate syncretic forms with apparent ease, the classifier is not able to do so in all cases, specifically, see the case of Ambiguity=4 in for the German Case feature.

The Layer-wise performance of the linear classifier can be found in Table 2.9,[12] and can be visualized in Figure 2.10. With regard to the layer-wise results, again we see that the classifier performs well on virtually every point-cloud subset, though there is apparent improvement moving through the layers, especially in the cases of German and Russian.

I speculate that the relatively strong performance of the linear classifier on the English, French, and Spanish point-clouds is a result of the simple morphological paradigms these languages possess relative to German and Russian. For example, while features such as gender are more often than not determinable from orthography in French, this is less so the case with German. Moving through the layers as more contextual information becomes available in the embeddings, the scores for German and Russian approach those of English,

12. Compare against Table B.2 in Appendix B.

Layer\Lang.	English	French	German	Russian	Spanish	Average
Input	0.93	0.95	0.85	0.88	0.94	0.91
1	0.93	0.97	0.88	0.9	0.96	0.93
2	0.95	0.98	0.9	0.92	0.96	0.94
3	0.96	0.99	0.91	0.93	0.96	0.95
4	0.96	0.98	0.92	0.94	0.96	0.95
5	0.97	0.98	0.94	0.95	0.97	0.96
6	0.97	0.99	0.95	0.95	0.97	0.96
7	0.97	0.98	0.95	0.95	0.96	0.96
8	0.97	0.98	0.95	0.95	0.97	0.96
9	0.97	0.98	0.95	0.95	0.97	0.96
10	0.96	0.98	0.94	0.95	0.96	0.96
11	0.96	0.98	0.94	0.95	0.95	0.96
12	0.96	0.98	0.94	0.95	0.95	0.96
Average	0.96	0.98	0.93	0.94	0.96	

Table 2.9: Layer-wise performance of linear classifier.

French, and Spanish.

2.4.3 Classifier 2: Non-linear Classifier

For the non-linear classifier, I used the *skorch* package[13] to implement a three-layer neural network with ReLU activation functions on the two hidden layers, where the hidden layers are of dimensionality 768. The results for language-feature pairs averaged over layers are in Table 2.10.

Feat.\Lang.	English	French	German	Russian	Spanish	Average
Case			0.86	0.85		0.85
Gender		0.96	0.87	0.84	0.99	0.91
Mood	0.97	0.92	0.91	0.99	0.87	0.93
Number	0.94	0.99	0.92	0.93	0.99	0.95
Person	0.97	0.98	0.95	0.99	0.94	0.96
Tense	0.98	0.99	0.94	0.96	0.98	0.97
VerbForm	0.88	1.0	0.97	0.99	0.98	0.96
Average	0.95	0.97	0.92	0.94	0.96	

Table 2.10: Weighted F1 scores for features using non-linear classifier.

13. Available at https://github.com/skorch-dev/skorch.

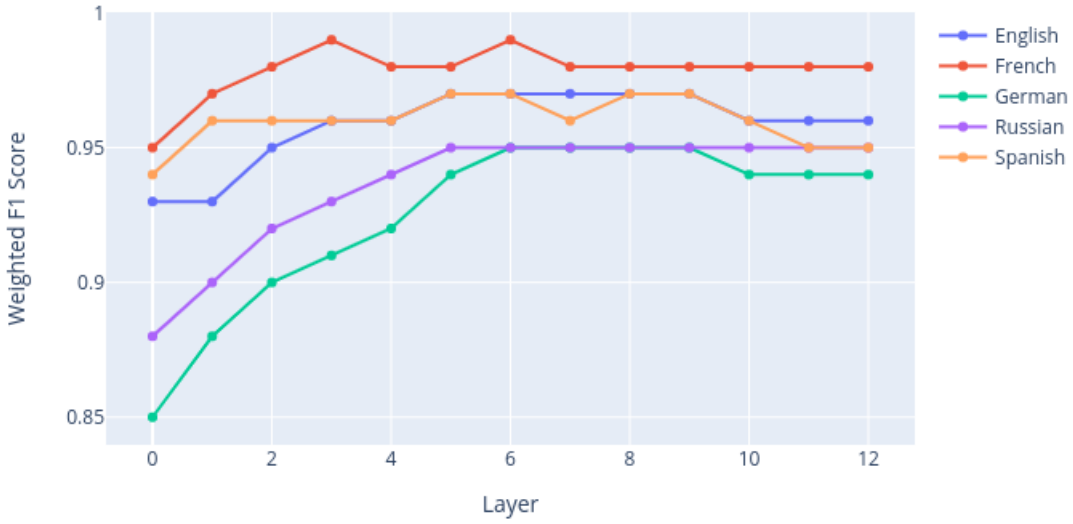


Figure 2.10: Visualization of layer-wise performance of linear classifier.

Here again we see strong results throughout the table, but what is most interesting is not that it performed well, but that on average it performed no better (worse, in fact) than the linear classifier. It then appears that a linear classifier is sufficient in virtually all cases to delineate morphological feature value for BERT embeddings.

The layer-wise scores for the non-linear classifier are as in Table 2.11, with its visualization as in Figure 2.11. The non-linear classifier shows more variance through the layers, but again shows that English, French, and Spanish perform better than do Russian or German. Also peculiarly, Layer 7 shows a dip for each language except Russian. In the end, the application of the non-linear classifier tells us little that the linear probe did not, other than reinforcing the apparent truth that BERT's embedding space can be partitioned into convex sub-regions correlated with feature value. BERT models appear strongly sensitive to morphological information at the level of inflectional features, and performance generally gets better through the layers.

Layer\Lang.	English	French	German	Russian	Spanish	Average
Input	0.93	0.95	0.85	0.9	0.93	0.91
1	0.94	0.97	0.88	0.89	0.96	0.93
2	0.96	0.98	0.9	0.92	0.94	0.94
3	0.96	0.98	0.9	0.93	0.97	0.95
4	0.96	0.98	0.93	0.94	0.95	0.95
5	0.96	0.98	0.95	0.95	0.96	0.96
6	0.96	0.98	0.95	0.95	0.96	0.96
7	0.85	0.93	0.94	0.95	0.95	0.92
8	0.97	0.98	0.93	0.95	0.96	0.96
9	0.96	0.98	0.95	0.95	0.96	0.96
10	0.96	0.98	0.94	0.95	0.95	0.96
11	0.96	0.98	0.86	0.94	0.95	0.94
12	0.96	0.98	0.92	0.95	0.95	0.95
Average	0.95	0.97	0.92	0.94	0.95	

Table 2.11: Layer-wise performance of non-linear classifier.

2.5 Experiment 3: Deep Embedding for Clustering Embeddings

2.5.1 Experiment Overview

While Experiment 2 in Section 2.4 dealt with supervised approaches to discovering inflectional morphological information in BERT's embeddings, here I focus on an unsupervised method of discovering that information. Specifically, I use the Deep Embedding for Clustering (DEC) method of Xie et al. (2016). As is, most clustering methods such as K-means clustering attempt to partition data directly in the original feature space.[14] The DEC method discussed here attempts to learn a lower dimensional feature representation of the data via a neural network while at the same time clustering the data in that reduced space.

Consider the problem as having n datapoints $X = \{x_1, ..., x_n\}$ such that we wish to partition X into k clusters. Rather than clustering directly in the space in which X is embedded, first transform the datapoints into a latent feature space via parameterized function f_θ; denote by $Z = \{z_1, ..., z_n\}$ the image of X under f_θ, i.e. $f_\theta(X) = Z = \{f_\theta(x_1), ..., f_\theta(x_n)\}$.

14. A notable exception is spectral clustering techniques (Von Luxburg, 2007), which first embed data points via a spectral embedding method (Belkin and Niyogi, 2003) and then cluster the dimensionality reduced data via a traditional method such as K-means clustering.

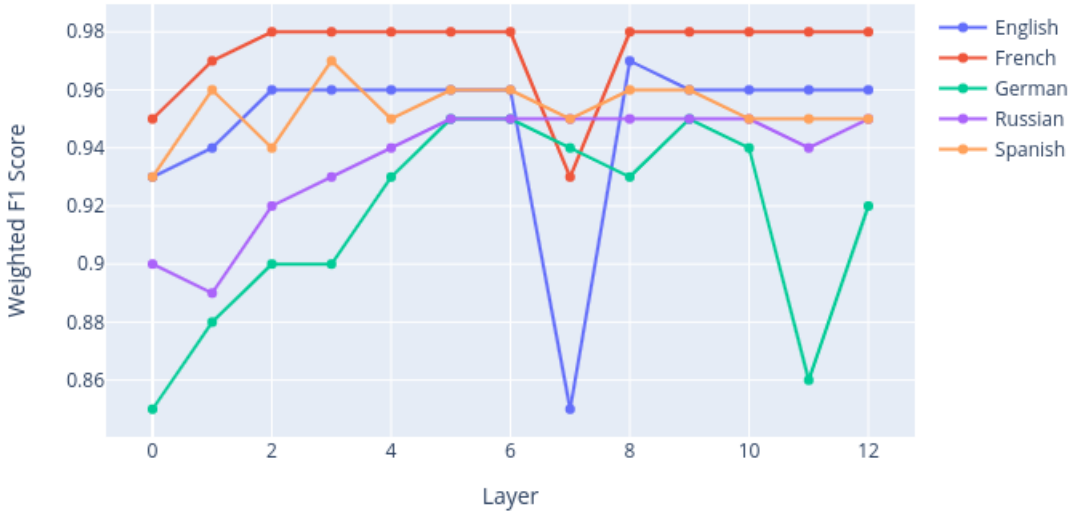


Figure 2.11: Visualization of layer-wise performance: Non-linear classifier.

The goal then is to cluster Z into k clusters, which is done via a K-means-like task, wherein we also learn k cluster centers $\{\mu_1, ..., \mu_k\}$. The method leading to the partition is a two-step process.

The first step of the process is initialization of the dimension-reducing function f_θ. This is done via stacked (denoising) autoencoders (SAE) (Vincent et al., 2010), which have been shown to induce well-separated representations likely to be helpful for a clustering task. A denoising autoencoder is a two-layer neural network wherein the first layer is called the encoder, and the second is called the decoder. It is defined in the following way, with $\mathbf{x}$ as input, *Dropout* an element-wise function which maps entry x_i to 0 with a 20% probability, otherwise returning the entry unchanged, W_i are weight matrices, and $\mathbf{b}_i$ are bias vectors.

$$\tilde{\mathbf{x}} \sim Dropout(\mathbf{x}) \tag{2.3}$$

$$\mathbf{h} = g_1(W_1\tilde{\mathbf{x}} + \mathbf{b}_1) \tag{2.4}$$

$$\tilde{\mathbf{h}} \sim Dropout(\mathbf{h}) \tag{2.5}$$

$$\hat{\mathbf{x}} = g_2(W_2\tilde{\mathbf{h}} + \mathbf{b}_2) \tag{2.6}$$

Denoising autoencoders are trained to minimize the mean squared error between input and output pairs, i.e. their loss function is as follows.

$$L(X, \theta) = \frac{1}{|X|} \sum_i \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|^2 \tag{2.7}$$

Here, $\mathbf{x}_i$ is some input example, $\hat{\mathbf{x}}_i$ the output of $decoder(encoder(\mathbf{x}_i))$, and θ the parameters of the system. Stacked autoencoders are a multi-layer autoencoder architecture in which the output of a composition of encoders $encoder_\ell(...encoder_1(\cdot))$ is fed to a composition of decoders in the opposite order, $decoder_1(...decoder_\ell(\cdot))$.[15] After training each layer of the SAE individually, the final stacked autoencoder is trained as a whole. After training, the decoder portion of the architecture is discarded, and the composition of encoders is the initialization of the dimensionality reducing function f_θ. Given initialized f_θ, standard K-means is run on $f_\theta(X)$, and the output centroids are the initialization of $\{\mu_1, ..., \mu_k\}$. Both representation function f_θ and centroids $\{\mu_1, ..., \mu_k\}$ are optimized in step 2 of DEC.

Much as in K-means clustering, the partitioning of data points in DEC is a function of the distance between a datapoint and its distance to the k cluster centers. The second step of DEC is the process whereby the partitioning of datapoints is optimized. Denote by $q_{i,j}$ the probability that datapoint i belongs to cluster j, and calculate it as follows.

$$q_{i,j} = \frac{(1 + \|\mathbf{z}_i - \boldsymbol{\mu}_j\|^2)^{-1}}{\sum_{j'}(1 + \|\mathbf{z}_i - \boldsymbol{\mu}_{j'}\|^2)^{-1}} \tag{2.8}$$

$q_{i,j}$ is iteratively updated to more closely approximate target distribution values $p_{i,j}$ via the minimization of KL divergence between the two, i.e. DEC uses as loss function the

15. The specifics of the architecture and training I employed for this experiment differ slightly from the one described in Xie et al. (2016), most notably in that I found better results using a two-layer SAE rather than four-layer SAE, and I do not normalize all vectors to unit length.

following.

$$L = KL(P\|Q) = \sum_i \sum_j p_{i,j} \log \frac{p_{i,j}}{q_{i,j}} \tag{2.9}$$

The target distribution values $p_{i,j}$ are calculated as in Equation 2.10, and are chosen such that they (i) improve cluster purity, (ii) emphasize data points assigned with high confidence, and (iii) normalize the influence of centroids such that large clusters do not dominate the update procedure. In Equation 2.10, $f_j = \sum_i q_{i,j}$, or the soft cluster frequency, the sum of probability mass allotted to class j over all nodes.

$$p_{i,j} = \frac{q_{i,j}^2 / f_j}{\sum_{j'} q_{i,j'}^2 / f_{j'}} \tag{2.10}$$

Hard clusters are obtained by taking the argmax of the soft clusters, and partitions are evaluated using the adjusted mutual information (AMI; Vinh et al. 2010) between the output hard clustering and the ground-truth partitions of morphological feature value. Identical partitions return a score of 1, while random partitions have an expected AMI of 0 (with negative scores possible). Each dataset is run for 5 epochs, and the output of the best epoch is used as the output clustering of DEC. Given manual inspection of multiple datasets, the algorithm usually converges before 5 epochs.

2.5.2 Results

Results for the unsupervised clustering results across feature are found in Table 2.12. These scores appear quite low compared with the results from the previous section, but it must be kept in mind that F1-scores and AMI scores are not directly comparable. As I hope to show in this section, the unsupervised method presented here provides a complementary view to the classifiers in the previous section, rather than a competing one. When considering the results in this section, it is most useful to compare them directly against the random baselines, which in this case are found in Table 2.13. See Figure 2.12 for a visualization

Feat.\Lang.	English	French	German	Russian	Spanish	Average
Case			0.0105	0.0088		0.0097
Gender		0.0071	0.0019	0.006	0.0041	0.0048
Mood	0.0774	0.413	0.1117	0.3075	0.1328	0.2085
Number	0.0127	0.0053	0.0069	0.0073	0.0037	0.0072
Person	0.1122	0.1183	0.0757	0.0468	0.0499	0.0806
Tense	0.018	0.3337	0.0131	0.0218	0.0659	0.0905
VerbForm	0.0367	0.1582	0.027	0.0493	0.0834	0.0709
Average	0.0514	0.1726	0.0353	0.0639	0.0566	

Table 2.12: Adjusted mutual information scores for features using deep embedding for clustering classification.

which reflects the stark contrast in performance.

Feat.\Lang.	English	French	German	Russian	Spanish	Average
Case			0.0101	0.0009		0.0055
Gender		0.0036	0.0017	0.0013	0.0181	0.0062
Mood	0.0051	0.0775	0.0193	0.0614	0.0066	0.034
Number	0.0002	0.0051	0.0011	0.0014	0.0015	0.0019
Person	0.0385	0.011	0.0172	0.0059	0.0039	0.0153
Tense	0.0064	0.0045	0.0005	0.0022	0.0029	0.0033
VerbForm	0.0018	0.0031	0.0009	0.0015	0.0004	0.0015
Average	0.0104	0.0175	0.0073	0.0106	0.0056	

Table 2.13: Adjusted mutual information scores for features using random baseline.

Compare Figure 2.12 with Figure 2.13, which plots the comparison of F1-scores from the linear model trained in Section 2.4. Thus, while Section 2.4 revealed that inflectional morphological feature values are almost perfectly recoverable by supervised methods, the unsupervised method of this section provides a better reflection of the importance pre-training plays in BERT's encoding of linguistic features.

The layer-wise results for BERT on the unsupervised DEC task are presented in Table 2.14, with the random baseline following in Table 2.15. Visualization is found in Figure 2.14. Here again, in all cases (with the exception of certain mid-late layers in English) the trained model performs far better than the random model, and the unsupervised task is shown to reveal that trained BERT models display a non-trivial amount of morphological

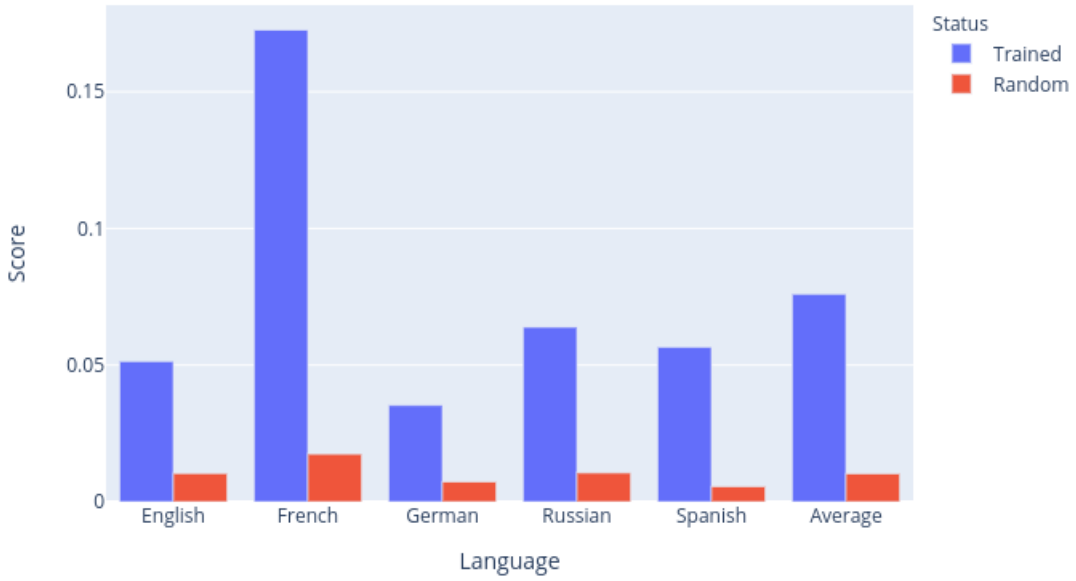


Figure 2.12: Comparison of AMI scores for trained BERT models vs. random baselines, results averaged over features and layers

information. However, counter to what was seen in the classifiers task where performance was maximized in the middle-to-late layers, here we see that most languages perform best in the early-to-mid layers.

This does not necessarily mean that the early-to-mid layers of BERT models are best for extracting morphological information, as it could simply be the relative amount of morphological information is maximized at this stage, and synctactic/semantic information "takes over" in later stages, which would be in line with the findings of Tenney et al. (2019). Without the benefit of supervision, the morphological information necessary for partitioning by feature value may be drowned out in later layers, resulting in the relatively strong showing in the early layers.

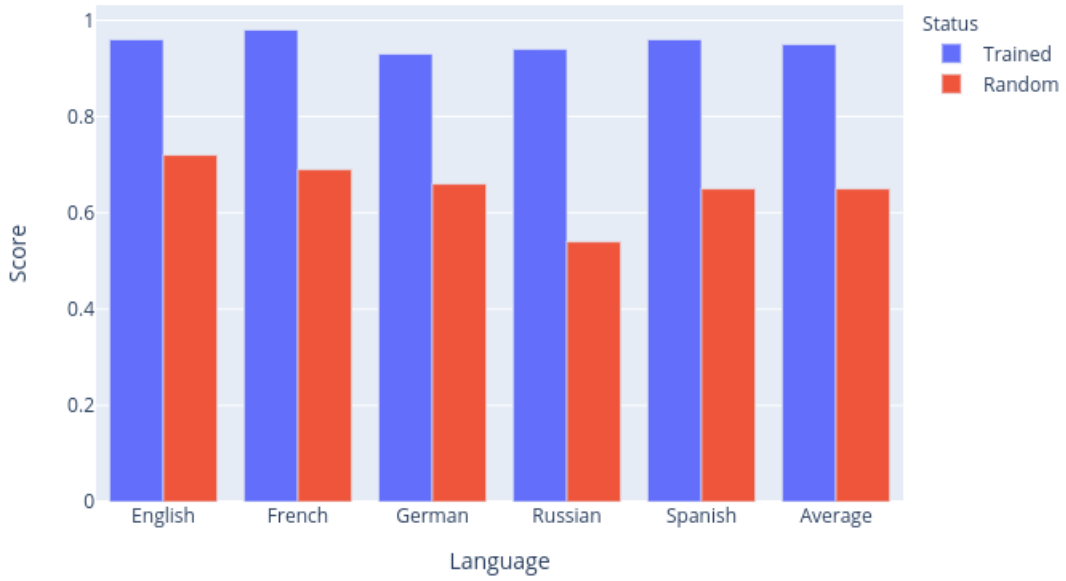


Figure 2.13: Comparison of weighted F1 scores for trained BERT models vs. random baseline, results averaged over features and layers

2.6 Conclusion

This chapter described three different experiments which explored contextualized embeddings of five Indo-European languages, specifically by investigating different subsets from those languages marked for different morphological features. In addition to experimenting on different language-feature pairs, the experiments also examined performance across layer, hoping to gain insight into the process by which BERT generates its embeddings through its layers.

The first of the experiments sought to elucidate certain aspects of the structure of hypothesized manifolds underlying the observed point clounds. It estimated the intrinsic dimensionality of the underlying space for each point-cloud, showing patterns of variation across language, feature, and layer. The variation across different features and languages was not intuitive, for instance more morphologically complex features (i.e. features with more possible

Layer\Lang.	English	French	German	Russian	Spanish	Average
Input	0.0623	0.1354	0.0611	0.1057	0.0616	0.0852
1	0.1083	0.1913	0.0845	0.1184	0.0582	0.1121
2	0.0691	0.1308	0.0329	0.1359	0.0764	0.089
3	0.0422	0.1517	0.0393	0.1044	0.1541	0.0983
4	0.0812	0.2009	0.017	0.1047	0.1147	0.1037
5	0.0138	0.1426	0.0178	0.0512	0.0851	0.0621
6	0.0126	0.221	0.0226	0.0248	0.0448	0.0651
7	0.0141	0.1376	0.0224	0.0212	0.0356	0.0462
8	0.0187	0.156	0.0141	0.0194	0.0253	0.0467
9	0.0143	0.1586	0.0156	0.0172	0.0159	0.0443
10	0.0108	0.1559	0.0235	0.0158	0.003	0.0418
11	0.0058	0.1986	0.0128	0.0238	0.0065	0.0495
12	0.1441	0.0896	0.0378	0.0778	0.055	0.0809
Average	0.046	0.1592	0.0309	0.0631	0.0566	

Table 2.14: Layer-wise performance of DEC.

Layer\Lang.	English	French	German	Russian	Spanish	Average
Input	0.0155	0.0183	0.0097	0.0152	0.0062	0.013
1	0.0169	0.0135	0.0119	0.0123	0.0046	0.0119
2	0.0162	0.0182	0.0128	0.0175	0.0048	0.0139
3	0.0048	0.0192	0.0102	0.0089	0.0072	0.01
4	0.0165	0.0236	0.0082	0.0084	0.0064	0.0126
5	0.0067	0.0224	0.0095	0.0108	0.0028	0.0104
6	0.0125	0.0139	0.0073	0.0073	0.0038	0.009
7	0.0116	0.0194	0.0053	0.0083	0.0015	0.0092
8	0.0118	0.0161	0.0054	0.0087	0.0021	0.0088
9	0.0084	0.0134	0.0081	0.0059	0.0019	0.0075
10	0.019	0.0209	0.0027	0.0096	0.0023	0.0109
11	0.0107	0.0096	0.0044	0.0081	0.0024	0.0071
12	0.0066	0.0045	0.0042	0.0061	0.0029	0.0049
Average	0.0121	0.0164	0.0077	0.0098	0.0038	

Table 2.15: Layer-wise performance for random baseline DEC.

feature values) do not necessarily result in representations with higher intrinsic dimensionality. Furthermore, whereas French showed the lowest intrinsic dimensionality estimation when estimated by a linear dimensionality-estimation method, it showed the highest when estimated by the more powerful neural network-based method.

The variation of intrinsic dimensionality across layer however showed more interesting

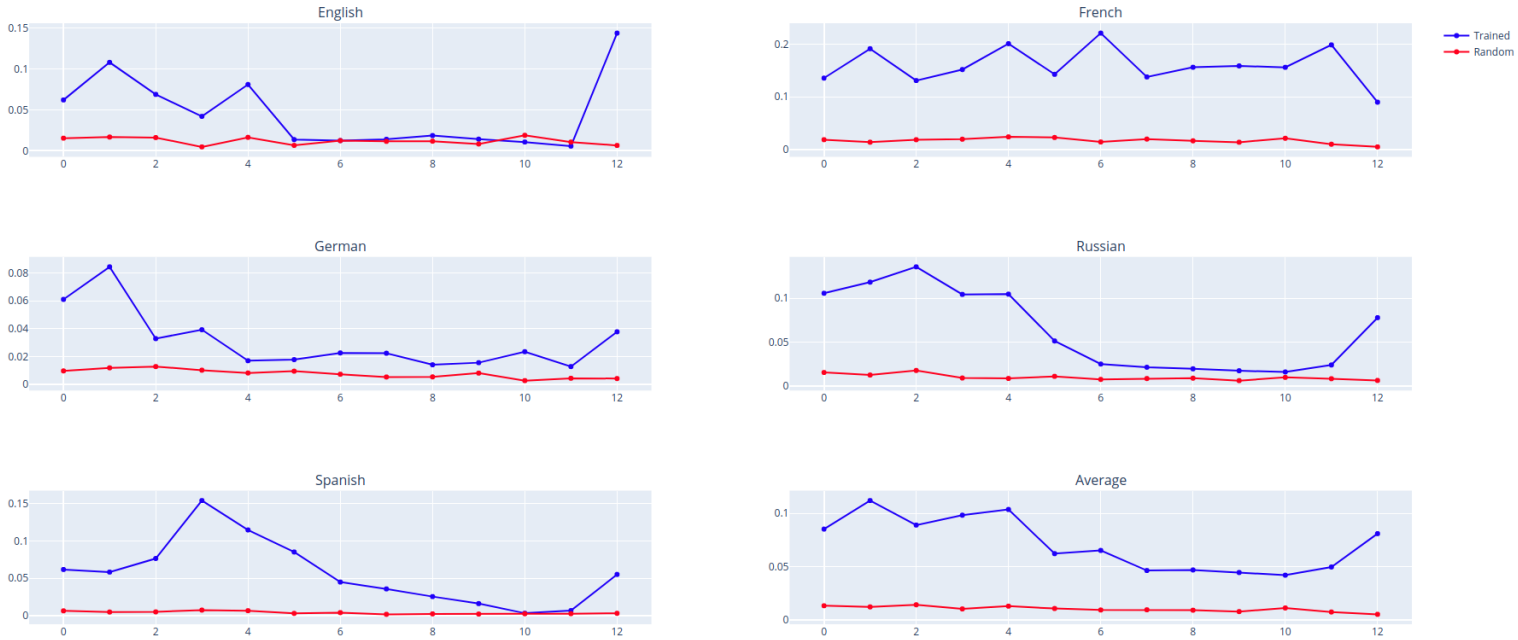


Figure 2.14: Comparison of weighted layer-wise AMI scores for trained BERT models vs. random baseline.

results. Per the linear PCA-based method, intrinsic dimensionality appeared to increase through the layers, finally plateauing in the mid-late layers with a steep drop for the late layers. When looking at the non-linear neural network-based estimations however, the opposite trend was observed. Early layer representations showed relatively high instrinsic dimenisonality, but these figures dropped quickly and stabilized by the mid layers. I hypothesized that this pattern of linear dimensionality estimation increasing while non-linear estimation decreases is analogous to space-filling curves.

The second experiment focused on directly extracting morphological information from contextualized embeddings by means of simple classification tasks. On both a linear and non-linear classifier, all data sets derived from BERT models performed highly in terms of F1-score, presenting strong evidence that BERT models predictably encode morphological features in their high-dimensional embeddings. Especially high performance by the linear classifier suggests that BERT embeds words into convex subregions indicative of morphological feature value. Furthermore, as would be expected, the confounds of syncretism and

large numbers of potential feature values correlate negatively with classifier performance, showing that while BERT does reflect morphological information, it does not do so perfectly and fails in predictable ways.

The third experiment addressed similar questions as the second, but used differing methods. In contrast with the supervised methods of the second experiment, the unsupervised method of the third experiment was apparently less capable of extricating morphological information from the contextualized embeddings. Specifically, the unsupservised clustering task did not return near perfect scores, as the linear classifier did. However, via direct comparison with random baseline scores it was shown that the unsupervised task did show BERT models encode non-trivial amounts of morphological information, and particularly in the early layers. Thus, the unsupervised method in this case provided complementary information to the supervised classes in Section 2.4.

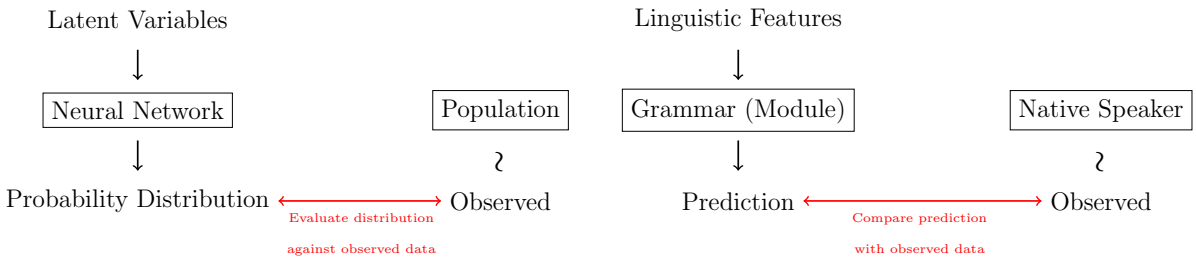
CHAPTER 3
DISENTANGLING (M)BERT EMBEDDINGS VIA GENERATIVE ASSUMPTIONS

3.1 Introduction

This chapter adapts the technique of disentangled representation learning (Higgins et al., 2016; Siddharth et al., 2017; Dupont, 2018) from the computer vision literature to a BERT models' contextualized word embeddings. Disentangled representation learning has as its objective to learn factorized representations of data, and in doing so arrive at more interpretable representations, where different factors account for different sources of variation in the data. In the context of the present chapter, this will entail disentangling—that is factoring out—different sources of linguistic information hypothesized to be contained in BERT embeddings. This chapter will show that it is possible to simultaneously tease out morphological, syntactic, and lexical semantic information from BERT embeddings by training a neural network model which explicitly factors BERT's high-dimensional space in a linguistically-motivated way.

In order to accomplish this, I employ a form of *generateive neural network*, specifically the *variational autoencoder* (VAE; Kingma and Welling, 2013), which allows for factoring high-dimensional continuous data such as BERT embeddings into multiple factors of variation. The course taken in this chapter is to factor the embeddings following assumptions from *generative grammar*. Before exploring the specifics of VAEs, it is worth considering the nomenclature *generative neural network*, and how analogies can be drawn to *generative grammar*, which partially motivate this chapter.

While generative neural networks and generative grammar have evolved as distinct fields, this coincidence of naming is not accidental, as at the proper level of abstraction they make virtually identical assumptions about the phenomena they seek to model. This assumption is that the data they model are the result of some underlying process operating on unseen



(a) Generative Neural Network (b) Generative Grammar

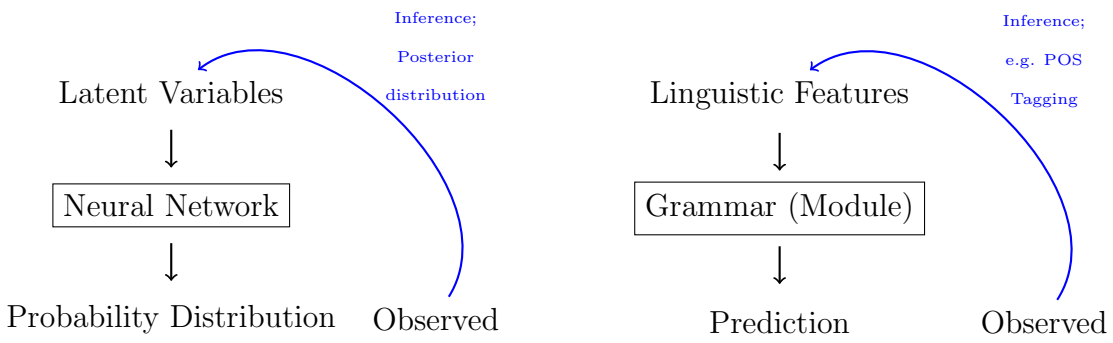
Figure 3.1: High-level schematic of generation in generative neural networks and generative grammar.

factors of variation. In generative neural networks, this process is modeled by a neural network, and the unseen factors of variation are referred to as *latent variables*. In generative grammar, the underlying process is (some module of) the grammar itself, often implemented as a transduction, and the unseen factors of variation are inputs to that grammar, generally linguistic features. High-level schematics of the frameworks for generative neural networks and generative grammar are in Figure 3.1.

In addition to the similar generative stories, there are also similarities in *inference* procedures. In generative neural networks, inference amounts to finding *posterior* distributions, or calculating probabilities of latent variables given observed data. In a generative linguistic framework, inference amounts to determining some linguistic features (e.g. as in tagging) or structure (e.g. as in parsing) responsible for an utterance. Figure 3.2 depicts the generative processes with inference.

This chapter models BERT embeddings with generative neural networks, playing on the analogy between generative neural networks and generative grammar in defining the latent variables of the model in such a way as to resemble linguistic assumptions. Continuiung the analogy, the grammar is then played by a neural network. Inference over the latent variables of the model then amounts to investigating BERT embeddings for linguistic information, and as will be made explicit below, encodes the embeddings in such a way as to factor the embeddings into different types of linguistic information.

This chapter assumes that there are three sources of linguistic information present in



(a) Generative Neural Network (b) Generative Grammar

Figure 3.2: High-level schematic of generation and inference in generative neural networks and generative grammar; inference procedures in blue.

an occurrence of a word: morphological features, syntactic features, and lexical semantics. Here, by morphological features I am referring to inflectional morphological features, such as those discussed in Chapter 2. Such morphological features take one of a fixed number of values, including a potential value of None. An example is the feature of Person, for which a word can be marked 1st, 2nd, 3rd, or None. By syntactic features, I am referring to part-of-speech, such as Noun, Verb, etc. Finally, by lexical semantics, I am referring to everything else which a native speaker understands about a word in context which is neither morphological nor syntactic. For illustration, consider the word *sources* in the following example sentences.

(1) Wal-Mart *sources* all of its goods from overseas.

(2) The investigator gathered intel from his *sources*.

In the first example, a native-speaker understands—if only subconsciously—that *sources* is a Verb (syntactic inference), and furthermore that it is marked Person=3, Tense=Present, and Number=Singular (morphological inference). Upon hearing the occurrence of *sources* in the second example, the same speaker would understand that it is a Noun, and marked Number=Plural, in addition to understanding any lexical information contained therein.

A generative neural network modeling BERT's contextualized embeddings—themselves

models of word occurrences—should then have three distinct latent variables; one discrete variable for morphology, one discrete variable for syntax, and one continuous variable for lexical semantics. Furthermore, the form of the morphological variable should somehow encode that there are multiple distinct features, and that these features have potentially differing numbers of outcomes. Likewise, the syntactic variable should have as many potential outcomes as there are parts-of-speech; a high-dimensional continuous variable is chosen so as to capture the remaining non-morphological and non-syntactic information, referred to here as lexical semantics.

Inference given such a model amounts to feature tagging with respect to the morphological and syntactic features, just as was done by the hypothetical native speaker mentioned above with example sentences (1) and (2). Many tasks are reasonable for performing inference over the leixcal semantic variable, and thus this chapter decides on two tasks. The first is a variant of the word sense disambiguation task[1] and the second is sentiment analysis. These tasks are chosed because they are each tasks heavily dependent on lexical semantics and agnostic with regard to morphosyntactic features.[2] A schematic of the model I propose is as in Figure 3.3.

This model assumes that BERT embeddings can be factorized into three latent variables, just as word occurrences can be factorized into morphological, syntactic, and semantic information. The model is trained to approximate the underlying probability distribution from which BERT embeddings are assumed to be drawn by training the neural network to maximize a probability density given observed BERT embeddings, with morphological features, syntactic features, and a vector representing lexical semantics as input.

In order to test the efficacy of factoring BERT embeddings in such a way, this chapter

1. Word sense disambiguation refers to the task of identifying a word's meaning in context. Here, I use a variant of the task in which the goal is to determine whether two separate instances of an ambiguous word have the same meaning or not. Going forward, I will refer to this task simply as word sense disambiguation for simplicity. Further details on the task follow in Section 3.4.3.

2. The word sense disambiguation datasets are controlled such that word senses do not vary across part-of-speech.

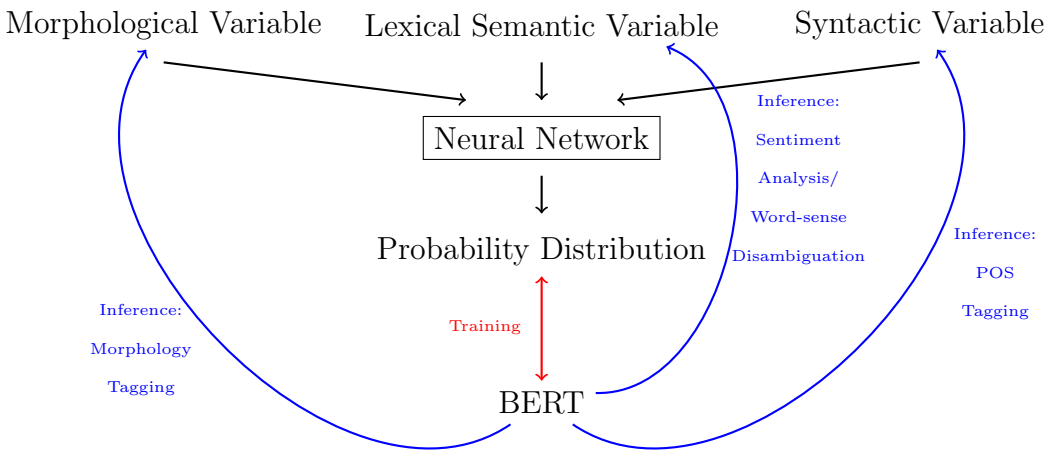


Figure 3.3: Proposed generative schematic for BERT embeddings.

runs four experiments by performing inference on each of the latent variables, using multilingual BERT, or MBERT, a BERT model trained on data from 104 different languages. A morphological tagging experiment is run on the morphological variable, a syntactic (i.e. part-of-speech) tagging experiment is run on the syntactic variable, and word sense disambiguation and sentiment analysis experiments are run using the lexical semantic variable. In each of the four experiments, I also perform experiments in a *zero-shot* setting, testing on data from a language not seen during training. In showing that it is possible to perform well in the zero-shot setting, I argue that the factor model is robustly extracting morphological, syntactic, and semantic information from MBERT embeddings, generalizing linguistic patterns cross-linguistically.

Furthermore, the ability to simultaneously perform both morphological feature inference, syntactic feature inference, word sense disambiguation, and sentiment analysis shows that factorizing MBERT embeddings in a manner consistent with assumptions from generative grammar is a viable means of learning disentangled for word embeddings. Disentangled representation learning—while an active topic in the computer vision literature—has not yet been explored to my knowledge on word embeddings.[3] The training of simple classifiers—

3. There is at least one study (Jain et al., 2018) investigating disentangled representations for NLP, but

while instructive—cannot provide this multi-faceted level of insight into word embeddings.

The remainder of this chapter is structured as follows. Section 3.2 provides background information for the rest of the chapter. Section 3.3 describes variational autoencoders, the framework used in this chapter, and provides specifics on the model to be used. Section 3.4 then describes the experiments for the chapter. Section 3.5 presents and discusses the results of the experiments, and Section 3.6 concludes the chapter.

3.2 Background Information

3.2.1 Notation

For this chapter, random variables are denoted by italicized uppercase Roman letters. There are four random variables of interest in this chapter. The first is the observed BERT embeddings; denote this variable by X. The second is our syntactic latent variable; let S denote this variable. The third is the morphological latent variable, denoted by M. Finally, let Z denote the latent variable representing lexical semantics.

As generative models attempt to learn a joint probability distribution over latent and observed variables, this chapter seeks to learn the joint distribution over observed BERT embeddings, and latent morphological, syntactic, and lexical semantic variables; denote this distribution by $P(X, M, S, Z)$. Let $P(X)$ then denote the marginal distribution over BERT embeddings, $P(M)$ the marginal distribution over morphological features, and so on. The probability function is denoted by $p(\cdot)$, for example $p(\mathbf{x})$ denotes the density function of $P(X)$. Conditional probabilities are written as $P(\cdot|\cdot)$, for example $P(S|X = \mathbf{x})$ denotes the conditional probability distribution over syntactic categories given BERT embedding $\mathbf{x}$; except when necessary for emphasis, this notation is shortened to $P(S|\mathbf{x})$. A sample from a distribution is denoted by a bold lowercase letter, e.g. $\mathbf{s} \sim P(S)$ denotes a sample from a probability distribution over syntactic categories. Additional notation will be introduced as

this study relates to document embeddings.

necessary.

3.2.2 Probability Distributions with Neural Networks

Probability Distributions

This chapter makes use of two types of probability distribution: Gaussian distributions[4] and categorical distributions. Gaussian distributions are continuous probability distributions over $\mathbb{R}^n$, and categorical distributions are discrete distributions over finite sets. Given the nature of morphological and syntactic features as discrete, probability distributions over variables S and M will be categorical. Probability distributions for the remaining variables—X for BERT embeddings and Z for the lexical semantic latent variable—will be Gaussian.

Gaussian distributions over $\mathbb{R}^n$ are fully specified by two parameters, a mean parameter $\boldsymbol{\mu} \in \mathbb{R}^n$, and a covariance matrix $\Sigma \in \mathbb{R}^{n \times n}$, where the covariance matrix is positive semi-definite. Gaussian distributions will be denoted $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$. The probability density function of a Gaussian distribution is as in Equation 3.1.

$$p(\mathbf{x} \in \mathbb{R}^n) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} exp(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^\top \Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu})) \tag{3.1}$$

The density function is important because it will be used below to determine the probability density of vectors such as MBERT embeddings; training will consist partly of maximizing this function with respect to embeddings.

Categorical distributions over finite support $\{1, \ldots, k\}$ are fully specified by a probability vector $\boldsymbol{\pi} \in \Delta^k \subset \mathbb{R}^k$, where $\Delta^k = \{\mathbf{x} \in \mathbb{R}^k \mid x_i \in \mathbf{x} \geq 0, \sum_i x_i = 1\}$ is the probability simplex. The restriction to Δ^k ensures that $\boldsymbol{\pi}$ properly defines a probability distribution over the support. Categorical distributions will be denoted as $Cat(\boldsymbol{\pi})$. The probability mass function of a categorical distribution is as in Equation 3.2.

4. Here, all Gaussian distributions are assumed to be multivariate Gaussian distributions.

$$p(i \in \{1, \ldots, k\}) = \pi_i \tag{3.2}$$

This says that the probability of $i \in \{1, \ldots, k\}$ is the i^{th} entry of probability vector $\boldsymbol{\pi}$. This function will be used for inference, determining the probability of different syntactic and morphological features given BERT embeddings.

Parameterizing Distributions with Neural Networks

Importantly, both of these distributions can be fully defined by a fixed number of parameters. Gaussian distributions are fully specified by two parameters, a mean vector denoted $\boldsymbol{\mu}$, and a covariance matrix denoted Σ, and categorical distributions can be fully specified by a probability vector $\boldsymbol{\pi}$. As this is the case, we can learn probability distributions by parameterizing them with neural networks, assuming the output of the neural networks matches the form of the parameters for the distributions.

As this is often done with categorical distributions in the natural language processing literature,[5] we start there. The common way of doing this is to use a softmax activation layer as the last layer of a neural network. The softmax activation function is as in Equation 3.3.

$$softmax(\mathbf{x} \in \mathbb{R}^k)_i = \frac{exp(x_i)}{\sum_{j=1}^{k} exp(x_j)} \tag{3.3}$$

The output of $softmax(\mathbf{x})$ is then a vector in the probability simplex $\Delta^k \subset \mathbb{R}^k$, and thus defines a licit categorical probability distribution. A *Categorical Neural Network* is schematized in Figure 3.4.

The parameterizing of Gaussian distributions via neural networks is similarly possible, though slightly more involved. As Gaussian distributions are specified by two parameters, the output of any neural network specifying a Gaussian distribution must have two outputs:

5. This was in fact done in Chapter 2.

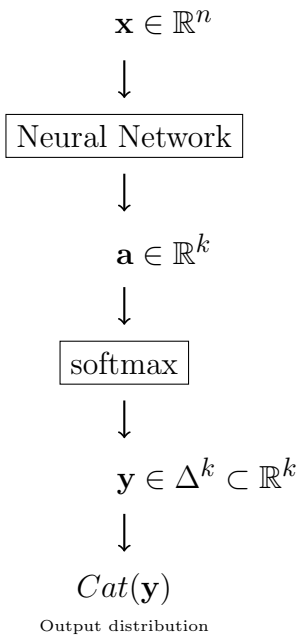


Figure 3.4: Schematic of a Categorical Neural Network, a neural network parameterizing a categorical distribution.

one for mean $\boldsymbol{\mu}$ and one for Σ. Beginning with $\boldsymbol{\mu}$, this can be any vector $\in \mathbb{R}^n$, and so there is no restriction on the output of this parameter. For Σ, rather than output a matrix, it is standard in the literature to assume that the covariance matrix is a diagonal matrix,[6] and simply output a vector which serves as the diagonal of the matrix. This vector is typically denoted $\boldsymbol{\sigma}^2$ to stress the fact that it is a vector and the *diagonal* of a covariance matrix rather than the covariance matrix itself. Furthermore, the vector is constrained to be non-negative, a constraint which can be met with e.g. an exponentiating layer as the final layer of the network. A *Gaussian Neural Network*, a neural network parameterizing a Gaussian distribution, is as depicted in Figure 3.5.

Parameterizing probability distributions with neural networks in this way provides a means of fitting distributions to potentially complex data using the non-linear function-learning flexibility of deep neural networks. It is standard practice in the literature to

6. This is a rather strong constraint on the types of distribution that the network can output, and there is a large literature— particularly in the computer vision community—which investigates methods of producing more expressive probability distributions in an efficient fashion.

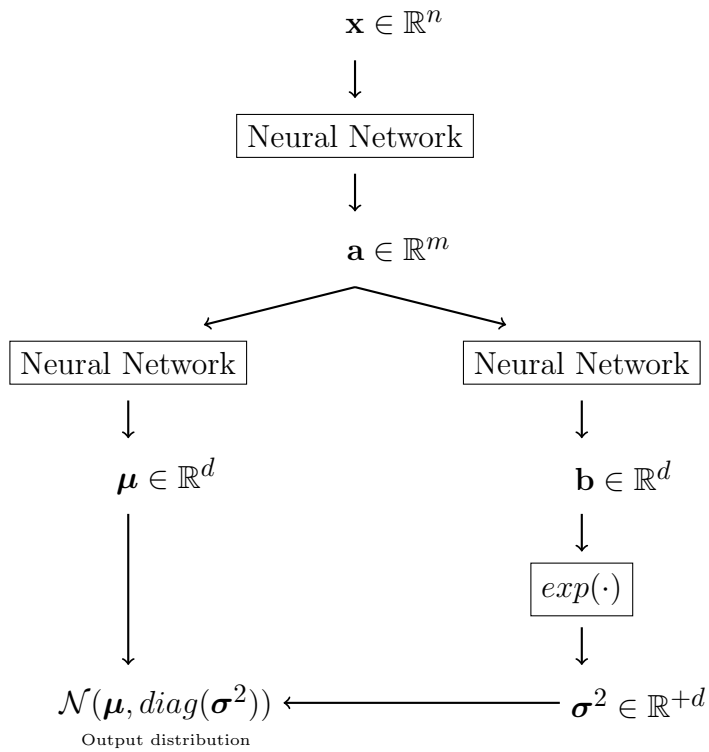


Figure 3.5: Schematic of a Gaussian Neural Network, a neural network parameterizing a Gaussian distribution.

denote neural-network parameterized probability distributions explicitly. For instance, the probability distribution schematized in Figure 3.5 would be written $P(Y|\mathbf{x};\theta)$, where θ denotes the parameters of the neural networks. However, this extra notation will be omitted except when used for emphasizing a distribution is parameterized by a neural network.

3.3 Variational Autoencoders

3.3.1 Motivation and Formulation

Before adapting variational autoencoders to factor MBERT embeddings linguistically, it is helpful to first examine the variational autoencoder as it was originally conceived. Variational autoencoders were first introduced in Kingma and Welling (2013), and as a framework offer a solution to two challenges with generative neural networks as described above. The first challenge involves training the model, and the second involves inference over latent variables.

In order to discuss these challenges in context, consider the case in which one wishes to learn an unknown distribution from which data points $\mathbf{x} \in \mathbb{R}^n$ are drawn. VAEs assume a generative model for data X with one latent variable Z, seeking to learn the joint distribution $P(X, Z)$. Latent Z is assumed continuous, and generation proceeds by sampling $\mathbf{z} \in \mathbb{R}^m$ from pre-defined probability distribution $P(Z) = \mathcal{N}(\mathbf{0}, \mathbf{I})$,[7] and passing samples through a Gaussian Neural Network as in Figure 3.5 above, producing distribution $P(X|\mathbf{z}; \theta)$. VAEs learn to approximate the unknown data distribution with $P(X|\mathbf{z}; \theta)$ by iteratively updating the neural network parameters to maximize the probability density with respect to data points $\mathbf{x}$. This is schematized as in Figure 3.6.

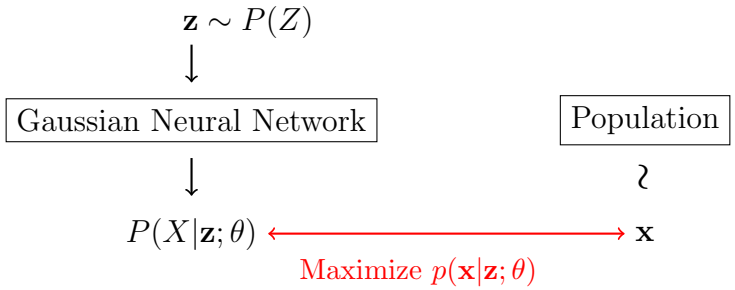


Figure 3.6: Schematic of single example for training generative model.

Unfortunately, sample-based learning is slow and computationally expensive in this case, particularly with larger dimensions (see Section 2.1 of Doersch (2016) for discussion on why this is the case). Furthermore, posterior inference—that is, calculation of $p(\mathbf{z}|\mathbf{x})$—is not tractable in this case as per Bayes' Rule, $p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x})}$, and calculation of $p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})dz$ involves integration over $\mathbf{z}$. The solution offered by VAEs is the introduction of an additional neural network for inference, as schematized in Figure 3.7.

As can be seen in the schematic, the variational autoencoder samples $\mathbf{z}$ not from $P(Z)$, but from $Q(Z|\mathbf{x})$, an approximation to the posterior $P(Z|\mathbf{x})$. Sampling $\mathbf{z}$ dependent on $\mathbf{x}$ greatly speeds up training, alleviating the first challenge with training generative neural networks mentioned above. At the same time, predicting variable Z given $\mathbf{x}$ as in

7. That is, the prior distribution over Z is an isotropic Gaussian, with the zero vector as mean and identity matrix as covariance matrix.

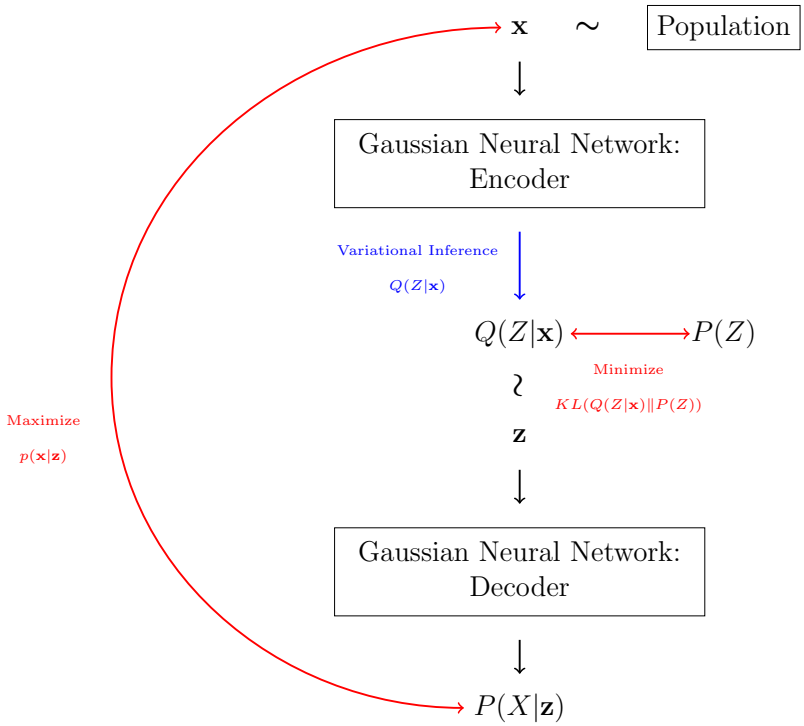


Figure 3.7: Schematic for variational autoencoder.

$Q(Z|\mathbf{x})$ performs approximate inference in place of the intractable $P(Z|\mathbf{x})$, thus alleviating the inference-related challenge. Use of an approximate posterior distribution like $Q(Z|\mathbf{x})$ in place of an intractable posterior like $P(Z|\mathbf{x})$ is known as a *variational* technique (Wainwright and Jordan, 2008). This fact, along with the nature of the network encoding data into a latent code and then decoding it for training, gives the variational autoencoder its name. The objective to maximize for variational autoencoders is as in Equation 3.4.[8]

$$\mathbb{E}_{\mathbf{x}\sim Pop}[\mathbb{E}_{\mathbf{z}\sim Q(Z|\mathbf{x})}[p(\mathbf{x}|\mathbf{z}) - KL(Q(Z|\mathbf{x})\|P(Z))]] \tag{3.4}$$

In this objective, maximizing $p(\mathbf{x}|\mathbf{z})$ with respect to observed $\mathbf{x}$ encourages the decoder neural network to decode latent draws $\mathbf{z}$ to look like observed data $\mathbf{x}$, and simultaneously encourages the encoder to parameterize $Q(Z|\mathbf{x})$ such that it contains as much information about $\mathbf{x}$ as possible, since it must sample from this distribution to reconstruct $\mathbf{x}$ in the

8. This term is known as the ELBO (**E**vidence **L**ower **BO**und), for reasons which are beyond the scope of this chapter. However, see Appendix C for an explanation of the term and its derivation.

decoder. Minimizing the KL-divergence between $Q(Z|\mathbf{x})$ and $P(Z) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ encourages the encoder to encode $\mathbf{x}$ in such a way that the probability distribution $Q(Z|\mathbf{x})$ looks like the isotropic Gaussian, and so this acts like a regularization term in constraining the shape of the distribution.

3.3.2 Extension to Multiple Latent Variables

The standard VAE as just described assumes a lone continuous latent variable Z, in effect assuming that datapoints $\mathbf{x}$ can be effectively encoded into such a latent variable. Above, I proposed to model BERT embeddings with three latent variables, discrete morphological and syntactic latent variables M and S, and a continuous latent variable Z. Use of multiple latent variables amounts to an explicit factoring of the datapoints, and is one of two approaches in the literature of performing disentangled representation learning (Kingma et al., 2014; Dupont, 2018).[9] The rationale behind this approach is that by factoring datapoints into multiple latent variables, the underlying sources of variation will be encoded in separate variables. With this as background, this section extends the VAE to account for multiple latent variables so as to factor MBERT embeddings linguistically as suggested. The proposed model is as in Figure 3.8; compare against Figure 3.3.

However, as in the general case discussed above, the problems of slow training and intractable posterior inference hold in this setup as well, and so an additional variational inference neural network is necessary. The extended VAE setup I propose is schematized in Figure 3.9.[10]

Here, we see that there are three encoder neural networks, one Categorical Neural Network for latent variable M, another Categorical Neural Network for latent variable S, and one Gaussian Neural Network for latent variable Z. These three neural networks are tasked

9. The other approach, as exemplified in Higgins et al. (2016), attempts to encode datapoints in singular continuous spaces such that the individual dimensions are interpretable.

10. The extension of the VAE framework in such a way to assume multiple, mixed continuous/discrete latent variables has been proposed in the literature to address problems in computer vision as in Kingma et al. (2014) and Dupont (2018).

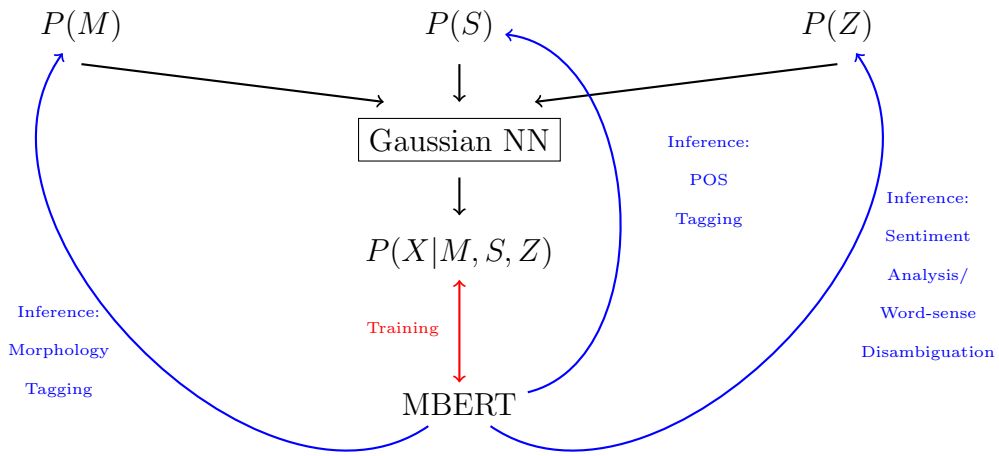


Figure 3.8: Proposed generative model for MBERT embeddings.

with distilling out of MBERT embedding $\mathbf{x}$ morphological, syntactic, and lexical semantic information respectively. The objective is as in Equation 3.5. Given MBERT embedding $\mathbf{x}$, call this objective $\mathcal{U}(\mathbf{x})$.

$$\mathbb{E}_{\mathbf{m}\sim Q(M|\mathbf{x})}\Big[\mathbb{E}_{\mathbf{s}\sim Q(S|\mathbf{x})}\Big[\mathbb{E}_{\mathbf{z}\sim Q(Z|\mathbf{x})}\Big[p(\mathbf{x}|\mathbf{m},\mathbf{s},\mathbf{z}) - KL[Q(M|\mathbf{x})\|P(M)] - KL[Q(S|\mathbf{x})\|P(S)] - KL[Q(Z|\mathbf{x})\|P(Z)]\Big]\Big]\Big] = \mathcal{U}(\mathbf{x}) \tag{3.5}$$

As in the general case of VAEs, datapoints are encoded into probability distributions, and then samples are taken from these encoded distributions and fed to a decoder to recreate the original datapoints. The term $p(\mathbf{x}|\mathbf{m},\mathbf{s},\mathbf{z})$ in Equation 3.5 encourages the output of the decoder to give the observed MBERT embedding $\mathbf{x}$ a high probability density given morphological feature(s) $\mathbf{m}$, syntactic feature $\mathbf{s}$, and lexical semantics $\mathbf{z}$. As before, latent variables are sampled from the output distributions of the encoders, $Q(M|\mathbf{x})$, $Q(S|\mathbf{x})$, and $Q(Z|\mathbf{x})$, rather than the prior distributions $P(M)$, $P(S)$, and $P(Z)$, greatly speeding up training. The prior distributions $P(M)$ and $P(S)$ are defined as uniform categorical distributions (i.e. each morphological/syntactic feature is equally likely), and the prior on lexical semantic

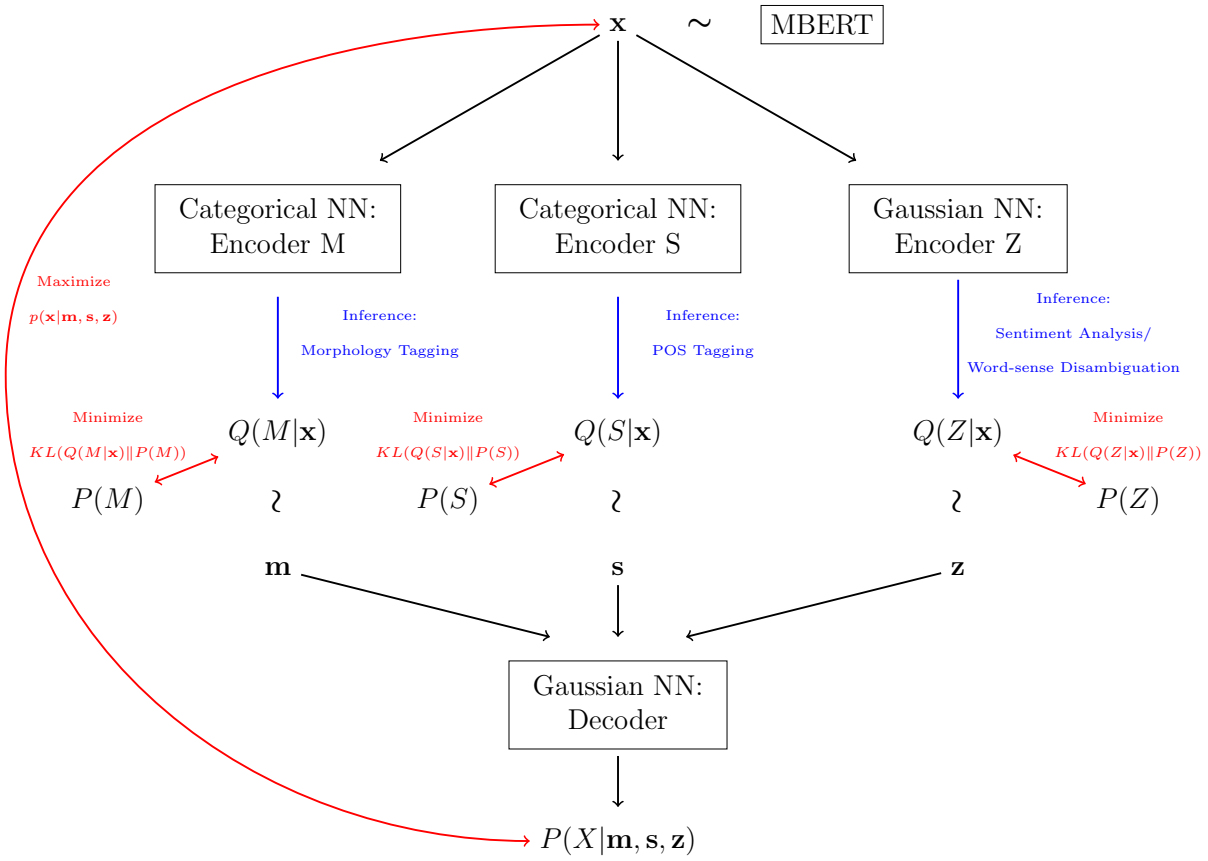


Figure 3.9: Linguistically factorized Variational Autoencoder for MBERT embeddings; unsupervised case.

information $P(Z)$ is the isotropic Gaussian $\mathcal{N}(\mathbf{0},\mathbf{I})$. These prior distributions again serve as regularizers, as the objective seeks to minimize the KL-divergence between each encoder output distribution and the relevant prior, e.g. $KL[Q(Z|\mathbf{x})\|P(Z)]$.

3.3.3 Extension to Semi-supervised Learning

However, care must be taken to ensure that the encoders correctly encode the information they are supposed to be responsible for, as given only the objective in Equation 3.5 there is no reason to assume morphological information will flow to $Q(M|\mathbf{x})$, syntactic information to $Q(S|\mathbf{x})$, and semantic information to $Q(Z|\mathbf{x})$. Indeed, it is a known difficulty with mixed continuous/discrete variational autoencoders that the encoders tend to ignore the discrete latent variable(s) and encode everything into the continuous variable. The "ignoring" of

a latent variable is known as *posterior collapse*, the phenomenon in which a variational posterior distribution collapses to the prior distribution.

In the case of the model described above, this would refer to the situation in which $Q(M|\mathbf{x}) \approx P(M)$ and/or $Q(S|\mathbf{x}) \approx P(S)$, which arises when the pressure to minimize $KL[Q(M|\mathbf{x})\|P(M)]$ and $KL[Q(S|\mathbf{x})\|P(S)]$ overcomes the pressure to encode information about $\mathbf{x}$ into $Q(M|\mathbf{x})$ and $Q(S|\mathbf{x})$. In mixed continuous/discrete variational autoencoders this is a frequent occurence (Dupont, 2018), as the model generally finds it most advantageous to encode sufficient information about $\mathbf{x}$ into the continuous variable (in this case $Q(Z|\mathbf{x})$) such that it can ignore the discrete latent variables, allowing them to collapse to their respective priors. In the event of posterior collapse of discrete variables in this model, inference over morphological and syntactic features would fail, as the probability distributions would encode nothing about word embedding $\mathbf{x}$ and would collapse to uniform distributions over features.[11]

To encourage the Categorical Neural Networks to avoid posterior collapse and to encode the discrete linguistic information they are responsible for, the system in this chapter is trained in a semi-supervised fashion (Kingma et al., 2014), meaning that some datapoints $\mathbf{x} \sim$ MBERT are equipped with morphological and syntactic tags which are used to train $Q(M|\mathbf{x})$ and $Q(S|\mathbf{x})$ directly. In the event of a supervised example, the model is trained slightly differently, as in Figure 3.10.

As can be seen, the training processes depicted in Figures 3.9 and 3.10 are virtually identical with three exceptions. The first is that in the latter case $\mathbf{x}$ is drawn from MBERT_{Tagged} rather than simply MBERT. This means that MBERT was used to encode sentences for which morphological and syntactic tags were available, rather than unlabeled text. The second is that given a morphological tag $M\text{-}Tag$ and syntactic tag $S\text{-}Tag$ for a word embedding

11. Posterior collapse is a problem not only with mixed continuous/discrete variatoinal autoencoder, but with variational autoencoders in general. It occurs when e.g. the generative model is sufficiently powerful that the need to draw from a variational posterior is obviated and reconstruction of the datapoints can proceed equally well simply drawing from the prior (Bowman et al., 2015).

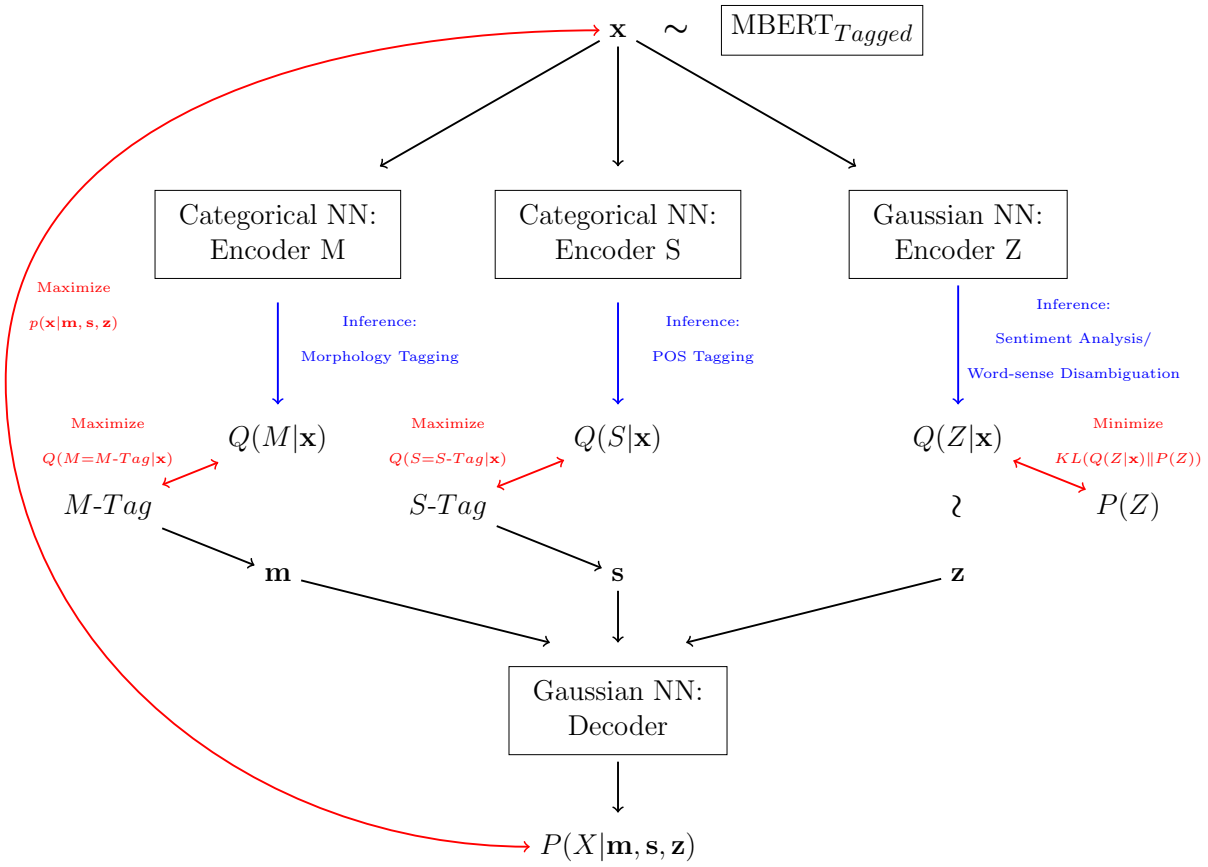


Figure 3.10: Linguistically factorized Variational Autoencoder for MBERT embeddings; supervised case.

$\mathbf{x}$, morphological inference is performed so as to maximize $Q(M{=}M\text{-}Tag|\mathbf{x})$ and syntactic inference so as to maximize $Q(S{=}S\text{-}Tag|\mathbf{x})$. This direct training ensures that morphological information is encoded into $Q(M|\mathbf{x})$ and syntactic information into $Q(S|\mathbf{x})$, and whatever information in $\mathbf{x}$ that is not morphological or syntactic is left to be routed through $Q(Z|\mathbf{x})$. The final difference is that $\mathbf{m}$ and $\mathbf{s}$ are no longer sampled from $Q(M|\mathbf{x})$ and $Q(S|\mathbf{x})$, but rather the provided $M\text{-}Tag$ and $S\text{-}Tag$ are fed directly to the decoder as $\mathbf{m}$ and $\mathbf{s}$. The objective is as in Equation 3.6, adapted from Kingma et al. (2014);[12] given MBERT embedding $\mathbf{x}$, morphological tag $\mathbf{m}$, and syntactic tag $\mathbf{s}$, call this objective $\mathcal{S}(\mathbf{x},\mathbf{m},\mathbf{s})$.

12. Detailed discussion of the derivation of this objective can be found at the particularly helpful blog-post at `https://bjlkeng.github.io/posts/semi-supervised-learning-with-variational-autoencoders/`.

$$\mathbb{E}_{\mathbf{z}\sim Q(Z|\mathbf{x})}\Big[p(\mathbf{x}|\mathbf{m},\mathbf{s},\mathbf{z}) + p(\mathbf{m}) + p(\mathbf{s}) - KL[Q(Z|\mathbf{x})\|P(Z)]\Big] + q(\mathbf{m}|\mathbf{x}) + q(\mathbf{s}|\mathbf{x}) = \mathcal{S}(\mathbf{x},\mathbf{m},\mathbf{s}) \tag{3.6}$$

The final objective for training the semi-supervised VAE given dataset of unlabeled MBERT embeddings, call it MBERT_U, and dataset of labeled MBERT embeddings, call it MBERT_L, is as in Equation 3.7.

$$\mathcal{J}(\text{MBERT}_U,\ \text{MBERT}_L) = \mathbb{E}_{\mathbf{x}\sim\text{MBERT}_U}\,\mathcal{U}(\mathbf{x}) + \mathbb{E}_{(\mathbf{x},\mathbf{m},\mathbf{s})\sim\text{MBERT}_L}\,\mathcal{S}(\mathbf{x},\mathbf{m},\mathbf{s}) \tag{3.7}$$

Training a semi-supervised variational autoencoder in such a way has proven useful for learning disentangled representations for images in the computer vision literature, e.g. Siddharth et al. (2017) disentangle in images of hand-written digits discrete underlying sources of variation (underlying digit) from continuous sources of variation (such as digit thickness). Below, I use the model just described to show that it is useful for disentangling morphological, syntactic, and lexical semantic information from MBERT's contextualized word embeddings as well. This is done via simultaneous inference over morphological features, syntactic features, and sentiment analysis. Specifics of these experiments follow.

3.4 Experiment Design

The experiments in this chapter make use of a total of three languages: Spanish, Galician, and English, using the MBERT model to embed words from all three.[13] Tagged and untagged data are made available only for Spanish when training the variational autoencoder, while zero-shot evaluation is performed on Galician and English.

13. Word embeddings are derived from MBERT as they were in Chapter 2: averaging the embeddings of all tokens which make up a word.

For morphological and syntactic inference, experiments are run on Spanish and Galician. The GSD Spanish treebank (Nivre et al., 2016) is used for Spanish tags, and the test set from the TreeGal (Garcia, 2016) treebank is used for zero-shot evaluation of Galician. For lexical semantic inference, the tasks of word sense disambiguation and sentiment analysis have been chosen. For word sense disambiguation, the Word-in-Context (WiC) dataset (Pilehvar and Camacho-Collados, 2018) is used, which consists of examples where two sentences are given, and each sentence contains a particular word with multiple possible meanings. The task is to predict whether the word in question has the same meaning in each of the two sentences, or whether the meaning is different between them. This task is then a binary classification task, and all sentences are in English. For the sentiment analysis experiments, I use the Spanish and English portions of the multilingual Amazon reviews corpus (Keung et al., 2020), using the 1-5 star rating as a proxy for sentiment with 1 star reviews being the most negative, and 5 star reviews the most positive. As the VAE model will not be trained with any English data, inferring the lexical semantics of English word embeddings in these two experiments amounts to zero-shot inference. Results for all experiments are reported after the epoch of training which results in the highest average dev-set scores for the morphological and syntactic inference tasks in Spanish.

3.4.1 Experiment 1: Morphological Feature Tagging

As mentioned, the task of morphological feature tagging in these experiments amounts to inference over the morphological variable M in the description above. This means using Encoder M to encode MBERT embedding $\mathbf{x}$ into categorical distribution $Q(M|\mathbf{x})$, and assigning the morphological feature value(s) associated with the highest probability mass.

Given the nature of inflectional morphology in languages like Spanish and Galician, the manner in which one would encode a word occurrence's morphological features/values as a categorical variable is to treat each possible collection of morphological features and values

Feature Values	Encoding
$[1]$	$[1,0,0,0,0,0,0,0,0,0,0]$
$[2]$	$[0,1,0,0,0,0,0,0,0,0,0]$
$[3]$	$[0,0,1,0,0,0,0,0,0,0,0]$
$[Sing]$	$[0,0,0,1,0,0,0,0,0,0,0]$
$[Plur]$	$[0,0,0,0,1,0,0,0,0,0,0]$
$[1, Sing]$	$[0,0,0,0,0,1,0,0,0,0,0]$
$[1, Plur]$	$[0,0,0,0,0,0,1,0,0,0,0]$
$[2, Sing]$	$[0,0,0,0,0,0,0,1,0,0,0]$
$[2, Plur]$	$[0,0,0,0,0,0,0,0,1,0,0]$
$[3, Sing]$	$[0,0,0,0,0,0,0,0,0,1,0]$
$[3, Plur]$	$[0,0,0,0,0,0,0,0,0,0,1]$

Table 3.1: Toy example of enumeration of collections of features and values, with associated one-hot encoding.

as atomic.[14] For instance, assume a language which marks only for Person (with values 1, 2, and 3) and Number (with values singular and plural). One possible encoding for such a toy example of morphological features would be as in Table 3.1.[15]

Such an encoding abstracts out any relation between word occurrences which share feature values, e.g. the encoding for $[1, Sing]$ is completely orthogonal to the encoding for $[1, Plur]$, in spite of the fact that they are each marked with Person=1. An alternative approach, and the one used for experiments in this chapter, is to encode morphological features separately as distinct morphological variables. Each morphological feature would then be represented by a categorical variable with $v + 1$ possible outcomes, where v is the number of possible feature values, and each variable has an extra outcome of $None$, as not all word occurrences are marked for each morphological feature. An example of such an encoding scheme is as in Table 3.2.

The first scheme, while simpler implementationally, disregards similarities between combinations of feature values, and furthermore does not allow for a fine-grained analysis of individual morphological features. The second encoding allows to compare performance

14. This is the approach followed for morphological tagging in e.g. Kondratyuk and Straka (2019).

15. As is common in machine learning, one-hot encoding is used to encode categorical variables rather than enumeration.

Feature Values	Person Encoding	Number Encoding	Combined Encoding
$[1]$	$[0, 1, 0, 0]$	$[1, 0, 0]$	$[0, 1, 0, 0, 1, 0, 0]$
$[2]$	$[0, 0, 1, 0]$	$[1, 0, 0]$	$[0, 0, 1, 0, 1, 0, 0]$
$[3]$	$[0, 0, 0, 1]$	$[1, 0, 0]$	$[0, 0, 0, 1, 1, 0, 0]$
$[Sing]$	$[1, 0, 0, 0]$	$[0, 1, 0]$	$[1, 0, 0, 0, 0, 1, 0]$
$[Plur]$	$[1, 0, 0, 0]$	$[0, 0, 1]$	$[1, 0, 0, 0, 0, 0, 1]$
$[1, Sing]$	$[0, 1, 0, 0]$	$[0, 1, 0]$	$[0, 1, 0, 0, 0, 1, 0]$
$[1, Plur]$	$[0, 1, 0, 0]$	$[0, 0, 1]$	$[0, 1, 0, 0, 0, 0, 1]$
$[2, Sing]$	$[0, 0, 1, 0]$	$[0, 1, 0]$	$[0, 0, 1, 0, 0, 1, 0]$
$[2, Plur]$	$[0, 0, 1, 0]$	$[0, 0, 1]$	$[0, 0, 1, 0, 0, 0, 1]$
$[3, Sing]$	$[0, 0, 0, 1]$	$[0, 1, 0]$	$[0, 0, 0, 1, 0, 1, 0]$
$[3, Plur]$	$[0, 0, 0, 1]$	$[0, 0, 1]$	$[0, 0, 0, 1, 0, 0, 1]$

Table 3.2: Toy example of enumeration of possible feature bundles and associated feature-value encoding.

across different morphological features, as was done in Chapter 2.

The treatment of variable M as multiple distinct variables, M_{Person}, M_{Number}, etc., requires a slight alteration in the presentation above. There is no longer a single encoder for morphological information, but rather multiple encoders, one for each feature. Probability distribution $Q(M|\mathbf{x})$ is then no longer a single categorical probability distribution, rather it is the joint distribution over all morphological features, and is defined as in Equation 3.8.[16]

$$Q(M|\mathbf{x}) = \prod_{feat \in feats} Q(M_{feat}|\mathbf{x}) \tag{3.8}$$

There are 21 different morphological features in the treebanks used in this chapter. The number of values a feature takes varies from feature to feature, with as few as one value (e.g. for the morphological feature called Polite) to as many as nine values, as in the case of the morphological feature PronType. A summary of the morphological features and associated

16. This definition of $Q(M|\mathbf{x})$ as the product of individual feature distributions $Q(M_{feat}|\mathbf{x})$ assumes that the probability distributions over different morphological features are independent. This is a simplifying assumption which does not hold in natural language, as there are clearly dependencies between the values of different features in natural language. For instance, if a word occurrence is marked with Case=Nominative, the probability that Mood=None—i.e. that the word is not marked for Mood—is very high. Nonetheless, the results below show that this simplifying assumption does not preclude the system from learning to encode morphological features from MBERT embeddings. The prior distributions $P(M_{feat})$ over individual feature variables are still assumed uniform; every feature value is equally likely.

Morphological Features	Associated Feature Values
AdpType	Prep
Case	Acc, Com, Dat, Nom
Clitic	Yes
Definite	Def, Ind
Degree	Abs, Cmp, Sup
Foreign	Yes
Gender	Com, Fem, Masc, Neut, Unsp
Mood	Cnd, Imp, Ind, Sub
NumType	Card, Frac, Mult, Ord, Range, Sets
Number	Plur, Sing, Unsp
Number[psor]	Plur, Sing
Person	1, 2, 3
Polarity	Neg
Polite	Form
Poss	Yes
PrepCase	Npr, Pre
PronType	Art, Dem, Emp, Ind, Int, Neg, Prs, Rel, Tot
Reflex	Yes
Tense	Fut, Imp, Past, Pqp, Pres
VerbForm	Fin, Ger, Inf, Part
Voice	Pass

Table 3.3: Morphological feature and associated values from treebanks used in this chapter. For descriptions of features/values, see `https://universaldependencies.org/u/feat/index.html`.

values as given in the treebanks is as in Table 3.3.

While treebanks such as the GSD Spanish treebanks used in this chapter can be credited for thoroughness with regard to their morphological descriptions of words, many of the features and values listed in Table 3.3 are not relevant from a theoretical linguistic perspective, at least not with regard to inflectional morphology. As such, this chapter restricts its attention to the morphological features discussed above in Chapter 2. The relevant subset of morphological features and associated values—along with the addition of a None value for each feature—are described in Table 3.4.

The morphological variable M in the VAE model in this chapter then consists of seven constituent morphological variables, one for each of Case, Gender, Mood, Number, Person, Tense, and VerbForm, with number of possible outcomes for each variable ranging from three

Morphological Features	Associated Feature Values
Case	None, Acc, Com, Dat, Nom
Gender	None, Fem, Masc
Mood	None, Cnd, Imp, Ind, Sub
Number	None, Plur, Sing
Person	None, 1, 2, 3
Tense	None, Fut, Imp, Past, Pres
VerbForm	None, Fin, Ger, Inf, Part

Table 3.4: Restricted set of morphological features and associated values to be used in encoding morphological variable M.

to five. Morphological inference is inferring feature value for each of these seven features, and inference is counted as correct if and only if all seven features are correctly inferred for value.

3.4.2 Experiment 2: Syntactic Feature Tagging

The encoding of syntactic variable S is simpler than the encoding of M. Inference over syntactic features, i.e. calculation of $Q(S|\mathbf{x})$, in this case amounts to part-of-speech tagging, which can be implemented as a simple k-way classification task. Here, there are eighteen distinct parts-of-speech as defined in the treebanks used in this chapter. These parts-of-speech are Adjective, Adposition, Adverb, Auxiliary, Coordinating Conjunction, Determiner, Interjection, Noun, Numeral, Particle, Pronoun, Proper Noun, Punctuation, Subordinating Conjunction, Symbol, Verb, Other, and a special value of None. S is then a categorical variable with 18 possible outcomes, and given MBERT embedding $\mathbf{x}$ a syntactic category is assigned by taking the outcome with the greatest probability mass in $Q(S|\mathbf{x})$.

3.4.3 Experiment 3: Word Sense Disambiguation

A variant of the word sense disambiguation task was chosen as the first task with which to perform inference over continuous latent variable Z, the latent variable responsible for capturing lexical semantic information in MBERT embeddings. Given the model here, in-

Ambiguous Word	Text 1	Text 2	Shared Meaning
Air	*Air* pollution	Open a window and let in some *air*	True
Justify	*Justify* the margins	The end *justifies* the means	False

Table 3.5: Examples from WiC dataset. Table adapted from Table 1 of Pilehvar and Camacho-Collados (2018).

ference over Z amounts to calculating $Q(Z|\mathbf{x})$. Usually, word sense disambiguation refers to the task of algorithmically identifying the meanings of words in context (Navigli, 2009). In this thesis, I use the term to refer to a simplified version of the task, which ultimately reduces to binary classification. Given two texts in which the same ambiguous word appears, the task is to determine whether the ambiguous word bears the same meaning in each of the two texts. This task is conducted using the Word-in-Context (WiC) dataset (Pilehvar and Camacho-Collados, 2018). Two examples from this dataset are as in Table 3.5.

All examples of ambiguous words are either nouns or verbs, and all examples are controlled for part-of-speech, meaning the intances of the ambiguous word in the texts will either both be nouns or both be verbs. However, examples are not controlled for capitalization or inflectional morphology. For instance, the ambiguous verb *justify* in the second example in Table 3.5 appears as *Justify* in the first text, while it appears as *justifies* in second text. The WiC dataset comes in train-dev-test splits of 5,428 , 638, and 1,400 examples each. However, because gold-standard tags are not provided for the test set, I perform no hyper-parameter optimization for this experiment and report final scores for perfomance on the dev set.

Performing this task is slightly different than morphological and syntactic tagging tasks above. For those tasks, inference itself (i.e. calculating $Q(M|\mathbf{x})$ and $Q(S|\mathbf{x})$) performed tagging, while in this case it is necessary to calculate $Q(Z|\mathbf{x})$, and then to use this as input to an additional classifier. This additional classifier takes the shape of a two-layer neural network with 100 dimensions in the hidden layer, ReLU activation for the hidden layer, and a single output neuron with sigmoid activation. The input is the concatenation of the word embeddings produced by MBERT given the contexts of the two texts. This is the same layout

as Pilehvar and Camacho-Collados (2018) for comparison's sake.[17] It is important to note that the weights of the binary classifier are updated separately from the VAE, thus no signal from the word sense disambiguation task is reaching the Z-variable encoder responsible for $Q(Z|\mathbf{x})$.

3.4.4 Experiment 4: Sentiment Analysis

Sentiment analysis was chosen as the second task to measure inference over continuous latent variable Z. The task of sentiment analysis here is like the word sense disambiguation task in that it requires the training of an additional classifier to be used after calculation of $Q(Z|\mathbf{x})$. It is unlike the three tasks above, however, in that it is not a word-level task, but a task over sentences, or sometimes longer passages. As this chapter makes use of the multilingual Amazon reviews corpus (Keung et al., 2020), sentiment analysis is done at the level of individual reviews, which can theoretically range from single words to longer passages of text consisting of multiple paragraphs.

Additional steps must be taken then to extract sentiment information at the reivew-level from Z, which is a variable containing lexical semantics of individual words. To extract a review-level representation from word-level representations, this chapter trains an additional transformer encoder on top of word-level representations. This is schematized in Figure 3.11. As in the task above, weights are updated separately for the encoder and classifier such that no signal from the sentiment analysis task is reaching the Z-variable encoder responsible for $Q(Z|\mathbf{x})$.

While the multilingual Amazon review corpus provides 200,000 training reviews for each language, this chapter makes use of only a 20,000 example subset for training; the full 5,000 reviews for the dev set and test set are used. Each data set in the corpus is equally weighted for reviews of different numbers of stars, i.e. the number of 1-star reviews is the same as the

17. The score from Pilehvar and Camacho-Collados (2018) was obtained using the BERT base model trained on English. MBERT as used here is also a BERT base model, but trained on 103 languages in addition to English. Thus, the two scores will not be perfectly comparable.

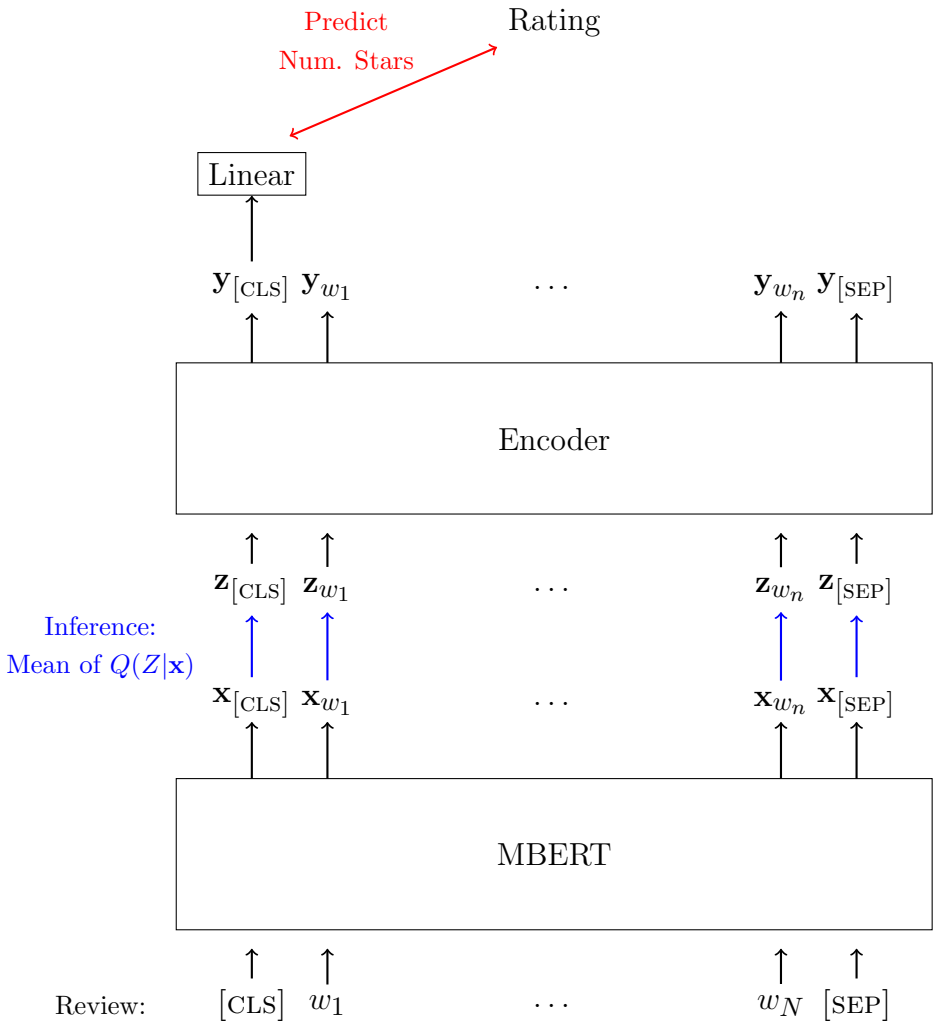


Figure 3.11: Schematic for sentiment analysis task given Amazon review. Weights updated only for encoder and linear layer.

number of 2-star reviews, which is the same as the number of 3-star reviews, and so on.

Finally, while neural networks are generally adept at the task of sentiment analysis, so-called fine-grained sentiment analysis, which goes beyond the simple negative-positive dichotomy, has proven to be a much more difficult task. Consider the Stanford Sentiment Treebank (SST) dataset of Socher et al. (2013) which consists of movie reviews and associated star ratings. This dataset comes in two varieties, one is a simple binary (negative-positive) classification task called SST-2, and the other a five-way classification called SST-5, which uses the same 1-5 rating system as the Amazon review corpus used here. While the BERT-Base model scores 91.2% when applied to SST-2, it achieves only 53.2% when applied to SST-5. As this is the case, I also show results for experiments using a modified Amazon review corpus where scores of 1 and 2 stars are deemed negative, 3-star reviews are discarded, and 4 and 5 star reviews are deemed positive.

3.5 Results

The results for the four experiments detailed above are found in Table 3.6.

Language	Morph. Tagging	POS Tagging	Sent. Analysis (1-5)	Sent. Analysis (Binary)	WiC
English*			38.6%	75.8%	55.6%
Galician*	68.2%	82.7%			
Spanish	89.1%	93.7%	39.1%	76.8%	

Table 3.6: Results on tasks involving inference over latent variables. Asterisk $*$ indicates all scores are in zero-shot regimen.

For the morphological tagging task, recall that variable M is constructed of seven distinct categorical variables, one for each of the morphological features of Case, Gender, Mood, Number, Person, Tense, and VerbForm. An example is calculated as correct for the purposes of reporting in Table 3.6 if and only if the correct feature value for each feature is inferred given an MBERT embedding. Under these conditions, Spanish reported a score of 89.1%, while Galician reported a score of 68.2% in a zero-shot training scenario. While the scores for Spanish are not state-of-the-art—compare this result against Table 2.7 from Chapter 2,

where Spanish averages well over 90% on average at morphological tagging—they show that morphological information has been successfully disentangled to a large degree. The results for Galician are unsurprisingly lower given that no Galician word embeddings, either labeled or unlabeled, were provided to the variational autoencoder for training.

The results of the syntactic inference task show a similar story, with Spanish reporting an accuracy of 93.7% and Galician showing 82.7%. State-of-the-art POS tagging generally approaches 97% on universal dependencies treebanks, meaning that while Spanish fails to report state-of-the-art scores under the proposed disentangled representation learning framework, its near state-of-the-art performance in concert with the favorable results from morphological feature tagging should be considered encouraging for the prospects of simultaneously disentangling multiple linguistic factors of variation from contextual word embeddings. Again, zero-shot Galician results lag significantly behind Spanish results, however accuracy above 80% is still indicative that in a large majority of cases, it is possible to infer syntactic information from MBERT embeddings in a language-neutral fashion.

On the Word-in-Context dataset, the system recorded a score of 55.6% accuracy on the English-only dataset. As the expected score for a random classifier is 50.0%, the latent variable Z can be said to contain some information with regard to lexical semantics. However, the score reported in Pilehvar and Camacho-Collados (2018) for the BERT base model is 60.2%, meaning that the autoencoder responsible for encoding Z is likely losing a good deal of the information originally present in the embeddings.

As discussed, two variants for the sentiment analysis task are considered. The first variant uses the Amazon review corpus as is, with ratings on a 1-to-5 scale. The second is the binary task, where neutral (i.e. 3-star) reviews are discarded, and the remaining reviews are lumped together as either positive or negative. For the fine-grained task, Spanish showed classification accruacy of 39.1%, while English results slightly trailed behind at 38.6%. Again, it is clear that some portion of sentiment information contained in MBERT embeddings is being captured in the Z latent variable, as the expected result were there no such information

would be 20% accuracy. However, Keung et al. (2020) report scores of 56.9% and 56.5% for Spanish and English respectively using MBERT, so clearly much of the information is being lost during encoding into Z. On the binary sentinment analysis task, the scores for Spanish and English jump significantly up to 76.8% and 75.8%.

Returning to the morphological tagging task, Table 3.7 shows the feature-level results for Galician and Spanish.

Language	Case	Gender	Mood	Number	Person	Tense	VerbForm	Total
Galician*	99.1%	78.2%	99.1%	85.4%	95.5%	97.8%	97.5%	68.2%
Spanish	99.8%	94.0%	99.1%	95.0%	98.8%	98.8%	98.5%	89.1%

Table 3.7: Feature-level results for morphological tagging task. Asterisk $*$ indicates all scores are in zero-shot regimen.

These results show that using variational autoencoders for disentangled representation learning is a viable means of extracting information at the level of morphological features. In the case of Spanish, the values for Case, Mood, Person, Tense, and VerbForm are all correctly inferred above 98% of the time, and only Gender is inferred correctly less than 95% of the time. Galician likewise scores very highly on a majority of the features, with Case, Mood, Person, Tense, and VerbForm all being correctly inferred more than 95% of the time. However, the features of Gender and Number perform significantly worse than the other features.

3.6 Conclusion

This chapter opened with a comparison between generative neural networks and generative grammar, noting that at the proper level of abstraction, the assumptions behind each of the enterprises are virtually identical. Playing on the analogy between the two, the chapter then designed a generative neural network based on assumptions from generative grammar and used the variational autoencoder framework to investigate means of distilling representations of word embeddings specific to different types of linguistic information. To do this, I applied

the technique of disentangled representation learning so as to factor vector representations of word occurrences into their assumed underlying sources of variation.

As MBERT word embeddings are models of words in context and this chapter assumes three different underlying sources of linguistic information for words in context in natural language, the model was designed to factor MBERT embeddings into three latent variables: a discrete morphological variable, a discrete syntactic variable, and a continuous lexical semantic variable. This entailed training separate encoding neural networks to extract morphological, syntactic, and lexical semantic information from word embeddings.

While this factoring of word embeddings is based in linguistic intuition, certain simplifications were made which are not in line with linguistic intuition. The most obvious simplification, as mentioned in footnote 16, has to do with the form of latent variable M. Recall that $Q(M|\mathbf{x}) = \prod_{feat \in feats} Q(M_{feat}|\mathbf{x})$, meaning that M actually consists of multiple individual M_{feat} variables, one for each morphological feature. The simplification is that given the definition of $Q(M|\mathbf{x})$ as the product of separate feature probability distributions, each of the individual feature probability distributions are assumed independent. Morphological features are clearly not independent in natural language, for example with Mood marking precluding Case marking.

In spite of any simplifications, the viability of factoring MBERT embeddings in such a way was shown by strong performance on inference tasks over each of the three latent variables in three languages. The tasks were morphological feature tagging, syntactic feature (i.e. POS) tagging, a variant of the word sense disambiguation task, and sentiment analysis. Morphological and syntactic tagging tasks were performed for Spanish and Galician, the word sense disambiguation task was performed using only English, and sentiment analysis was performed for Spanish and English. Training data in the form of both unlabeled MBERT embeddings, as well as MBERT embeddings with associated morphological and syntactic tags, were provided only for Spanish; all experiments for English and Galician were thus zero-shot experiments.

In all experiments, while scores fell short of state-of-the-art, they were well above random, meaning that while some of the information from the original embeddings was lost, the model was successfully able to disentangle disparate types of linguistic information into separate latent variables. Variational autoencoders as described in this chapter are thus a viable means of factoring continuous word embeddings, just as they've been shown useful for factoring high-dimensional representations of images in the computer vision literature.

It is hoped that this chapter will serve as impetus for researchers to explore new methods of applying linguistic assumptions to build and train neural networks, as the parallels between generative neural networks and generative grammar should prove a useful feeding ground for researchers going forward. Several potential avenues future of research present themselves, such as experimenting with different factorizations of latent variables (e.g. direct dependence of morphological variables upon syntactic variables), and latent variable models of sentences.

CHAPTER 4
PHONOLOGICAL AND PRAGMATIC INFORMATION IN KOREAN AFFIX EMBEDDINGS

4.1 Introduction

The previous chapters examined questions pertaining mainly to different BERT models' encoding of inflectional morphological features in Indo-European languages. They proceeded chiefly by investigating word embeddings for evidence of morphological information via classification tasks, and determined that in large part BERT-style models reflect this information in their word embeddings.

In contrast, this chapter focuses on BERT models' encoding of morphological information in Korean, a language typologically distinct from Indo-European languages which exhibits rigidly agglutinative morphology. The agglutinative nature of Korean provides two advantages when studying morphological information in embedding frameworks. The first is that there is largely a one-to-one mapping between morpheme and meaning/role which is absent in fusional languages common in the Indo-European family. The second advantage is related to the first. Agglutinative languages are more amenable to morphological parses, meaning in languages like Korean it is more feasible to directly examine a model's embedding of individual morphemes denoting some morphological feature of interest, rather than simply words marked for that feature. For example, suppose one is interested in tense marking. Whereas a BERT model's subword tokenizer might tokenize as atomic the English word *sang*, the Korean equivalent *noraehaessta*[1] might be tokenized as *norae-ha-ess-ta* (Gloss: *song*-DO-PST-DECL), allowing direct access to tense-marking morpheme *-ess*. This straightforward decomposability of words means that BERT's subword tokenization process in languages like Korean often provides embeddings for affixes which mark morphological features, rather

1. Romanized Korean in this chapter follows the Revised Romanization of Korean standard as set by the National Institute of Korean Language. See `https://www.korean.go.kr/front_eng/roman/roman_01.do`.

than simply words which are marked for those features. This chapter takes advantage of this fact and focuses on affix embeddings in Korean rather than word embeddings.

Another difference of this chapter with previous chapters is the nature of the experiments themselves. While the previous chapters primarily focused on performance on classification tasks, this chapter primarily uses visualization and data exploration techniques, specifically principal component analysis (PCA) and Self-Organizing Maps (SOMs). In doing so, it will be shown how the contextualized affix embeddings of a Korean BERT model—known as KoBERT[2]—reflect linguistic intuition and analysis with regard to two types of linguistic phenomena which have not yet been addressed in the BERTology literature. These types of phenomena are morphophonology, with Korean exhibiting a robust pattern of phonologically-driven allomorphy, and the pragmatic concepts of formality and honorifics, which Korean exhibits grammatically through affixation.[3] The overarching question behind the chapter is then to determine the extent to which KoBERT's affix embeddings can be said to behave like the Korean affixes they represent with regard to morphophonology and pragmatics.

4.2 Sources of Variation in Korean Affixes

As discussed in Edmiston and Kim (2019), Korean affixes are particularly suitable for the study of morphological information in embedding models due to (i) the aforementioned agglutinative nature of Korean, and (ii) the varied types of linguistic information they contain. With regard to the types of linguistic information that Korean affixes contain, they display phonological information in that Korean exhibits phonologically-driven allomorphy for a large portion of suffixes. Specifically, Korean displays vowel harmony (Finley, 2006) in affixation, as well as a host of other phenomena which show that affixes are sensitive to whether the stem they attach to ends in a consonant or not. Korean affixes also display syn-

2. `https://github.com/SKTBrain/KoBERT`

3. To my knowledge, the lone exception of work addressing morphophonological and pragmatic information in high-dimensional continuous embeddings is Edmiston and Kim (2019), which addresses these two phenomena—as well as certain syntactic and semantic aspects of Korean affixes—in static word embeddings.

tactic information in that they are amenable to being analyzed as the spell-out of syntactic heads (Koopman, 2005), and have different syntactic attaching sites (Cho and Sells, 1995). In addition, many Korean affixes perform semantic roles, such as discourse marking or focus marking. Finally, Korean affixes are used to express pragmatic concepts, such as formality and different levels of politeness. See Yeon and Brown (2019) for a detailed descriptive account of the usage of many Korean affixes, and Korean grammar in general.

4.3 Sourcing Affix Embeddings

The affix embeddings used in all experiments in this chapter come from the KoBERT model, a standard BERT model with 12 layers and 12 attention heads trained exclusively on Korean data. The embeddings used in experiments for this chapter were sourced either from embeddings of sentences from Korean's Wikipedia dump, or a collection of Korean subtitles.[4] The use of both academic language as found in Wikipedia, and the more colloquial style as found in subtitles, allowed for greater coverage of Korean affixes, particularly with regard to honorifics.

Like other BERT models, KoBERT makes use of the wordpiece algorithm for subword tokenization.[5] As the wordpiece algorithm has no explicit notion of morphology, KoBERT's tokenizer does not mark which tokens are stems and which are affixes, and an external morphological tagger was necessary. In order to identify and collect only those tokens which were of relevance to this study, I used the Mecab tokenizer and morphological analyzer (Kudo, 2006) trained on the Korean language, as available in the KoNLPy package (Park and Cho, 2014).[6] Among the 43 different tags the Mecab analyzer recognizes, I identified 18 which could be classified as affixes, and thus potentially relevant for this study. Of these 18, nine are nominal particles, such as case markers, five are verbal suffixes, such as mood

4. Subtitles sourced from `http://cineaste.co.kr/`.

5. Unlike Chinese or Japanese, modern Korean orthography separates words by space, thus no word-level tokenization is necessary.

6. The KoNLPy package is available at `https://konlpy.org/en/latest/`.

markers, and the remaining affixes are derivational affixes, such as light-verbs which attach directly to nominals in order to derive verbs. An example of use of the Mecab analyzer is as in Figure 4.1, with KoBERT's wordpiece sub-word tokenization included for reference. As can be seen in the example, the Mecab analyzer parses words into individual morphemes, and assigns each morpheme a part-of-speech tag, with the part-of-speech tag used to distinguish stems from affixes.

Input:		*naneun*	*dwaejigogiga*	*jeil*	*masitda*	
Morphological parse:		*na-neun*	*dwaejigogi-ga*	*jeil*	*masit-da*	
Gloss:		*I*-TOPIC	*pork*-NOM	*most*	*be.tasty*-DECL	
Mecab Output:		[(*na*, NP), (*neun*, JX)]	[(*dwaejigogi*, NNG), (*ga*, JKS)]	[(*jeil*, MAG)]	[(*masit*, VA), (*da*, EF)]	
Wordpiece Output:	[CLS]	_naneun	_dwae jigo giga	_je il	_mas itda	[SEP]
Translation:			"I like pork best."			

Figure 4.1: Example use of the Mecab morphological analyzer's part-of-speech tagger. Affixes are tagged in red.

As there exist homophones amongst Korean affixes, when collecting affix embeddings it is necessary to index them with their Mecab tag along with their orthographic form. For example, (*neun*, JX) is distinct from (*neun*, ETM), where the former is a topic marker and the latter is a relativizer. Fortunately, the Mecab analyzer provides excellent performance and is able to faithfully disambiguate such homophones in most cases.

Unfortunately, as can be seen in the divergence between **Mecab Output** and **Wordpiece Output** in Figure 4.1, KoBERT's tokenizer and the Mecab tokenizer do not always tokenize words in exactly the same way. For example, *naneun* "I-TOPIC" is tokenized as atomic by KoBERT, and thus no embedding for the topic marker *-neun* can be extracted in such examples. Nonetheless, agreement between the tokenizers is widespread enough and the corpora large enough to have been able to extract representative samples for each of the desired affixes.

In total, there were 103 different affixes identified by the Mecab analyzer as belonging to the 18 affix tags. Of these 103, 43 participated in either some form of phonologically-driven allomorphy, or were marked pragmatically as being either honorific or non-honorific. These

43 affixes were used in experiments. This resulted in 28,325 contextual affix embeddings in total. The number of representations for a particular affix was capped at 750, and the average number of contextualized embeddings for an affix was ≈ 641.28.

4.4 Experimental Methods

4.4.1 Principal Component Analysis—Redux

Section 2.3.2 introduced principal component analysis (PCA; Hotelling, 1933), a technique useful for multivariate data analysis which can also serve as a means of dimensionality reduction. Recall that in PCA, dimensionality reduction from $\mathbb{R}^p$ to $\mathbb{R}^d$, $p > d$, is achieved by projecting onto the d principal directions, which are the eigenvectors associated with the d largest eigenvalues of the data's covariance matrix. The first principal direction is the direction along which there is the most variance in the data, the second principal direction is the direction in the orthogonal complement to the first in which there exists the most remaining variance in the data, and so forth.

Visualization via PCA typically entails reducing dimensionality to $\mathbb{R}^2$ by projecting the original data points onto the first two principal directions. In this chapter, I project KoBERT embeddings to $\mathbb{R}^2$ in this fashion for visualization, coloring data points according to the behavior of the affix they represent with regard to the relevant linguistic phenomenon. For example, below honorific affix embeddings are visualized as red, and non-honorifics as blue. In addition, I also project KoBERT embeddings onto single dimensions, such as in Figures 4.2 and 4.3, where datapoints sampled from different distributions can be more easily contrasted. In projecting onto singular dimensions, I also project onto dimensions beyond the first two princpal components. This proves helpful in later experiments where it is shown that certain linguistic distinctions, while not evident in the first two principal components, become apparent when viewing projections onto later principal components. In such cases, the natural interpretation is that the embeddings under investigation do reflect the rele-

vant linguistic distinction, though that distinction is not one of the two principal drivers of variation in the dataset.

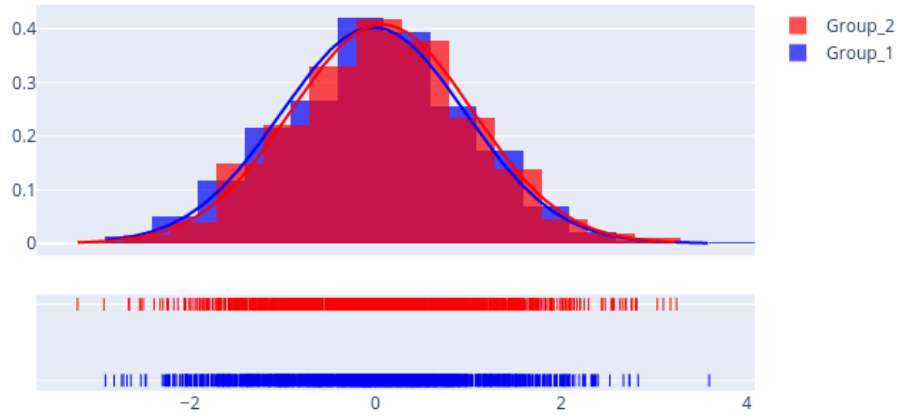


Figure 4.2: Example in which samples are drawn from two nearly identical distributions.

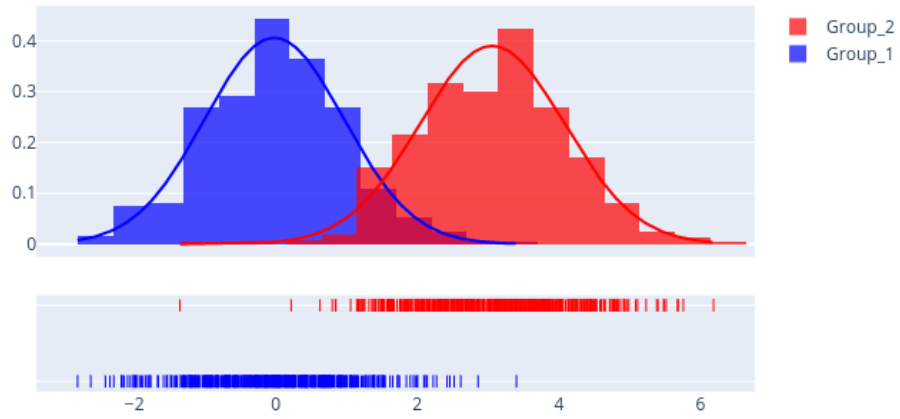


Figure 4.3: Example in which samples are drawn from two clearly distinct distributions.

4.4.2 Self-Organizing Maps

While PCA will serve to provide visualizations of affix embeddings at the scale of individual dimensions, Self-organizing (Feature) Maps (Kohonen 1982, 2012; SOMs) will serve to provide visualizations which take into account the full 768-dimensional affix embeddings. SOMs are a variant of unsupervised neural network, similar to single-layer feed-forward networks in architecture. However, rather than perform regression or classification, self-organizing maps seek to optimally represent potentially high-dimensional datasets in a low-dimensional discrete space. This is accomplished by a mixing of competitive learning (Ahalt et al., 1990) and cooperative learning (Liu and Djurdjanovic, 2008), as detailed below.

Specifically, SOMs seek to represent dataset $\mathcal{X} \in \mathbb{R}^{N \times d}$—$N$ data points in $\mathbb{R}^d$—with a network $\mathcal{V}$ given a user-specified topology. Typically, this topology is a 2-dimensional planar grid with x rows and y columns, resulting in $|\mathcal{V}| = x \cdot y$ nodes; Figure 4.4 ahead uses such a user-defined topology. Furthermore, associated with each node $v \in \mathcal{V}$ is a weight vector in $\mathbb{R}^d$, the same dimensionality as input datapoints $x \in \mathcal{X}$. These weight vectors are typically collected in a weight matrix W indexed by nodes $v \in \mathcal{V}$. The SOM algorithm is described in pseudocode in Algorithm 5. In a given epoch, for each data point x the algorithm identifies the nearest weight vector $W[v]$ in the high-dimensional space; this weight vector is known as the best-matching unit for x. This weight vector is then updated, along with all weight vectors in the neighborhood of v, to be more like datapoint x, where the neighbors of v are determined by the topology of $\mathcal{V}$. Generally, as the algorithm progresses through the epochs, the neighborhood function becomes more restrictive (Liu and Djurdjanovic, 2008). The learning rate α is also a function of epoch, decreasing through the epochs.

The competitive learning aspect of the algorithm—where only one node and its neighborhood are chosen to be updated—then results in weight vectors of dissimilar nodes in $\mathcal{V}$ receiving different updates in the high-dimensional space. The cooperative learning aspect of the algorithm—updating nodes together in the same neighborhood—encourages weight vectors of similar nodes in $\mathcal{V}$ to occupy similar regions in the high-dimensional input space.

Algorithm 5 Self-organizing Map

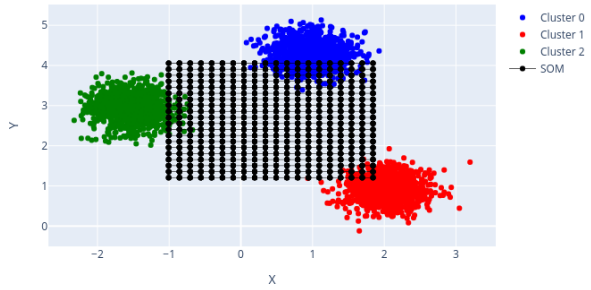
Input: points $\mathcal{X} \in \mathbb{R}^{N \times d}$, network $\mathcal{V}$
Initialize W ▷ $W \in \mathbb{R}^{|\mathcal{V}| \times d}$
for $epoch \in [1, ..., n\_epochs]$ **do**
    for $x \in \mathcal{X}$ **do** ▷ Iterate over data points
        $BMU = \underset{v \in \mathcal{V}}{argmin}(dist(x, W[v]))$ ▷ Find best-matching (i.e. closest) unit
        for $u \in Neighborhood(BMU, epoch)$ **do**
            $W[u] = W[u] + \alpha(epoch) \cdot (x - W[u])$ ▷ Update vectors closer to x

The result is then that the underlying topology of the high-dimensional dataset is reflected in the low-dimensional representation. This algorithm can be interpreted as a form of vector quantization, with high-dimensional datapoints being compressed into $|\mathcal{V}|$ clusters, where the relative relations between clusters are maintained.

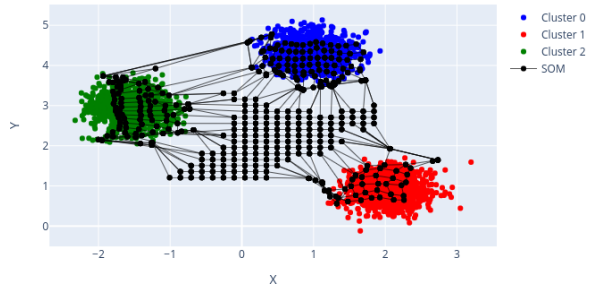
Self-organized maps can be most intuitively visualized when the dataset to be modeled is 2-dimensional. In such a case, weight vectors $W[v]$ themselves are 2-dimensional, and thus can be plotted and visualized directly. The learning process for such a 2-dimensional example can be seen unfolding in Figure 4.4. Here, $\mathcal{V}$ is a 20×20 grid and as the network learns, $\mathcal{V}$ begins to reflect the fact that the data are constructed from three clusters, where class—denoted by colors red, green, and blue—is perfectly correlated with cluster.

An alternative means of visualization, which is particularly useful when the input data are greater than 2-dimensional, is the so-called U-Matrix. The U-Matrix is a matrix representation of an SOM where each entry in the matrix represents a distinct node $v \in \mathcal{V}$. For example, the U-Matrix for the SOM in Figure 4.4 would be a 20×20 matrix, as the SOM's topology is that of a 20×20 grid. The value of the entry in the U-Matrix which represents node v is the average Euclidean distance between the weight vector of v, $W[v]$, and its neighbors' weight vectors $W[u], u \in neighborhood(v)$. Figure 4.5 shows the U-Matrix of the SOM from Figure 4.4 after Step 6. This representation shows that the SOM has learnt the underlying three-clustered structure of the data, with each of the three clusters represented by a dark patch.

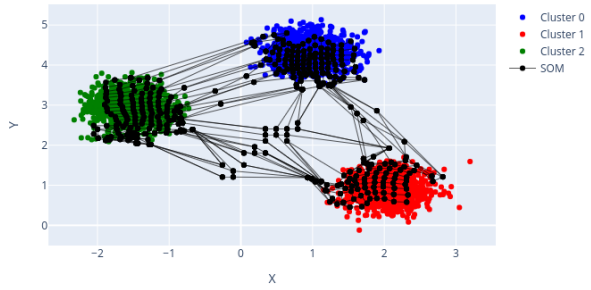
As the example in Figure 4.4 is 2-dimensional, which nodes are influenced by points



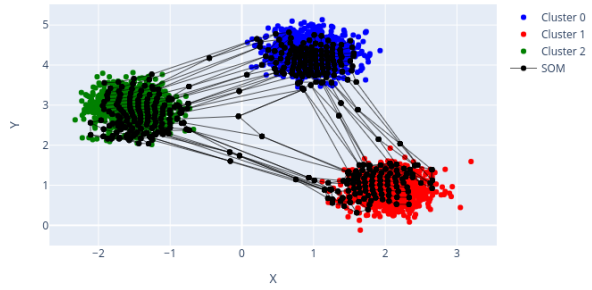
(a) Step 1: Initialization



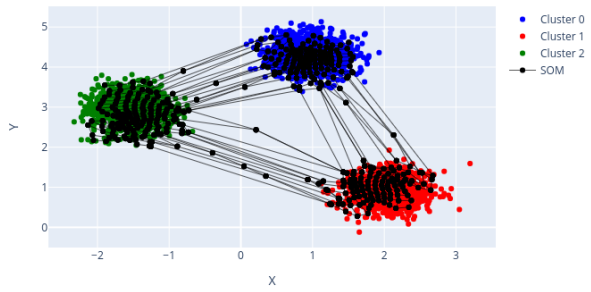
(b) Step 2



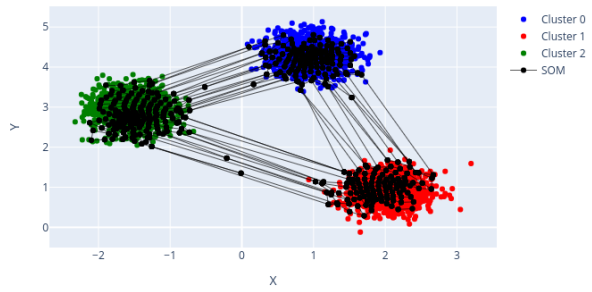
(c) Step 3



(d) Step 4



(e) Step 5



(f) Step 6

Figure 4.4: Self-organizing Map example where dataset consists of three clusters and is 2 dimensional.

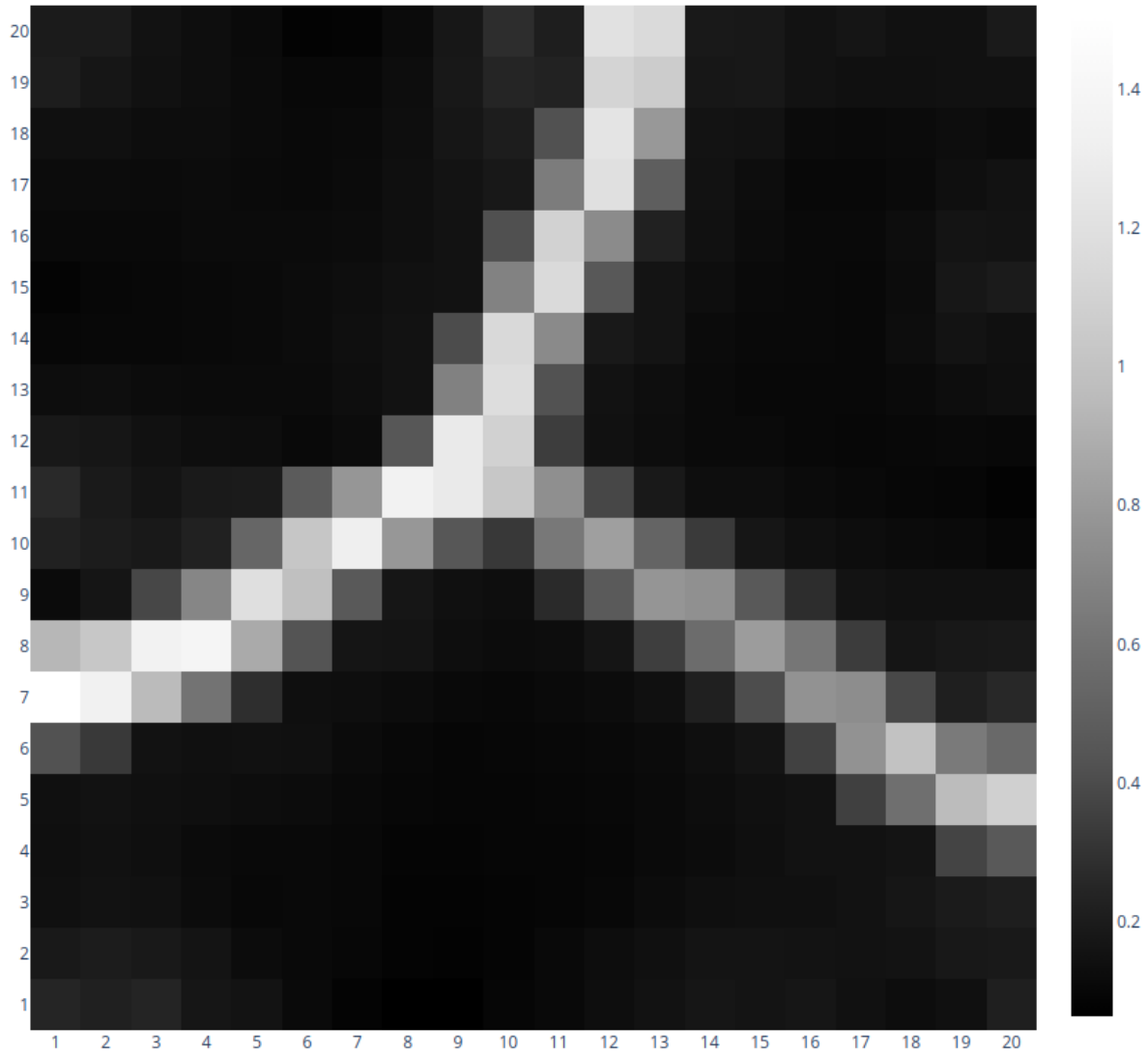


Figure 4.5: U-Matrix representation of self-organizing map from Figure 4.4 after training. Greyscale coloring reflects average Euclidean distance to neighboring nodes.

belonging to which class is evident via visualization of the weight vectors. For instance, nodes on the far left are influenced by the green cluster. However, in higher dimensions visualizations like Figure 4.4 become impossible, and while U-Matrix-style visualizations allow visualization from underlyingly higher dimensional data, they only provide a high-level description of the data's topology; U-Matrix visualizations like Figure 4.5 do not provide any information on the class membership of the data points which influence the weight vectors.

In other words, they do not show whether the identified clusters are "pure" with regard to class distinctions of the datapoints.

As such, the SOM visualizations used in this chapter to describe high-dimensional affix embeddings add additional information to the U-Matrix figures. This information takes the form of additional markers for each node v in the U-Matrix representation. The color of a node's marker signifies which class the datapoints influencing a node's weight vector belong to. For example, if a node is the best-matching unit for only datapoints of the green class, then that node's marker will be colored green, whereas if it is the best-matching unit for datapoints of the red and green classes, it will have both red and green markers. The size of a marker details how many datapoints of a given class that node's weight vector is a best-matching unit for. For example, if a node is a best-matching unit for many datapoints of the green class, it will have a relatively large green marker. An example taken from a three-cluster dataset in $\mathbb{R}^5$ is as in Figure 4.6. This figure correctly shows that the data can be partitioned into three distinct clusters as there are three distinct dark patches. In addition, the node markers show that each of these clusters is highly correlated with a different class; one cluster is dominated by the blue class, one by the red class, and one by the green class.

For comparison's sake, consider an example of the same 5-dimensional dataset with three clusters but in which the classes have been shuffled and thus are not correlated with the topology of the dataset. Such an example is as in Figure 4.7. Note that each of the three classes appears indiscriminately throughout the U-Matrix representation, with each class frequently appearing in each of the three clusters.

Below, SOMs will be used to perform visualizations of 768-dimensional contextual affix embeddings as produced by KoBERT, and classes for each of the embeddings will be assigned based on either the morphophonological or pragmatic properties of the affix. If such properties are reflected in KoBERT's embedding space, we expect the topology of the space as reflected by the U-Matrix to be correlated with the classes, and the visualization should resemble Figure 4.6 with distinct regions dominated by different classes. If, however, such

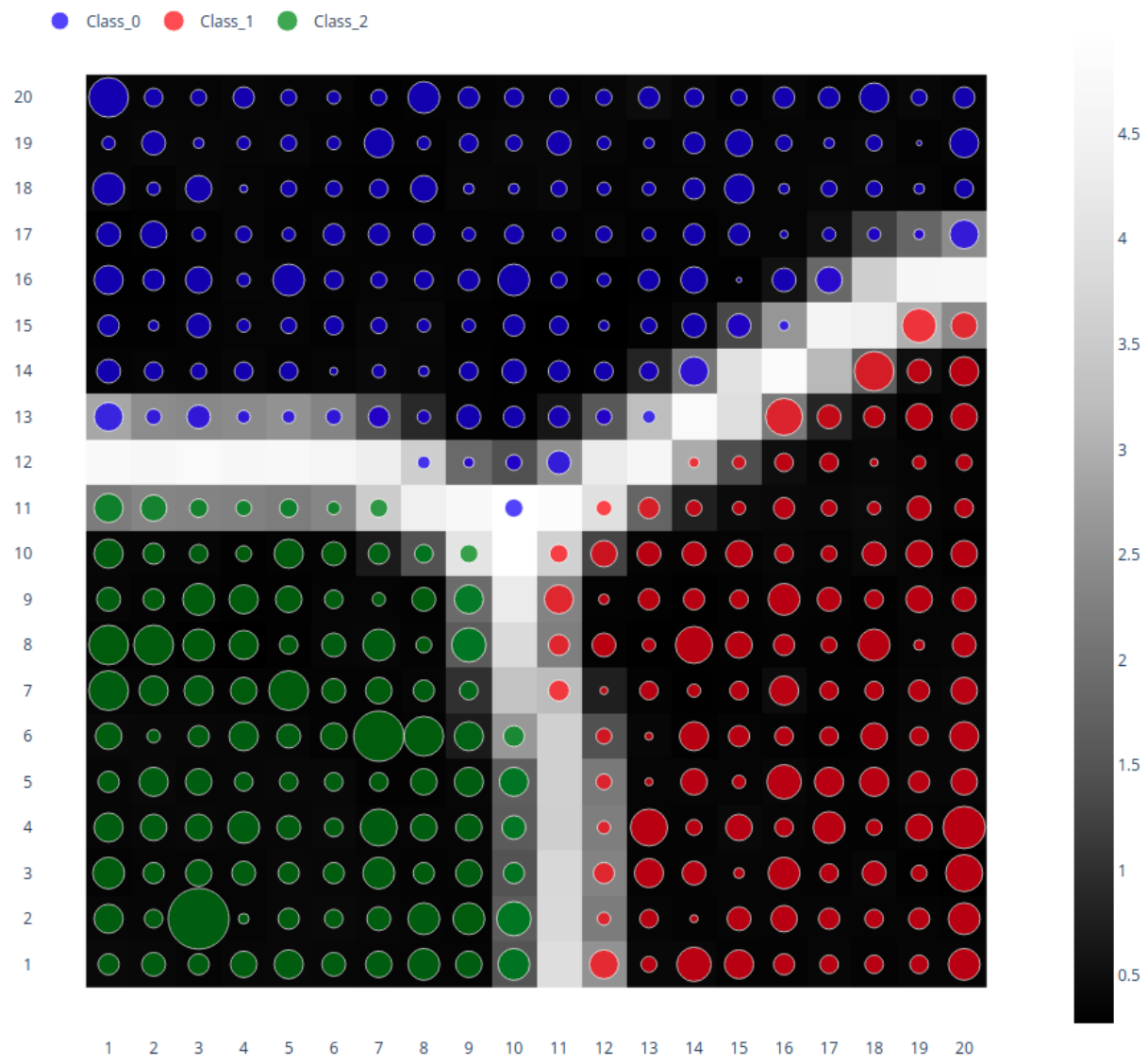


Figure 4.6: U-Matrix representation of Self-organizing map trained on 5-dimensional data. Underlying data generated from three neatly separable clusters.

properties are not well reflected in KoBERT's embedding space, we expect the topology of the space to be uncorrelated with the classes, and the visualization should resemble Figure 4.7, with significant intermixing of classes.

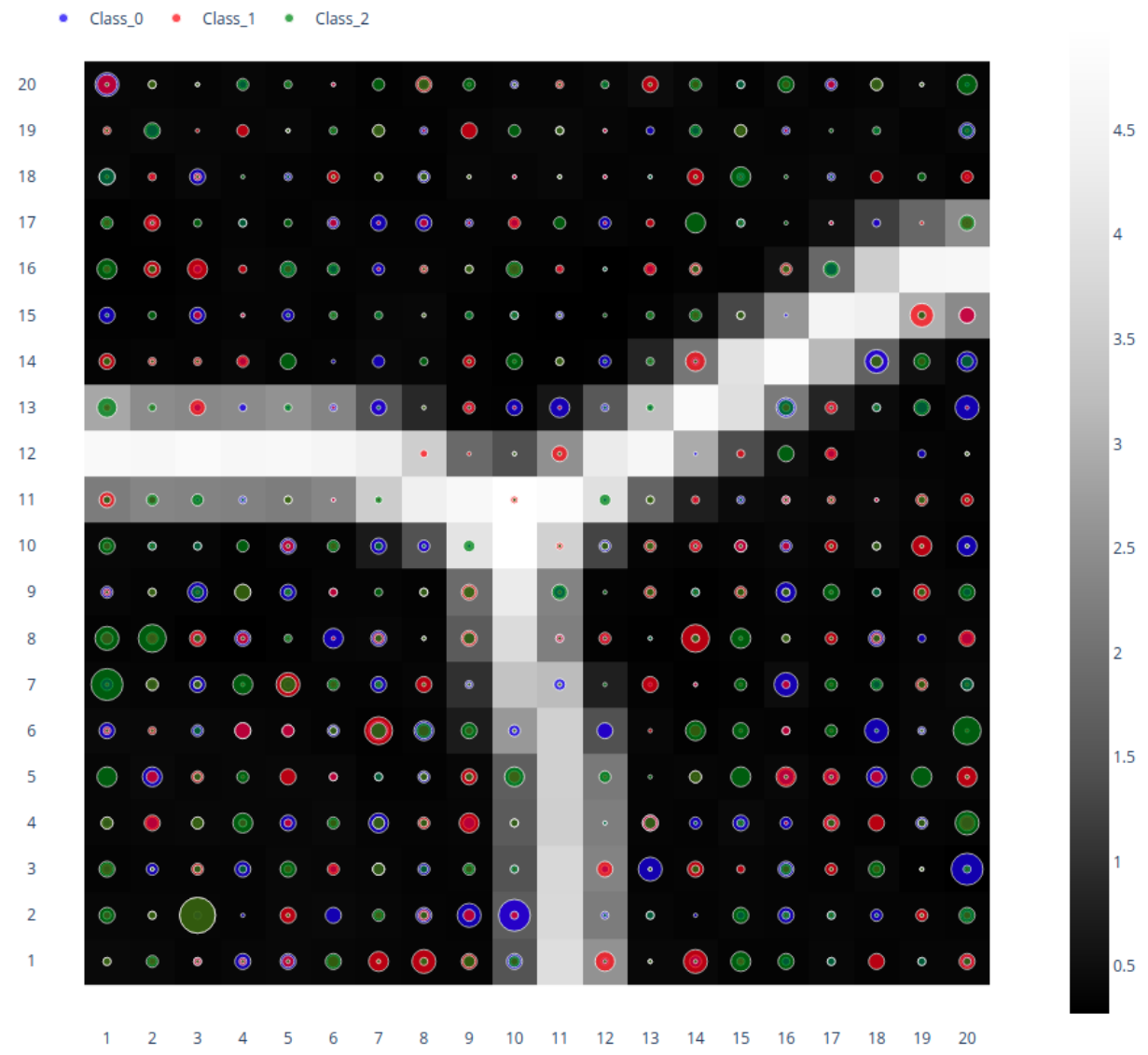


Figure 4.7: U-Matrix representation of Self-organizing map trained on 5-dimensional data. Underlying data generated from three non-separable clusters.

4.5 Phonology Experiments

4.5.1 Phonology: Phonologically-driven Allomorphy

Affixation in Korean exhibits phonologically-driven allomorphy for a large number of suffixes, with different sources driving this allomorphy. As an illustrative example of such phonologically-driven allomorphy, consider the purposive affix *-reo* and verb stems *sa* 'buy' and *meok* 'eat'. The purposive affix when attached to these verb stems results in *sa-reo*

"to buy" and *meog-eureo* "to eat". Depending on phonological context, affix *-reo* then either presents with an epenthetic '*eu*' or not.[78] Epenthesis here is driven by the presence of a stem-final consonant. In addition to epenthesis, the presence/absence of a stem-final consonant also decides other allomorphic pairs, such as nominative case markers *-i*/*-ga*, the former of which attaches to consonant-final stems, and the latter vowel-final stems. I refer to this alternation as the stem-final C-V alternation going forward.

Korean suffixes can then be divided into three groups with regard to the stem-final C-V alternation. The first is those suffixes which do not participate in the this C-V alternation; the exponents of these suffixes do not depend on whether the stem to which they attach ends in a vowel or consonant. The second is those suffixes which attach to stems ending in consonants, which I call C-suffixes, and the third is those suffixes which attach to stems ending in vowels, V-suffixes.[9] Table 4.1 lists several suffix pairs which participate in this allomorphy, divided into particles which attach to nominals and sentential connectives which attach to verbs.

In addition to the stem-final C-V alternation, Korean also exhibits phonologically-driven allomorphy by displaying vowel harmony for certain of its verbal suffixes (Lee, 1993). Vowels in Korean are grouped into so-called light vowels (/æ/, /ö/, /a/, /o/), dark vowels (/e/, /ü/, /ə/, /u/), and neutral vowels (/i/, /ɨ/). So-called light-vowel suffixes attach to stems whose final vowel is light, and dark-vowel suffixes attach to stems whose final vowel is either dark or neutral. Table 4.2 houses the vowel harmony allomorphic pairs relevant for the experiments in this chapter.

7. See Kim (2006) for a generative account of such alternations.

8. In fact, there is a third variant *-leo* which appears when the verb stems ends in /l/, however this variant is not evident in the orthography and will not be considered here.

9. This is a slight simplification, as certain suffixes are bifurcated not by whether they attach to stems ending in consonants or stems ending in vowels, but rather by whether they attach to stems ending in non-liquid consonants or other. *-reo* discussed above is such an example.

Suffix Type	Gloss	C-SUFFIX	V-SUFFIX	Examples
Particles	Nominative	*-i*	*-ga*	*saram-**i***, *chingu-**ga***
	Accusative	*-eul*	*-leul*	*saram-**eul***, *chingu-**leul***
	Ablative	*-eurobuteo*	*-robuteo*	*saram-**eurobuteo***, *chingu-**robuteo***
	Instrumental	*-euro*	*-ro*	*saram-**eoro***, *chingu-**ro***
	Topic	*-eun*	*-neun*	*saram-**eun***, *chingu-**neun***
	"as"	*-euroseo*	*-roseo*	*saram-**eoroseo***, *chingu-**roseo***
	"and"	*-kwa*	*-wa*	*saram-**kwa***, *chingu-**wa***
	"with"	*-kwa*	*-wa*	*saram-**kwa***, *chingu-**wa***
	conjunctive	*-ina*	*-na*	*saram-**ina***, *chingu-**na***
	auxiliary	*-ina*	*-na*	*saram-**ina***, *chingu-**na***
Connecting suffixes	"because"	*-euni*	*-ni*	*meok-**eoni***, *ha-**ni***
	"but"	*-euna*	*-na*	*meok-**euna***, *ha-**na***
	"if"	*-eumyeon*	*-myeon*	*meok-**eumyeon***, *ha-**myeon***
	"while"	*-eomyeonseo*	*-myeonseo*	*meok-**eumyeonseo***, *ha-**myeonseo***
	"and"	*-eumyeo*	*-myeo*	*meok-**eumyeo***, *ha-**myeo***
	"in order to"	*-euryeo*	*-ryeo*	*meok-**euryeo***, *ha-**ryeo***

Table 4.1: List of pairs of suffixes participating in the stem-final C-V alternation, i.e. which display allomorphy depending on stem-final phonemes. C-SUFFIX refers to suffixes which attach to stems ending in consonants. V-SUFFIX refers to suffixes which attach to stems ending in vowels. Nominal stems *saram* and *chingu* mean *person* and *friend* respectively; Verbal stems *meok* and *ha* mean 'eat' and 'do' respectively.

Gloss	LIGHT-SUFFIX	DARK-SUFFIX	Examples
Past	*-ass*	*-eoss*	*mag-**ass**-ta*, *meog-**eoss**-ta*
Connecting suffix	*-a*	*-eo*	*mag-**a** beoryeo, meog-**eo** beoryeo*
Sentence-final ending	*-a*	*-eo*	*ar-**a**, iss-**eo***

Table 4.2: List of pairs of suffixes participating in vowel harmony. The pairs in rows 2 and 3 are homophones, but differ syntactically in important ways, e.g. the connective suffix is strictly non-finite, while the opposite is true for the sentence-final ending. Stems *mak*, *meok*, *ar*, and *iss* mean *block*, *eat*, *know*, and *to.exist* respecitively.

4.5.2 Experiment Design

As discussed above, Korean displays two types of phonologically-driven allomorphy: stem-final C-V allomorphy, and vowel harmony allomorphy. In the first case, suffixes are divided into groups which attach to stems ending in consonants and stems ending in vowels. In the case of vowel harmony, suffix pairs are divided into groups which attach to stems whose final vowel is either dark/neutral, or stems whose final vowel is light.

In order to test whether KoBERT is sensitive to such morphophonological phenomena, I assume it is enough to show that the model's embeddings of these suffixes can be bifurcated

in the continuous space along the same lines as displayed in Tables 4.1 and 4.2. Experiments in this subsection consist of experiments on nominal particles participating in the stem-final C-V alternation, experiments on connecting suffixes participating in the stem-final C-V alternation, and suffixes participating in vowel harmony.

In each of the three cases, embeddings for all participating suffixes are assembled into a point cloud and assigned to one of two clusters, according to whether they are C-suffixes or V-suffixes in the case of the stem-final C-V alternation, or according to whether they are light vowel suffixes or dark/neutral vowel suffixes in the case of the vowel-harmony alternation. The point-cloud is then visualized using principal component analysis so as to provide an easily interpretable visualization of the data, and also determine if the relevant phonological distinction is evident in the first two principal axes of variation. In the affirmative case, one can interpret this as meaning that the allomorphy is a significant driver of variation in the embeddings. In the event the groups are not easily distinguished by projection onto the first two principal components, projection onto other principal components are also visualized. Subsequent experiments train self-organizing maps on the same point-clouds, with the results visualized as above in Figure 4.6. That is, the resulting U-Matrix of the SOM is visualized, and each node in the U-Matrix is colored according to the morphophonological behavior of the data points which influence that node's weight vector.

4.5.3 Results

C-V Alternation: Particle Subset

Table 4.1 identifies ten different nominal particles which can be divided into C-suffix forms and V-suffix forms, resulting in twenty different affixes as identified by Mecab. A point-cloud consisting of contextualized embeddings for all twenty affixes is projected to two dimensions via PCA, and colored according to their either being C-suffixes or V-suffixes. This visualization is in Figure 4.8.

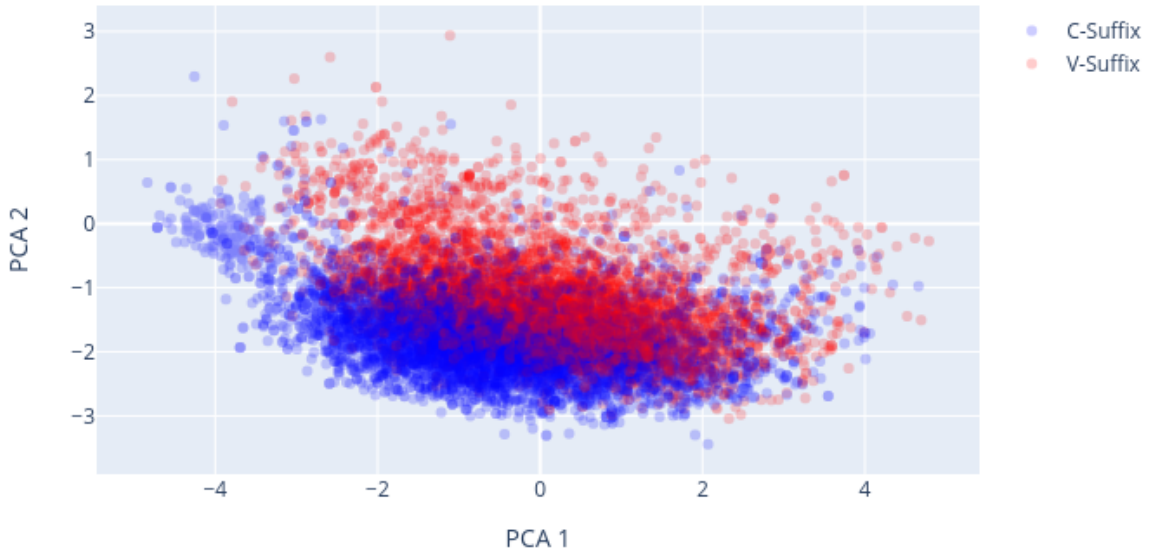


Figure 4.8: PCA visualization of nominal particle allomorphs.

From this visualization, it appears that there is significant overlap between the two groups of affixes along the first two principal directions, with the red and blue point clouds showing significant overlap. The fact that there is little difference between the two affix groups in the first two principal directions is significant, as it means that allomorphy in this case is not one of the two principal drivers of variation for KoBERT embeddings. Figures 4.9 and 4.10 better illustrate this by providing visualizations of the distributions—fit with normal curves—of each suffix group along the first two principal directions individually.

However, just because there is little difference between the groups along the first two principal directions does not mean that KoBERT is insensitive to the allomorphy. In fact, visualization of projection of the affix embeddings onto the fourth principal direction shows that indeed KoBERT *does* make the distinction, and robustly so. This is visualized in Figure 4.11.

Further evidence of KoBERT's sensitivity to this morphophonological pheonomenon

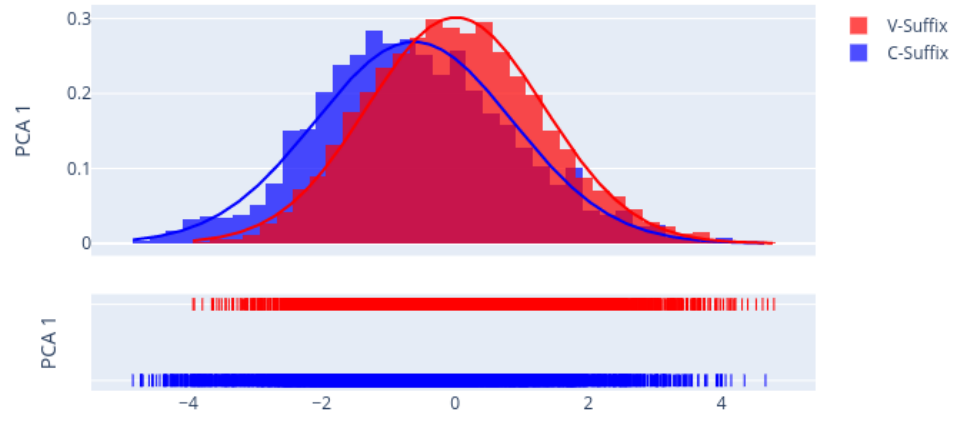


Figure 4.9: Visualization of particle embeddings projected onto the first principal direction.

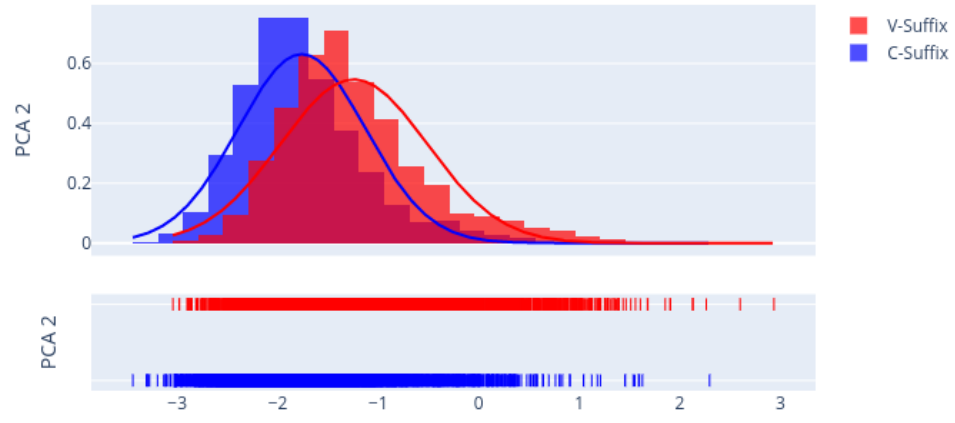


Figure 4.10: Visualization of particle embeddings projected onto the second principal direction.

comes from SOM visualization, which unlike PCA, deals with the full 768 dimensional representations. Such a visualization is as in Figure 4.12. This visualization shows that KoBERT's high-dimensional space can be divided neatly along morphophonological lines

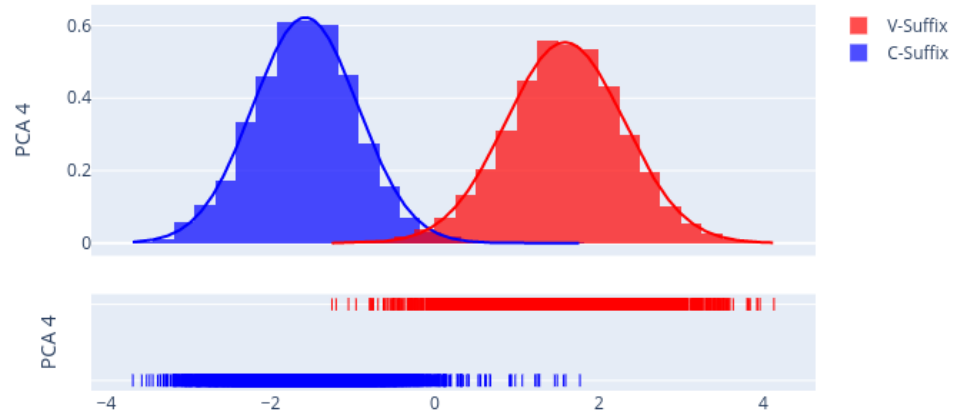


Figure 4.11: Visualization of particle embeddings projected onto the fourth principal direction.

in the case of Korean nominal particle embeddings, as the C-Suffix/V-Suffix distinction is highly correlated with the topology discovered by the SOM. Half of the map is dominated by blue nodes, and the other half by red nodes, with very virtually no intermixing between the classes.

C-V Alternation: Connective Subset

Table 4.1 identifies six different verbal connective suffixes which can be divided into V-suffix forms and C-suffix forms, resulting in twelve different affixes as identified by Mecab. Again, a point-cloud consisting of contextualized embeddings of these twelve forms is visualized via PCA as in Figure 4.13, revealing that again, there is significant overlap between the affix groups. Figures 4.14 and 4.15 display the distributions of each suffix group along the first two principal components as before.

Interestingly, again it is projection onto the fourth principal component which best shows that KoBERT is sensitive to the C-V alternation. This is visualized in Figure 4.16. Also as

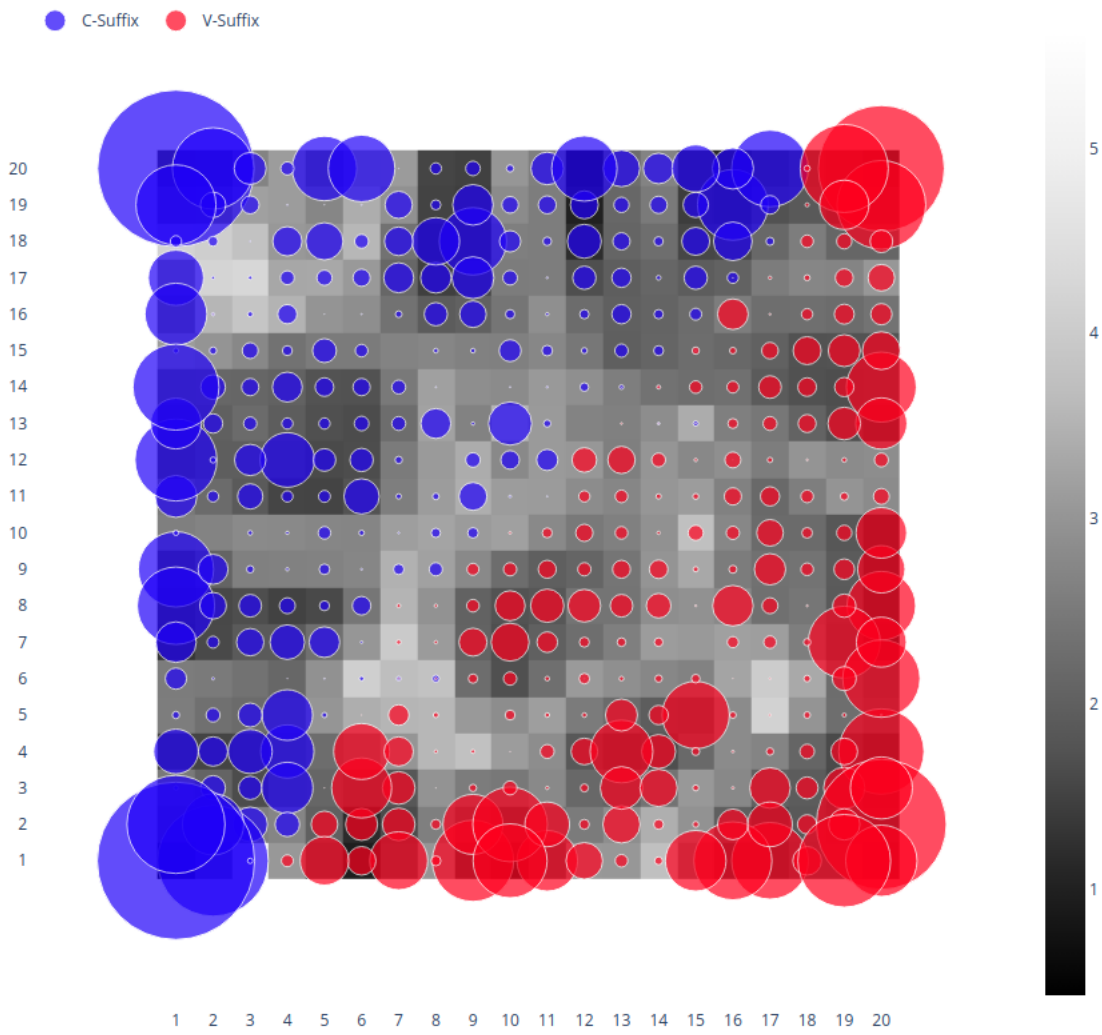


Figure 4.12: SOM visualization of nominal particle allomorphs.

above, the U-Matrix visualization of the SOM in Figure 4.17 shows that KoBERT embeddings of verbal connective suffixes in Korean are just as separable along morphophonological lines as the nominal particles discussed above, with the node markers showing that half of the space is dominated by C-Suffix affixes, and the other half by V-Suffix affixes.

It appears then that we can answer in the affirmative that BERT-style models like KoBERT are sensitive to morphophonological variation when it is evident in the orthography, at least in the case of the stem-final C-V alternation. In both subsets of affixes discussed for this phenomenon, it is possible to partition KoBERT's embedding space, placing C-suffixes

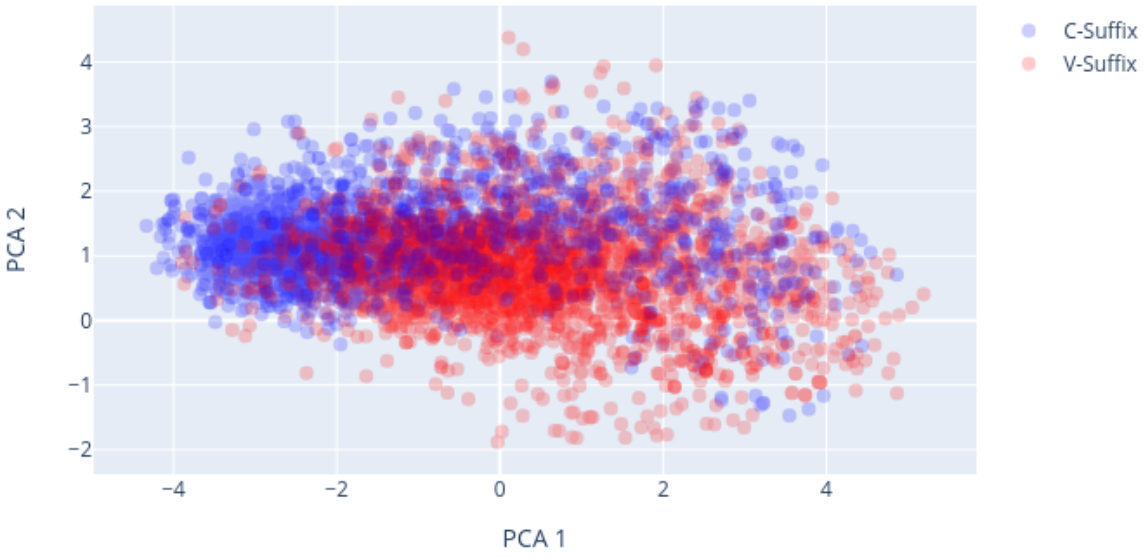


Figure 4.13: PCA visualization of connective particle allomorphs.

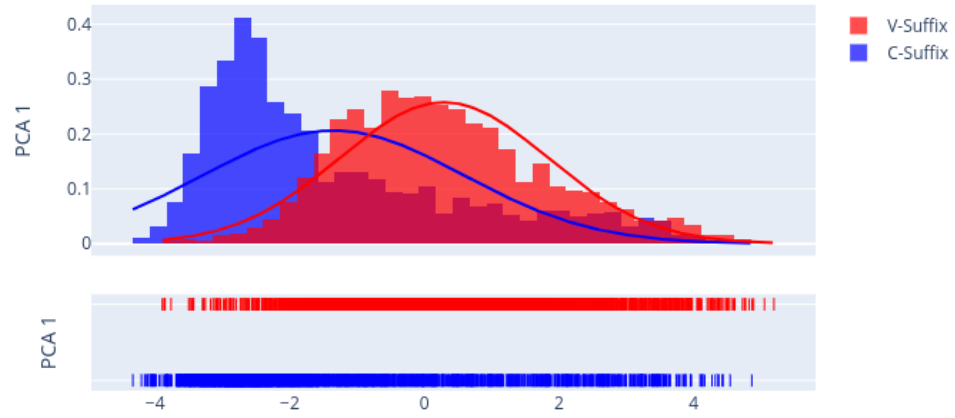


Figure 4.14: Visualization of connective embeddings projected onto the first principal direction.

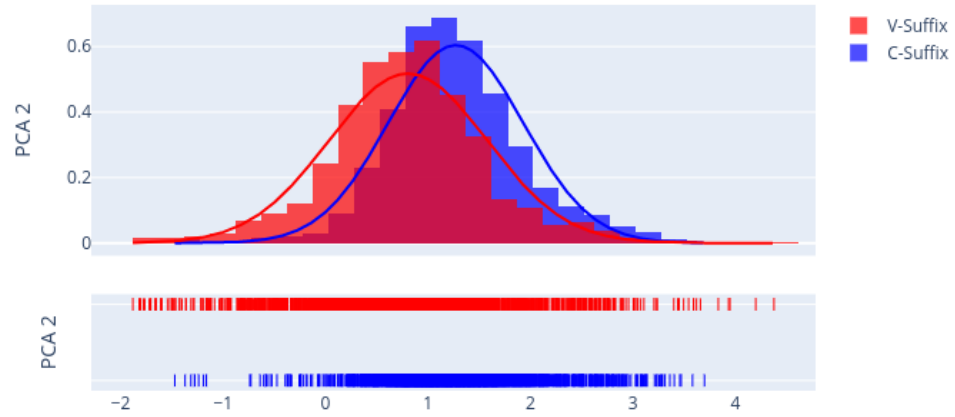


Figure 4.15: Visualization of connective embeddings projected onto the second principal direction.

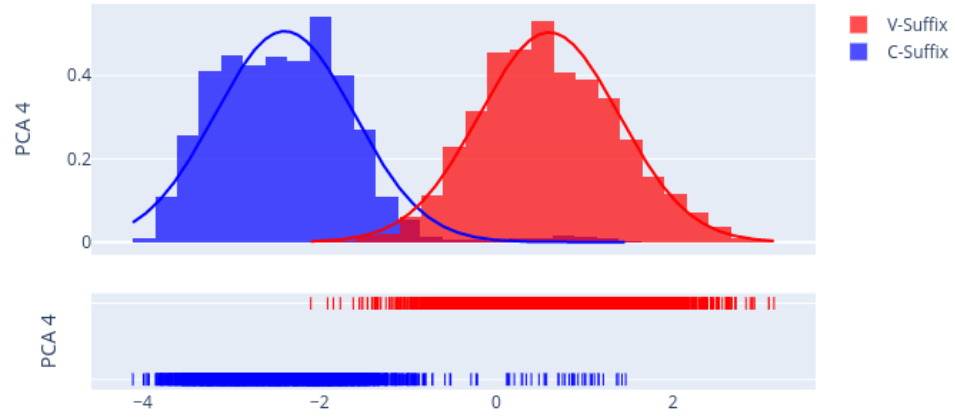


Figure 4.16: Visualization of connective embeddings projected onto the fourth principal direction.

and V-suffixes on opposing ends of the space.

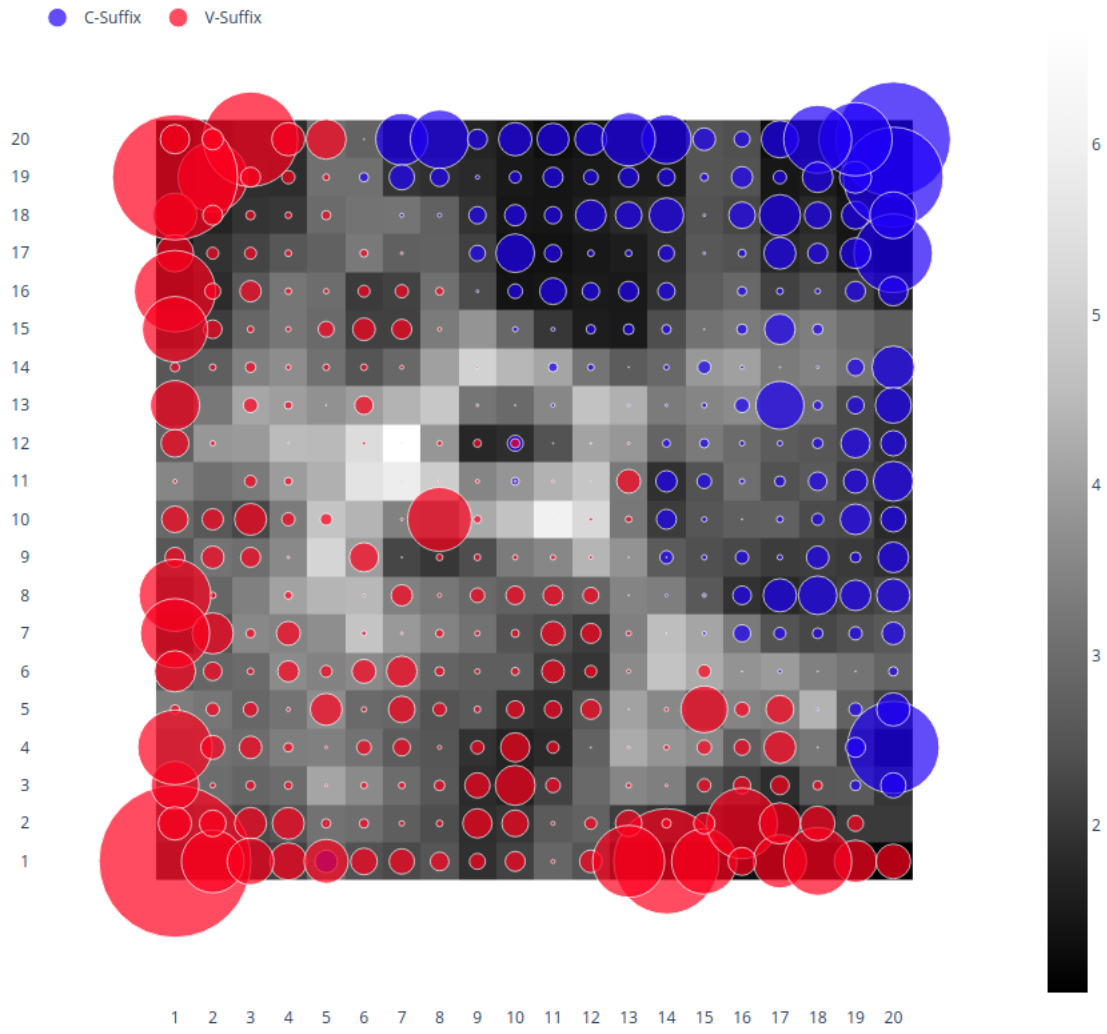


Figure 4.17: SOM visualization of verbal connective allomorphs.

Vowel Harmony

Table 4.2 lists three different allomorph pairs which alternate according to vowel harmony. This results in six different affixes as identified by Mecab. Again, contextualized embeddings are gathered for these affixes and assembled into a point-cloud, which are then visualized by PCA. This visualization is as in Figure 4.18, with distributions on individual principal directions in Figures 4.19 and 4.20.

Much as in the case of the C-V alternation above, there appears to be little difference between the two groups of affixes with regard to the first and second principal components.

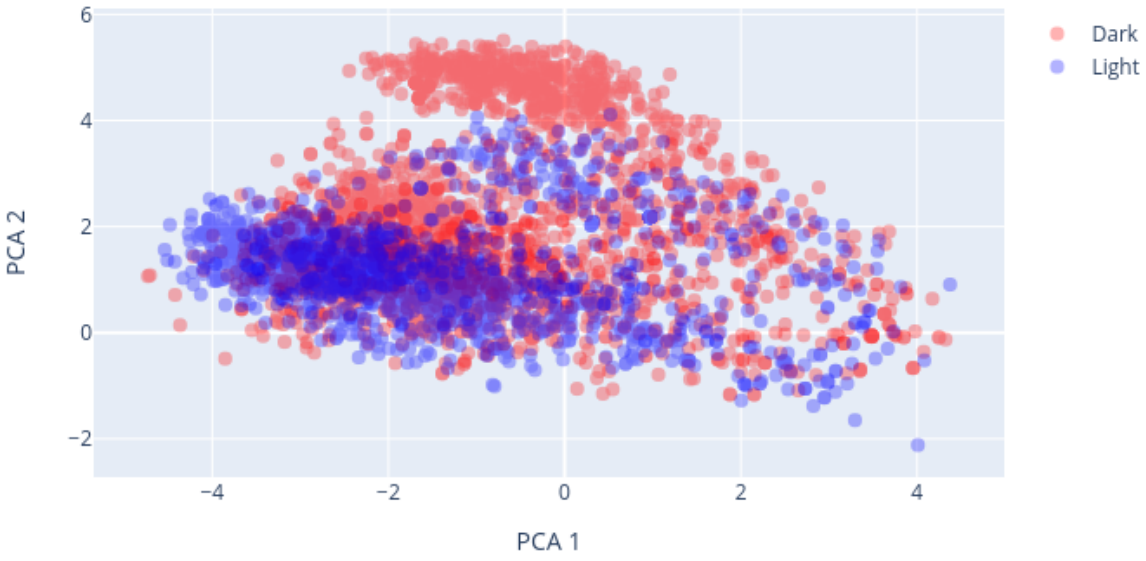


Figure 4.18: PCA visualization of vowel harmony allomorphs.

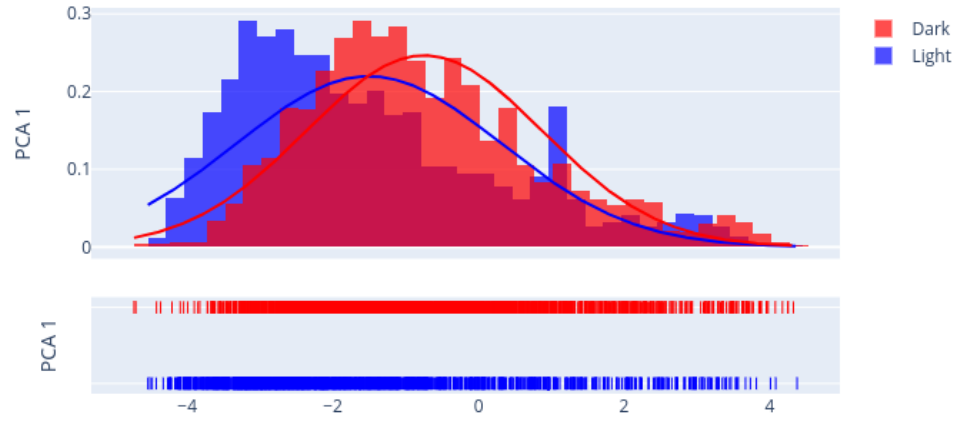


Figure 4.19: Visualization of vowel harmony-participating embeddings projected onto the first principal direction.

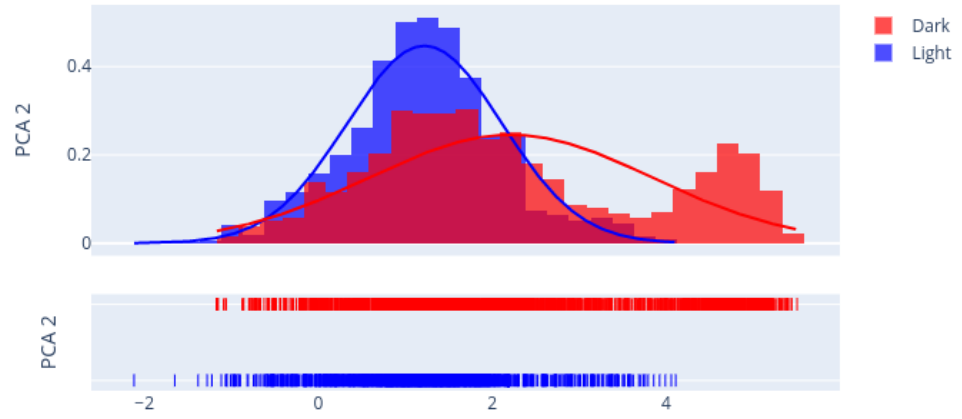


Figure 4.20: Visualization of vowel harmony-participating embeddings projected onto the second principal direction.

Also as in the cases above, however, projecting onto later principal components does reveal dimensions along which the relevant distinction appears evident. In this case, it is the tenth principal component, which is visualized as in Figure 4.21.

Again, SOM visualization reveals that in the underlying high-dimensional space the difference between the two affix groups is significant. This visualization is as in Figure 4.22.

It is worth noting in the SOM visualization that while the underlying space appears bifurcated along the relevant morphophonological distinction, the division of the space appears less neat, with a majority of the space being devoted to the dark-vowel suffixes. This is perhaps to be expected, as dark-vowel suffixes attach to stems ending in either dark-vowels *or* neutral vowels, while light-vowel suffixes attach only to stems whose last vowel is a light-vowel. Finally, whereas the C-V alternation appears evident at the fourth principal direction, vowel harmony appears evident only at the tenth principal direction and the distinction is less clear. The conclusion which can be drawn here is that KoBERT, while apparently sensitive to the vowel-harmony distinction in Korean, is less so than it is the stem-final C-V distinction.

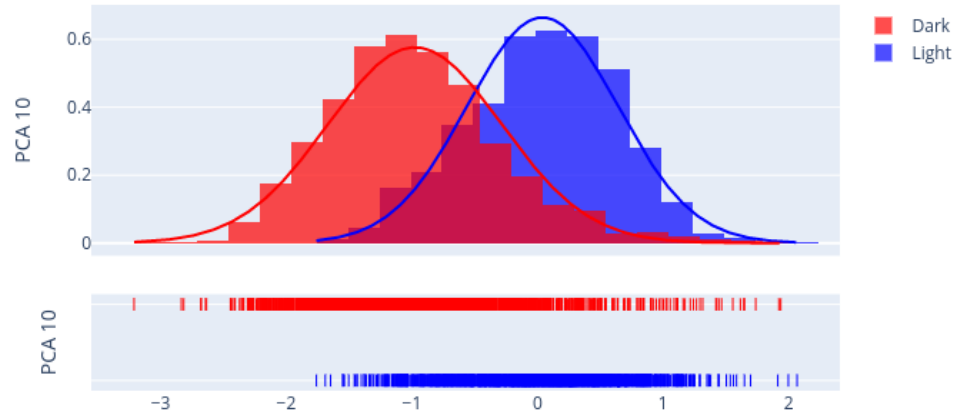


Figure 4.21: Visualization of vowel harmony-participating embeddings projected onto the tenth principal direction.

4.6 Pragmatics Experiments

4.6.1 Pragmatics: (Non-)Honorifics

One of the defining features of the Korean language is its elaborate system of pragmatic affixes, which reflect social relationships between not only the speaker and hearer, but also amongst speaker and third parties, as well as hearer and third parties (Brown, 2015). Amongst the affixes responsible for conveying such pragmatic informaiton, certain affixes attach to nominal stems, such as honorific case markings, while others attach to verbal stems, for example mood markers specifically marked for differing levels of politeness or formality. While there are different ways of classifying so-called Korean Speech-levels, one influential method is as in Figure 4.3, first proposed by Sohn (2001).

This figure identifies six different speech-levels. These six speech-levels are split between honorifics and non-honorifics. This chapter focuses on this binary split, distinguishing only between honorific and non-honorific.

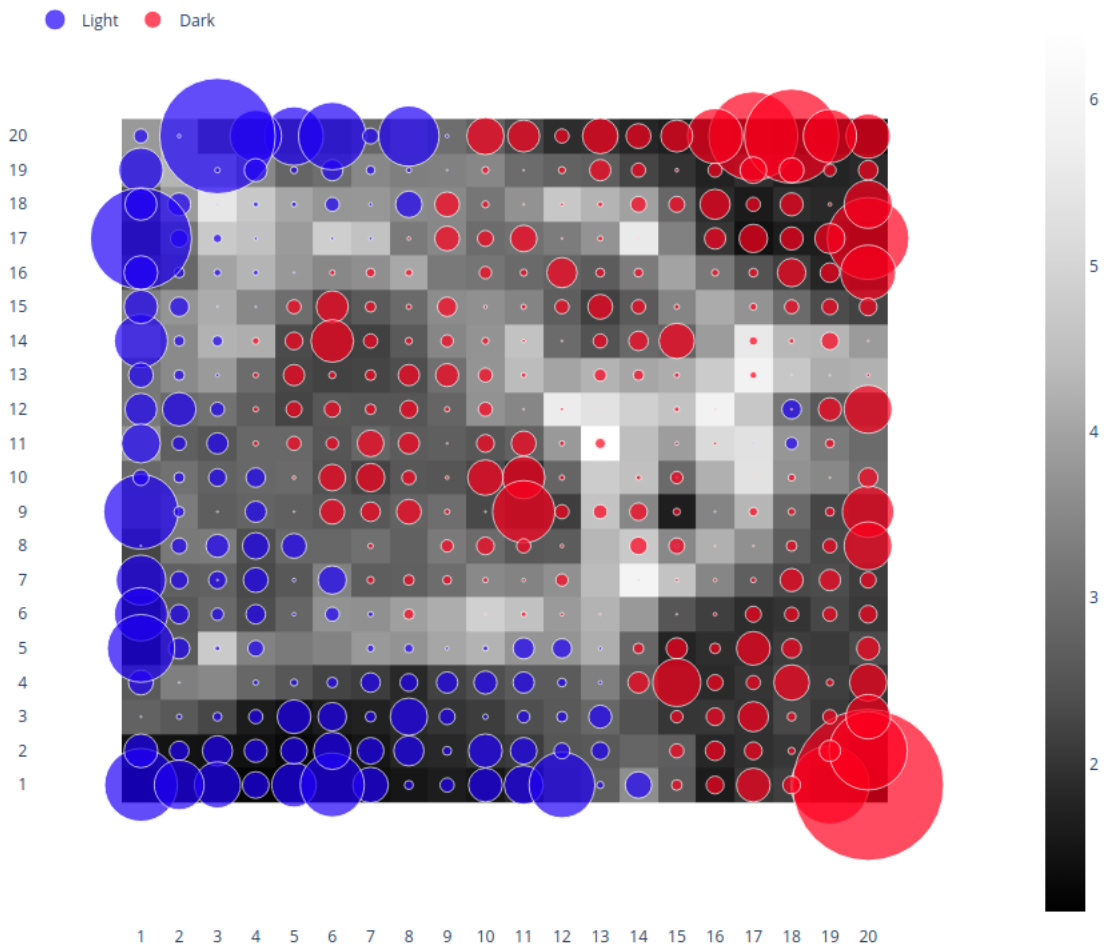


Figure 4.22: SOM Visualization of vowel harmony allomorphs.

	Speech-level	Example
Honorific	Deferential	*meok-**seupnida***
	Polite	*meok-**eoyo***
Non-honorific	Blunt	*meok-**so***
	Familiar	*meok-**ne***
	Intimiate	*meok-**eo***
	Plain	*meok-**neunda***

Table 4.3: The six Korean speech levels per Sohn (2001). Adapted from Figure 2 of Strauss and Eun (2005).

4.6.2 Experiment Design

Four pairs of affixes are identified for the pragmatics experiments in this chapter, and are listed in Table 4.4. According to the classification system found in Table 4.3, the honorific

forms of three of the four pairs are considered *polite*. The remaining form, *-seupnida*, is deemed deferential.

Gloss	Non-honorific	Honorific	Example
Inflection	*-eo*	*-eoyo*	*iss-**eo***, *iss-**eoyo***
Tag-question	*-ji*	*-jyo*	*iss-**ji***, *iss-**jyo***
Declarative	*-ta*	*-seupnida*	*iss-**ta***, *iss-**seupnida***
Evidential Exclamation	*-ne*	*-neyo*	*iss-**ne***, *iss-**neyo***

Table 4.4: List of pairs of suffixes which alternate according to pragmatic considerations. Specifically, these suffxes alternate depending on whether an utterance is casual, or is marked as polite/formal via an honorific. Stem *iss* means 'there is'.

As before, the experiments consist of running visualization techniques, designed to see whether the embedding space can be partitioned in the expected way. These are PCA projection onto the first two principal components for visualization in $\mathbb{R}^2$, PCA projection onto individual components for visualization in $\mathbb{R}$, and U-Matrix visualization from a trained self-organizing map.

4.6.3 Results

A PCA visualization of the pragmatic affixes is as in Figure 4.23, with distirubtions on the individual principal dimensions in Figures 4.24 and 4.25.

From these visualizations, it appears that there is little to differentiate the honorific affixes from the non-honorific affixes. Though again, projection onto a later principal component does show that the distinction is made in the embeddings. In this case, projection onto the sixth principal direction shows a significant difference in distribution between the two types of affix. Finally, the SOM visualization in Figure 4.27 shows that in the high-dimensional space the two groups are readily discernable, suggesting KoBERT is capable of making pragmatic distinctions.

Figure 4.23: PCA visualization of assorted pragmatic affixes.

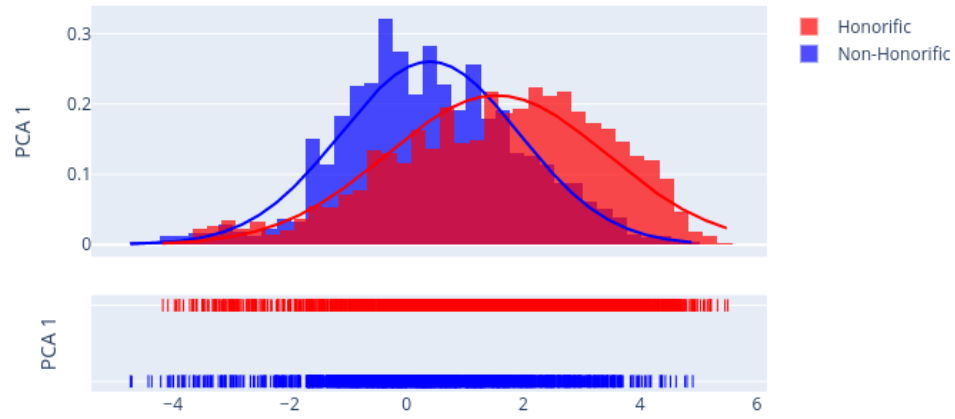


Figure 4.24: Visualization of pragmatic embeddings projected onto the first principal direction.

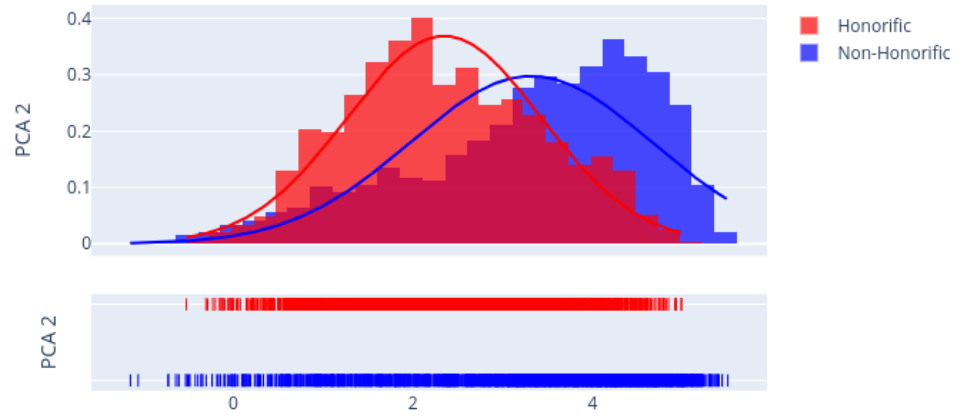


Figure 4.25: Visualization of pragmatic embeddings projected onto the second principal direction.

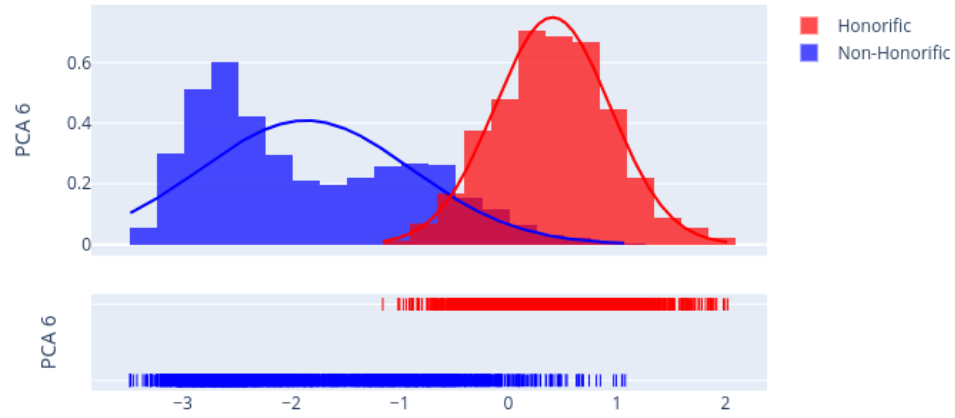


Figure 4.26: Visualization of pragmatic embeddings projected onto the sixth principal direction.

4.7 Conclusion

The results in this chapter have shown that contextual KoBERT embeddings of certain Korean affixes are distributed in a manner which is consistent with those affixes' linguistic

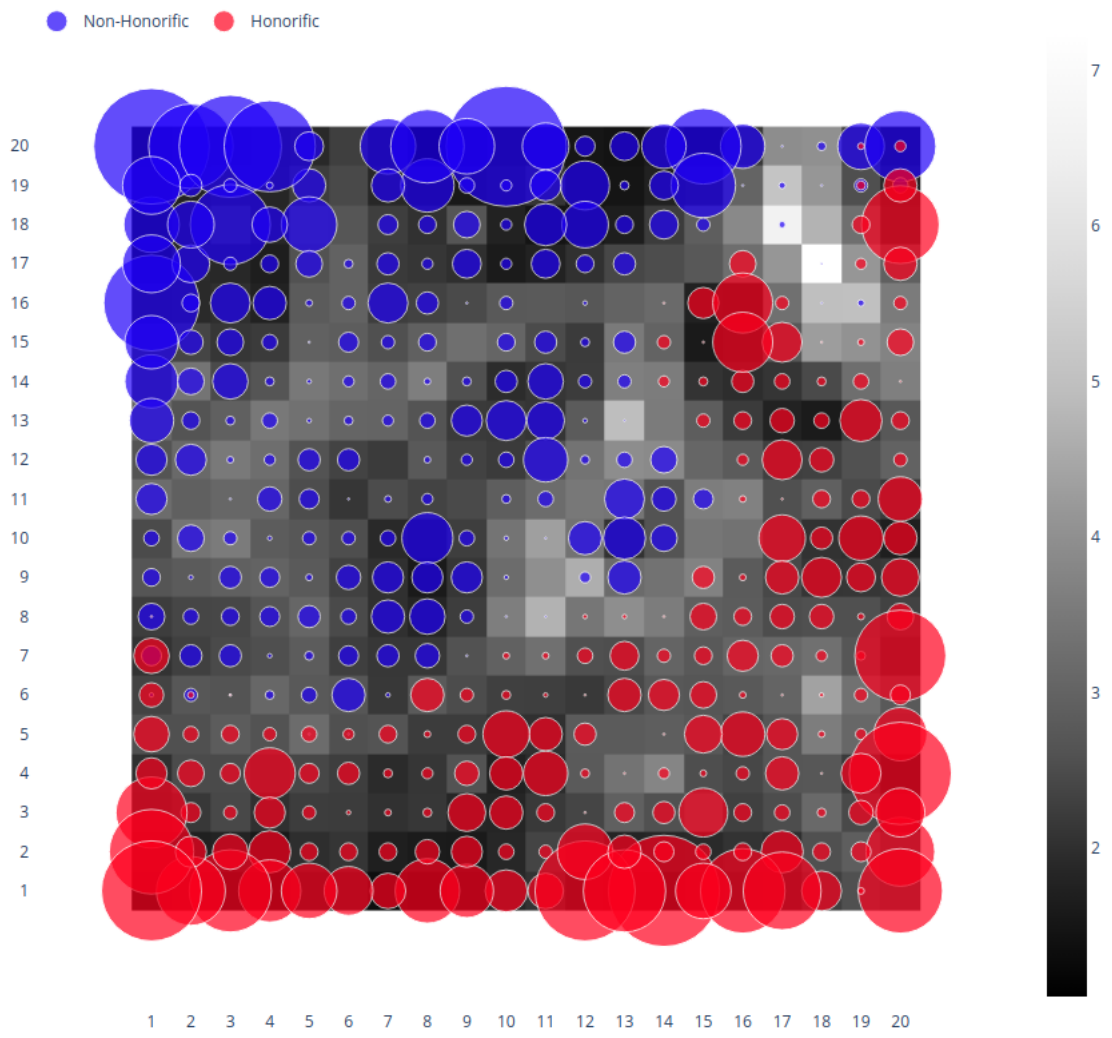


Figure 4.27: SOM visualization of assorted pragmatic affixes.

behavior, at least with regard to certain morphophonological and pragmatic phenomena. This was accomplished through data-visualization, using principal component analysis as well as self-organizing maps. These data-visualization techniques provided complementary views of KoBERT's affix embeddings. PCA visualizations allowed for a more fine-grained view of the data, and helped identify how significant the relevant distinctions were in the high-dimensional space, e.g. as the fourth largest driver of variation in the case of the C-Suffix/V-Suffix distinction. Meanwhile, SOM visualizations provided a holistic view of KoBERT's affix embeddings in the full 768 dimensions. The results in this chapter are

significant because excepting Edmiston and Kim (2019), there has been no literature (to my knowledge) which addresses these aspects of linguistic knowledge in continuous embeddings.

While it is difficult to say for certain, it is worth speculating as to the reasoning behind the positive results in this chapter. With regard to the morphophonological experiments, the distributions of C-suffixes and V-suffixes participating in the stem-final C-V alternation are complementary. As BERT-style models are purely distributional models, it is then reasonable to expect that KoBERT is sensitive to this fact. The bifurcation of the continuous space as seen in examples like Figure 4.12 would then be a reflection of KoBERT's sensitivity to these complementary distributions.

The distributions between familiar (non-honorific) and formal/polite (honorific) are not complementary, and thus the results of the pragmatics experiments are slightly more difficult to interpret. However, there are distributional cues which could be used to distinguish familiar from formal/polite speech. Such signals exist in specialized vocabulary, such as *japsusida*, a very formal alternative to *meokda* '*to eat*'. The use of such specialized vocabulary is correlated with other formal markers, such as honorific case markings. In addition to specialized vocabulary, the presence of other affixes also can be indicative of a register, particularly certain nominal affixes. Such affixes which can cue a listerner to whether the register is formal or casual include the vocative case marker, which generally connotes familiarity, and alternatively honorific case-markers which connote formality, such as honorific nominative marker *-kkyeseo*.

Using methods and a language uncommon in computational linguistics and natural language processing, this chapter has sought to show that BERT-style models are sensitive to distributional cues present in language in ways not yet explored. It is the hope that this chapter inspires other approaches which are out of the mainstream, as well as the exploration of models trained on understudied languages.

CHAPTER 5

CONCLUSION

Continuous representations of linguistic entities, such as words and affixes as examined in this dissertation, have proven to be indispensible for achieving state-of-the-art performance on a host of natural language processing tasks. Many of these tasks presumably require knowledge of linguistic structures which are generally described using the language of discrete mathematics. The assumption of this dissertation has been that the continuous representations produced by successful continuous models such as BERT actually encode much of the discrete linguistic structures theorized by linguists, and do so in an interpretable manner.

The aim of this thesis has then been to better understand the representations produced by BERT models from a linguistically-informed perspective. Contributions have been to explore aspects of language hitherto un(der)studied in the BERTology literature, and doing so largely by introducing novel methods from fields outside of NLP. The primary focus of the BERTology literature has been on syntactic information, however this thesis has demonstrated that BERT models robustly encode inflectional morphological features, and also are capable of encoding morphophonological distinctions, as well as pragmatic distinctions such as honorific/polite speech.

As mentioned in Chapter 1, investigating BERT models for linguistic information not only aids in better understanding BERT and how continuous models may or may not encode linguistic structures, but also aids in better understanding the types and amount of linguistic information which can be learnt from unlabeled text without linguistic priors, with BERT's representations serving as a proxy for what can be captured purely through distributional information.

Understanding very large and very complex models such as BERT is an enormous undertaking, and the question of the necessity of linguistic priors for learning natural language is larger still. This closing chapter recaps the modest contributions made to these fields of inquiry in this dissertation by reviewing each of Chapters 2, 3, and 4 in turn, while also

noting certain potential avenues for future research.

5.1 Chapter Reviews and Future Directions

5.1.1 Inflectional Morphology in BERT

Chapter 2 presented three sets of experiments investigating inflectional morphological features in BERT models' representations, as well as how the internal representations of BERT embeddings evolve through the layers. This was done using datasets constructued of BERT embeddings for words marked for specific inflectional features, investigating a total of five distinct Indo-European languages.

The first set of experiments investigated the intrinsic dimensionality of different point clouds of BERT embeddings, including the internal representations of BERT embeddings, i.e. the representations as produced at different layers. The experiments were conducted using two methods. The first method was PCA, and therefore assumed underlying linearity of the BERT manifold. The second involved a non-linear method of investigating intrinsic dimensionality. It was found that when estimating intrinsic dimensionality with PCA, estimates generally increased from layers 0 to 7, but decreased from layer 8 onward. This differed with what was found when estimating dimensionality via the non-linear method. When estimating non-linearly, intrinsic dimensionality was estimated to decrease sharply from layers 0 to 5, before roughly evening out from layers 6 onward. I hypothesized that this divergence between the linear and non-linear approximations suggests that as BERT embeddings move through the layers, the manifold is increasingly resembling a space-filling curve. These methods therefore indirectly address the shape of the high-dimensional manifold BERT embeddings are hypothesized to be drawn. Interestingly, it was found that there was little correlation between morphological complexity of words from which embeddings were gathered and intrinsic dimensionality.

The second set of experiments consisted of training classifiers to predict inflectional fea-

ture values given sets of embeddings produced by different layers of BERT. In experiments with both linear and non-linear classifiers, it was shown that morphological feature values could be predicted with high fidelity given BERT embeddings from different layers, showing that such morphological information is encoded in BERT's representations. Furthermore, the performance of the linear classifier as matching or even out-performing the non-lienar classifier suggests that BERT's embedding space can be partitioned into convex sub-regions according to inflectional feature value.

In addition to showing that BERT is encoding such inflectional information in its continuous embeddings, this set of experiments also examined the role that syncretism plays in classifiers' ability to distinguish between BERT embeddings from word occurrences with different feature values. It was shown that there is a negative correlation between the amount of values a word can take for a particular feature and classifiers' ability to correctly predict feature values. This is an unsurprising result, but important in that it exposes an instance in which BERT is failing to match human-level performance with regard to a linguistic task, as native speakers are able to effortlessly parse ambiguous word forms.

Finally, this set of experiments also compared BERT's abilities to classify different features, and also compared different layers in BERT for performance on classification. The results were that while BERT is able to classify inflectional feature values quite well across the board, there were differences between different features. For example, the highest classification scores were for the Tense feature, while the lowest were for Case. With regard to layer, scores generally peaked by the sixth layer in all cases for the linear classifier, while they generally peaked earlier for the non-linear classifier.

The third and final set of experiments consisted again of training classifiers on BERT embeddings, but this time in an unsupervised manner. The training of an unsupervised clustering method on BERT embeddings offered a complementary view to the linear and non-linear classifiers from the preceding experiment in that while the supervised experiments showed that inflectional information is encoded in BERT's representations, the unsupervised

experiments showed that this information is not readily available without supervised signal.

5.1.2 Disentangling BERT embeddings

Chapter 3 began by drawing analogies between generative neural networks, a type of neural network frequently used in the computer vision literature, and generative grammar. In doing so, it motivated the use of generative neural networks to model BERT embeddings using assumptions from generative grammar. Specifically, these assumptions are that different types of linguistic information are present in word occurrences in natural language, such as discrete morphological information, discrete syntactic information, and continuous lexical semantic information. These types of linguistic information were modeled in the generative neural network framework as discrete and continuous latent variables. Performing (variational) inference over these latent variables allows for simultaneous inference of morphological and syntactic tags, as well as other lexical semantic tasks (in the case of Chapter 3, word sense disambiguation and sentiment analysis).

Continuing the analogy of modeling the latent variables as linguistic features, neural networks then model the grammar. Neural networks are already widely used for tasks like parsing and grammar induction, offering far wider coverage than hand-written grammars, but the latent-variable framework of generative neural networks allows for a more natural means of importing linguistic structure into these models.

While the results on the experiments in this chapter were not state-of-the-art, they showed that disentangled representation learning serves as a potential avenue to more linguistically interpretable representations, and enables the distillation of various types of linguistic information from high-dimensional embeddings. Furthermore, it was shown that the generative neural network framework serves as a means of reconciling neural network-based approaches with generative grammar assumptions, and allows for the injecting of linguistic priors into network architectures. The potential future research avenues are then virtually as wide as generative grammar itself.

5.1.3 Examining Affix Embeddings for Morphophonology and Pragmatics

Chapter 4 examined affix embeddings from a BERT model trained on the Korean language. Korean is a language which is generally understudied in the realm of NLP, and particularly with regard to evaluation methods, with there being no known precedent for a BERTology study using a Korean BERT model. This is a missed opportunity, as many of characteristics in Korean are shared by a large number of languages, and these characteristics are quite different than those in the more heavily studied Indo-European languages.

Chapter 4 provided examples where investigating BERT trained on non-Indo European languages can help understand more about aspects of language BERT is able to reflect. The first set of experiments made use of the fact that Korean shows predicatble patterns in morphophonology which are transparently reflected in orthography, meaning BERT models should in theory be sensitive to patterns. Experiment confirmed this suspicion, and provided the first piece of evidence that BERT encoded linguistic distinctions at the level of morphophonology.

A second example included Korean's rich system of honorifics and polite speech, which is encoded in the grammar in the form of specialized affixes. By examining embeddings of these affixes and comparing them with their non-honorific counterparts, experiments were able to show that BERT is also able to distinguish between pragmatic aspects of language as well.

Examination of Korean has thus demonstrated BERT's ability to encode natural language phenomena at the two hitherto unexplored levels of morphophonology and pragmatics. Very few of the world's languages have been examined via BERTology, meaning any facts regarding BERT, or the types of linguistic information readily learnable by models such as BERT, are going to be constrained by whatever can be learnt from typologically narrow examination. The most obvious avenue of future research is then to explore non-English, non-Indo-European languages for more diverse linguistic phenomena.

5.2 In Closing

Continuous distributed representations of linguistic entities have played a large part in the effectiveness of modern NLP methods. However, the increased effectiveness provided by these representations has come at the cost of interpretability. This work is part of an emerging literature which tries to better understand these black-box representations and models, particularly from a linguistic point of view. The modest contributions of this dissertation have shown that certain types of linguistic structure are robustly encoded in the continuous spaces of BERT, and demonstrates how methods from outside NLP can be repurposed in order to uncover yet further structure.

APPENDIX A

TREEBANK DETAILS FOR CHAPTER 2

For English, the following treebanks were used: the EWT treebank (Silveira et al., 2014), the GUM treebank (Zeldes, 2017), the LinES treebank (Ahrenberg, 2007), the English portion of ParTUT (Bosco et al., 2012), English-PUD (Zeman et al., 2018), and the English-Pronouns treebank (Munro, 2020). For French, the following treebanks were used: French Question Bank (Judge et al., 2006), the GSD French treebank (Nivre et al., 2016), French portion of ParTUT, French-PUD, Sequoia (Candito and Seddah, 2012), and the French Spoken Treebank, adapted from the Rhapsoide prosodic-syntactic treebank (Lacheret et al., 2014). For German the following treebanks were used: The HDT-UD treebank (Völker et al., 2019), and the GSD German treebank. For Russian, the following treebanks were used: The GSD Russian treebank, Russian-PUD, The SynTagRus treebank (Nivre et al., 2008), and the Taiga treebank (Lyashevskaya et al., 2016). For Spanish, the following treebanks were used: The AnCora treebank (Taulé et al., 2008), the GSD Spanish treebank, and Spanish-PUD.

APPENDIX B

RESULTS FOR RANDOM BASELINES

Feat.\Lang.	English	French	German	Russian	Spanish	Average
Case			0.58	0.28		0.43
Gender		0.61	0.5	0.4	0.71	0.56
Mood	0.77	0.76	0.78	0.87	0.63	0.76
Number	0.67	0.66	0.63	0.55	0.68	0.64
Person	0.93	0.82	0.75	0.78	0.7	0.8
Tense	0.75	0.7	0.77	0.49	0.67	0.67
VerbForm	0.49	0.6	0.62	0.45	0.5	0.53
Average	0.72	0.69	0.66	0.54	0.65	

Table B.1: Weighted F1 scores for random baseline models for features using linear classifier.

Layer\Lang.	English	French	German	Russian	Spanish	Average
Input	0.78	0.73	0.69	0.59	0.7	0.7
1	0.76	0.72	0.7	0.59	0.7	0.7
2	0.75	0.72	0.7	0.57	0.69	0.69
3	0.75	0.72	0.68	0.56	0.67	0.68
4	0.74	0.72	0.68	0.54	0.67	0.67
5	0.72	0.7	0.67	0.55	0.67	0.66
6	0.71	0.68	0.66	0.54	0.66	0.65
7	0.71	0.67	0.65	0.52	0.64	0.64
8	0.72	0.67	0.64	0.52	0.62	0.63
9	0.7	0.66	0.65	0.52	0.62	0.63
10	0.69	0.65	0.62	0.52	0.61	0.62
11	0.68	0.67	0.62	0.51	0.6	0.61
12	0.67	0.65	0.63	0.51	0.59	0.61
Average	0.72	0.69	0.66	0.54	0.65	

Table B.2: Weighted F1 scores for random baseline models for layers using linear classifier.

Feat.\Lang.	English	French	German	Russian	Spanish	Average
Case			0.55	0.31		0.43
Gender		0.6	0.5	0.43	0.73	0.56
Mood	0.81	0.72	0.83	0.92	0.49	0.75
Number	0.64	0.64	0.52	0.57	0.66	0.61
Person	0.93	0.77	0.79	0.82	0.64	0.79
Tense	0.77	0.7	0.77	0.47	0.58	0.66
VerbForm	0.51	0.46	0.58	0.44	0.53	0.5
Average	0.73	0.65	0.65	0.56	0.61	

Table B.3: Weighted F1 scores random baseline for features using non-linear classifier.

Layer\Lang.	English	French	German	Russian	Spanish	Average
Input	0.76	0.74	0.63	0.62	0.69	0.69
1	0.75	0.66	0.69	0.61	0.66	0.67
2	0.75	0.72	0.68	0.56	0.67	0.67
3	0.73	0.7	0.63	0.6	0.66	0.66
4	0.75	0.7	0.62	0.59	0.63	0.66
5	0.73	0.67	0.66	0.56	0.61	0.65
6	0.72	0.64	0.65	0.58	0.62	0.64
7	0.73	0.65	0.68	0.53	0.59	0.64
8	0.75	0.62	0.63	0.55	0.62	0.63
9	0.72	0.67	0.6	0.54	0.64	0.64
10	0.64	0.57	0.65	0.54	0.56	0.59
11	0.7	0.62	0.63	0.55	0.61	0.62
12	0.72	0.62	0.58	0.54	0.53	0.6
Average	0.73	0.66	0.64	0.57	0.62	

Table B.4: Weighted F1 scores for random baseline models for layers using non-linear classifier.

APPENDIX C

DERIVATION OF EVIDENCE LOWER BOUND (ELBO)

This appendix explains and derives the Evidence Lower Bound (ELBO) term used as the objective to optimize variational autoencoders. In the context of variational autoencoders, the goal is to maximize the probability of observed data $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, assuming a single latent variable Z and where $p_\theta(\mathbf{x})$ can be defined as $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})dz$. Here, $p_\theta(\mathbf{x}|\mathbf{z})$ is paramterized by a neural network with parameters θ. The optimization problem is then to find θ^*, which can be defined as follows.

$$\theta^* = \underset{\theta}{argmax}\, \mathbb{E}_{\mathbf{x}\sim X}\big[log\, p_\theta(\mathbf{x})\big] \tag{C.1}$$

Maximizing this term directly is intractable due to the integration term in $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})dz$. Furtermore, sample-based methods of approximation, where $\mathbf{z} \sim p(\mathbf{z})$, are prohibatively slow, particularly in high-dimensions. Training can be significantly sped up if $\mathbf{z}$ is drawn from the posterior $p_\theta(\mathbf{z}|\mathbf{x})$, however calculation of $p_\theta(\mathbf{z}|\mathbf{x}) = \frac{p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p_\theta(\mathbf{x})}$ is also intractable, due to calculation of the evidence term $p_\theta(\mathbf{x})$ requiring integration. Thus, rather than maximize the evidence $p_\theta(\mathbf{x})$, VAEs seek to maximize a lower bound on the evidence, the ELBO, by drawing from variational distribution $q_\phi(\mathbf{z}|\mathbf{x})$, which is an approximation to the true posterior $p_\theta(\mathbf{z}|\mathbf{x})$. The derivation of the lower bound is in Table C.1.

		(In)Equality	Justification
$log\, p_\theta(\mathbf{x})$	$=$	$\mathbb{E}_{q_\phi(\mathbf{z}\|\mathbf{x})}\left[log\, p_\theta(\mathbf{x})\right]$	Independence of $p_\theta(\mathbf{x})$ from $q_\phi(\mathbf{z}\|\mathbf{x})$
	$=$	$\mathbb{E}_{q_\phi(\mathbf{z}\|\mathbf{x})}\left[log\frac{p_\theta(\mathbf{x},\mathbf{z})}{p_\theta(\mathbf{z}\|\mathbf{x})}\right]$	Multiplication of $p_\theta(\mathbf{x})$ by $\frac{p_\theta(\mathbf{z}\|\mathbf{x})}{p_\theta(\mathbf{z}\|\mathbf{x})} = 1$
	$=$	$\mathbb{E}_{q_\phi(\mathbf{z}\|\mathbf{x})}\left[log\frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}\|\mathbf{x})}\frac{q_\phi(\mathbf{z}\|\mathbf{x})}{p_\theta(\mathbf{z}\|\mathbf{x})}\right]$	Multiplication by $\frac{q_\phi(\mathbf{z}\|\mathbf{x})}{q_\phi(\mathbf{z}\|\mathbf{x})} = 1$ and commutativity
	$=$	$\mathbb{E}_{q_\phi(\mathbf{z}\|\mathbf{x})}\left[log\frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}\|\mathbf{x})}\right] + \mathbb{E}_{q_\phi(\mathbf{z}\|\mathbf{x})}\left[log\frac{q_\phi(\mathbf{z}\|\mathbf{x})}{p_\theta(\mathbf{z}\|\mathbf{x})}\right]$	Log product rule and linearity of expectation
	$=$	$\mathbb{E}_{q_\phi(\mathbf{z}\|\mathbf{x})}\left[log\frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}\|\mathbf{x})}\right] + D_{KL}(q_\phi(\mathbf{z}\|\mathbf{x})\|p_\theta(\mathbf{z}\|\mathbf{x}))$	Definition of KL-Divergence
	$\geq$	$\mathbb{E}_{q_\phi(\mathbf{z}\|\mathbf{x})}\left[log\frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}\|\mathbf{x})}\right]$	Non-negativity of KL-Divergence
	$=$	$\mathbb{E}_{q_\phi(\mathbf{z}\|\mathbf{x})}\left[log\, p_\theta(\mathbf{x}\|\mathbf{z}) + log\, p(\mathbf{z}) - log\, q_\phi(\mathbf{z}\|\mathbf{x})\right]$	Chain rule and log product rule
	$=$	$\mathbb{E}_{q_\phi(\mathbf{z}\|\mathbf{x})}\left[log\, p_\theta(\mathbf{x}\|\mathbf{z}) - D_{KL}(q_\phi(\mathbf{z}\|\mathbf{x})\|p(\mathbf{z}))\right]$	Definition of KL-Divergence

Table C.1: Derivation of ELBO.

REFERENCES

Ahalt, Stanley C, Ashok K Krishnamurthy, Prakoon Chen, and Douglas E Melton. 1990. Competitive learning algorithms for vector quantization. *Neural networks* 3:277–290.

Ahrenberg, Lars. 2007. Lines: An English-Swedish parallel treebank. In *Proceedings of the 16th Nordic Conference of Computational Linguistics (NODALIDA 2007)*, 270–273.

Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* .

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Baroni, Marco, Georgiana Dinu, and Germán Kruszewski. 2014. Dont count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 238–247.

Belinkov, Yonatan, and James Glass. 2019. Analysis methods in neural language processing: A survey. *Transactions of the Association for Computational Linguistics* 7:49–72.

Belkin, Mikhail, and Partha Niyogi. 2003. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation* 15:1373–1396.

Beltagy, Iz, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150* .

Bengio, Yoshua, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research* 3:1137–1155.

Bojanowski, Piotr, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* 5:135–146.

Bosco, Cristina, Manuela Sanguinetti, and Leonardo Lesmo. 2012. The Parallel-TUT: a multilingual and multiformat treebank. In *Eight International Conference on Language Resources and Evaluation (LREC'12)*, 1932–1938. European Language Resources Association (ELRA).

Bowman, Samuel R, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349* .

Brown, Lucien. 2015. Honorifics and Politeness. *The handbook of Korean linguistics* 303–319.

Candito, Marie, and Djamé Seddah. 2012. Le corpus Sequoia: Annotation syntaxique et exploitation pour l'adaptation d'analyseur par pont lexical.

Cho, Young-mee Yu, and Peter Sells. 1995. A lexical account of inflectional suffixes in Korean. *Journal of East Asian Linguistics* 4:119–174.

Clark, Kevin, Urvashi Khandelwal, Omer Levy, and Christopher Manning. 2019. What Does BERT Look At? An Analysis of BERT's Attention. *arXiv preprint arXiv:1906.04341* .

Coenen, Andy, Emily Reif, Ann Yuan, Been Kim, Adam Pearce, Fernanda Viégas, and Martin Wattenberg. 2019. Visualizing and measuring the geometry of BERT. *arXiv preprint arXiv:1906.02715* .

Croom, Fred H. 2016. *Principles of Topology*. Courier Dover Publications.

De Ridder, Dick, and Robert PW Duin. 1997. Sammon's mapping using neural networks: a comparison. *Pattern Recognition Letters* 18:1307–1316.

Deerwester, Scott, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science* 41:391–407.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186.

Doersch, Carl. 2016. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908* .

Dupont, Emilien. 2018. Learning disentangled joint continuous and discrete representations. *arXiv preprint arXiv:1804.00104* .

Edmiston, Daniel. 2020. A Systematic Analysis of Morphological Content in BERT Models for Multiple Languages. *arXiv preprint arXiv:2004.03032* .

Edmiston, Daniel, and Taeuk Kim. 2019. Intrinsic evaluation of grammatical information within word embeddings .

Ettinger, Allyson. 2020. What BERT is not: Lessons from a new suite of psycholinguistic diagnostics for language models. *Transactions of the Association for Computational Linguistics* 8:34–48.

Finley, Sara. 2006. Vowel harmony in Korean and morpheme correspondence. *Harvard Studies in Korean Linguistics* 11:131–144.

Garcia, Marcos. 2016. Universal dependencies guidelines for the Galician-TreeGal treebank. Technical report, Technical Report, LyS Group, Universidade da Coruna.

Goldberg, Yoav. 2019. Assessing BERT's syntactic abilities. *arXiv preprint arXiv:1901.05287* .

Hannun, Awni, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. 2014. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567* .

Hewitt, John, and Christopher D Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4129–4138.

Higgins, Irina, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2016. beta-vae: Learning basic visual concepts with a constrained variational framework .

Hilbert, David. 1935. Über die stetige abbildung einer linie auf ein flächenstück. In *Dritter band: Analysis· grundlagen der mathematik· physik verschiedenes*, 1–2. Springer.

Hofmann, Valentin, Janet Pierrehumbert, and Hinrich Schütze. 2020. DagoBERT: Generating derivational morphology with a pretrained language model .

Hornik, Kurt. 1991. Approximation capabilities of multilayer feedforward networks. *Neural networks* 4:251–257.

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White. 1989. Multilayer feedforward networks are universal approximators. *Neural networks* 2:359–366.

Hotelling, Harold. 1933. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology* 24:417.

Htut, Phu Mon, Jason Phang, Shikha Bordia, and Samuel R Bowman. 2019. Do attention heads in BERT track syntactic dependencies? *arXiv preprint arXiv:1911.12246* .

Jain, Sarthak, Edward Banner, Jan-Willem van de Meent, Iain J Marshall, and Byron C Wallace. 2018. Learning disentangled representations of texts with application to biomedical abstracts. *arXiv preprint arXiv:1804.07212* .

Jawahar, Ganesh, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *ACL 2019-57th Annual Meeting of the Association for Computational Linguistics*.

Jones, Karen Sparck. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation* .

Judge, John, Aoife Cahill, and Josef Van Genabith. 2006. Questionbank: Creating a corpus of parse-annotated questions. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, 497–504. Association for Computational Linguistics.

Keung, Phillip, Yichao Lu, György Szarvas, and Noah A Smith. 2020. The multilingual Amazon reviews corpus. *arXiv preprint arXiv:2010.02573* .

Kim, Gyung-Ran. 2006. Suffix-centered allomorphy in Korean. *Studies in Phonetics, Phonology, and Morphology* 12:265–281.

Kim, Taeuk, Jihun Choi, Daniel Edmiston, and Sang goo Lee. 2020. Are Pre-trained Language Models Aware of Phrases? Simple but Strong Baselines for Grammar Induction. In *International Conference on Learning Representations*.

Kim, Yoon, Yacine Jernite, David Sontag, and Alexander Rush. 2016. Character-aware neural language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30.

Kingma, Diederik P, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. 2014. Semi-supervised learning with deep generative models. In *Advances in neural information processing systems*, 3581–3589.

Kingma, Diederik P, and Max Welling. 2013. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114* .

Kohonen, Teuvo. 1982. Self-organized formation of topologically correct feature maps. *Biological cybernetics* 43:59–69.

Kohonen, Teuvo. 2012. *Self-organizing maps*, volume 30. Springer Science & Business Media.

Kondratyuk, Dan, and Milan Straka. 2019. 75 languages, 1 model: Parsing universal dependencies universally. *arXiv preprint arXiv:1904.02099* .

Koopman, Hilda. 2005. Korean (and Japanese) morphology from a syntactic perspective. *Linguistic Inquiry* 36:601–633.

Kudo, Taku. 2006. Mecab: Yet another part-of-speech and morphological analyzer. *http://mecab. sourceforge. jp* .

Lacheret, Anne, Sylvain Kahane, Julie Beliao, Anne Dister, Kim Gerdes, Jean-Philippe Goldman, Nicolas Obin, Paola Pietrandrea, and Atanas Tchobanov. 2014. Rhapsodie: a prosodic-syntactic treebank for spoken French.

Landauer, Thomas K, Peter W Foltz, and Darrell Laham. 1998. An introduction to latent semantic analysis. *Discourse processes* 25:259–284.

Lau, Jey Han, and Timothy Baldwin. 2016. An empirical evaluation of Doc2Vec with practical insights into document embedding generation. *arXiv preprint arXiv:1607.05368* .

Le, Quoc, and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*, 1188–1196. PMLR.

Lee, John A, and Michel Verleysen. 2007. *Nonlinear dimensionality reduction*. Springer Science & Business Media.

Lee, Yongsung. 1993. Topics in the vowel phonology of Korean. Doctoral Dissertation, Indiana University.

Liu, Jianbo, and Dragan Djurdjanovic. 2008. Topology preservation and cooperative learning in identification of multiple model systems. *IEEE transactions on neural networks* 19:2065–2072.

Lyashevskaya, Olga, Kira Droganova, Daniel Zeman, Maria Alexeeva, Tatiana Gavrilova, Nina Mustafina, Elena Shakurova, et al. 2016. Universal dependencies for Russian: A new syntactic dependencies tagset. *Series: Linguistics, WP BRP 44/LNG* .

McCann, Bryan, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: contextualized word vectors. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 6297–6308.

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

Mikolov, Tomáš, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*.

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 3111–3119.

Munro, Robert. 2020. *Human-in-the-loop machine learning*. Manning.

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

Nivre, Joakim, Igor Boguslavsky, and Leonid Iomdin. 2008. Parsing the SynTagRus treebank of Russian. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, 641–648.

Nivre, Joakim, Marie-Catherine De Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, 1659–1666.

Nivre, Joakim, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal dependencies v2: An evergrowing multilingual treebank collection. *arXiv preprint arXiv:2004.10643* .

Park, Eunjeong L, and Sungzoon Cho. 2014. KoNLPy: Korean natural language processing in Python. In *Annual Conference on Human and Language Technology*, 133–136. Human and Language Technology.

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

Pennington, Jeffrey, Richard Socher, and Christopher D Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532–1543.

Peters, Matthew E, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365* .

Pilehvar, Mohammad Taher, and Jose Camacho-Collados. 2018. WiC: the Word-in-Context dataset for evaluating context-sensitive meaning representations. *arXiv preprint arXiv:1808.09121* .

Potapov, A, and MK Ali. 2002. Neural networks for estimating intrinsic dimension. *Physical Review E* 65:046212.

Qiu, Jiezhong, Hao Ma, Omer Levy, Scott Wen-tau Yih, Sinong Wang, and Jie Tang. 2019. Blockwise self-attention for long document understanding. *arXiv preprint arXiv:1911.02972* .

Qiu, Siyu, Qing Cui, Jiang Bian, Bin Gao, and Tie-Yan Liu. 2014. Co-learning of word representations and morpheme representations. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, 141–150.

Radford, Alec, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training .

Rogers, Anna, Olga Kovaleva, and Anna Rumshisky. 2021. A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics* 8:842–866.

Sagot, Benoît. 2018. A multilingual collection of CoNLL-U-compatible morphological lexicons. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Siddharth, Narayanaswamy, Brooks Paige, Jan-Willem Van de Meent, Alban Desmaison, Noah Goodman, Pushmeet Kohli, Frank Wood, and Philip Torr. 2017. Learning disentangled representations with semi-supervised deep generative models. In *Advances in neural information processing systems*, 5925–5935.

Silveira, Natalia, Timothy Dozat, Marie-Catherine de Marneffe, Samuel Bowman, Miriam Connor, John Bauer, and Christopher D. Manning. 2014. A gold standard dependency corpus for English. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*.

Socher, Richard, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, 1631–1642.

Sohn, Ho-Min. 2001. *The Korean Language*. Cambridge University Press.

Srivastava, Nitish. 2013. Improving neural networks with dropout. *University of Toronto* 182:7.

Strauss, Susan, and Jong Oh Eun. 2005. Indexicality and honorific speech level choice in Korean. *Linguistics* 43:611–651.

Taulé, Mariona, Maria Antònia Martí, and Marta Recasens. 2008. AnCora: Multilevel Annotated Corpora for Catalan and Spanish. In *Lrec*.

Tenney, Ian, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. *arXiv preprint arXiv:1905.05950* .

Tyers, Francis M., Mariya Sheyanova, Alexandra Martynova, Pavel Stepachev, and Konstantin Vinogradovsky. 2018. Multi-source synthetic treebank creation for improved cross-lingual dependency parsing. In *Proceedings of the Second Workshop on Universal Dependencies (UDW 2018)*, 144–150.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.

Vig, Jesse, and Yonatan Belinkov. 2019. Analyzing the structure of attention in a transformer language model. *arXiv preprint arXiv:1906.04284* .

Vincent, Pascal, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research* 11:3371–3408.

Vinh, Nguyen Xuan, Julien Epps, and James Bailey. 2010. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *The Journal of Machine Learning Research* 11:2837–2854.

Völker, Emanuel Borges, Maximilian Wendt, Felix Hennig, and Arne Köhn. 2019. HDT-UD: A very large Universal Dependencies treebank for German. In *Proceedings of the Third Workshop on Universal Dependencies (UDW, SyntaxFest 2019)*, 46–57.

Von Luxburg, Ulrike. 2007. A tutorial on spectral clustering. *Statistics and computing* 17:395–416.

Vulić, Ivan, Edoardo Maria Ponti, Robert Litschko, Goran Glavaš, and Anna Korhonen. 2020. Probing pretrained language models for lexical semantics. *arXiv preprint arXiv:2010.05731* .

Wainwright, Martin J, and Michael Irwin Jordan. 2008. *Graphical models, exponential families, and variational inference*. Now Publishers Inc.

Wang, Jianzhong. 2012. *Geometric structure of high-dimensional data and dimensionality reduction*, volume 5. Springer.

Warstadt, Alex, Amanpreet Singh, and Samuel R Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics* 7:625–641.

Wieting, John, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Towards universal paraphrastic sentence embeddings. *arXiv preprint arXiv:1511.08198* .

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

Wu, Yonghui, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* .

Xie, Junyuan, Ross Girshick, and Ali Farhadi. 2016. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, 478–487.

Yeon, Jaehoon, and Lucien Brown. 2019. *Korean: A Comprehensive Grammar*. Routledge.

Zeldes, Amir. 2017. The GUM corpus: Creating multilayer resources in the classroom. *Language Resources and Evaluation* 51:581–612.

Zeman, Daniel, Jan Hajic, Martin Popel, Martin Potthast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. 2018. CoNLL 2018 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual parsing from raw text to universal dependencies*, 1–21.