Contextualized Embeddings Encode Knowledge of English Verb–Noun Combination Idiomaticity

by

Samin Fakharian

Bachelor of Computer Software Engineering, K. N. Toosi University of Technology, 2018

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Computer Science

In the Graduate Academic Unit of Faculty of Computer Science

Supervisor(s):	Paul Cook,PhD, Computer Science
Examining Board:	Arash Habibi Lashkari, Post-Doc, Computer Science, Chair
	Scott Bateman, PhD, Computer Science
External Examiner:	Christine Horne, PhD, French

This thesis is accepted by the
Dean of Graduate Studies

THE UNIVERSITY OF NEW BRUNSWICK

April, 2021

Abstract

English verb-noun combinations (VNCs) consist of a verb with a noun in its direct object position, and can be used as idioms or as literal combinations (e.g., *hit the road*). As VNCs are commonly used in language and their meaning is often not predictable, they are an essential topic of research for NLP. In this study, we propose a supervised approach to distinguish idiomatic and literal usages of VNCs in a text based on contextualized representations, specifically BERT and RoBERTa. We show that this model using contextualized embeddings outperforms previous approaches, including the case that the model is tested on instances of VNC types that were not observed during training. We further consider the incorporation of linguistic knowledge of lexico-syntactic fixedness of VNCs into our model. Our findings indicate that contextualized embeddings capture this information.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Natural language processing (NLP) is a branch of artificial intelligence that deals with textual data and allows machines to read, understand and analyze natural language data. The root of natural language processing is in the 1950s. An article titled "Computing Machinery and Intelligence" [68] proposed the Turing test as a measure of intelligence, a task requiring automated analysis and generation of natural language, but was not expressed as a problem separate from artificial intelligence at the time. In this test, a human evaluator determines whether a machine can produce human-like responses through communications between a device and a human partner. Suppose the evaluator, who would be aware that one of the two interaction partners is a computer, is not able to distinguish which responses are from the artificial partner. In that case, the computer is said to pass the Turing test. The communications are limited to a text-only channel such as a computer keyboard and screen.

The nature of human language makes NLP difficult due to the ambiguity of human language, and it is not easy for computers to understand natural languages. To fully comprehend human language, one must both understand the words and the connections between them in order to convey the intended meaning. In the past

years, computers can better learn models to understand and extract meaning from written and spoken language with the help of neural networks for machine learning, and these approaches are used in many NLP tasks.

There are a wide range of NLP problems. For example, in the part-of-speech (POS) tagging task, a system tries to determine the POS for each token in a sentence. A category of words with similar grammatical properties is a part-of-speech. Noun, verb, adjective, adverb, pronoun, preposition are common parts-of-speech. In the sentence *I like to watch TV* the expected output could be the sequence *pronoun*, *verb*, *preposition*, *verb*, *noun*. In named entity recognition (NER) the system tries to find the proper names such as people or places and the type of each such name (e.g. person, location, organization) given an input sentence. Machine translation (MT) [29] is another widely studied and difficult task in which a system is developed in order to translate a given sentence or document into a different language.

Multiword expressions (MWEs) are lexicalized combinations of multiple words, which display some form of idiomaticity [6]. MWEs include a wide variety of phenomena such as fixed expressions (e.g., *by and large*), light verb constructions (e.g., *take a walk*), and verb-noun combinations (e.g., *see stars*) and they are frequently used in many languages in both text and speech. MWEs have varying degrees of compositionality, i.e., the degree to which each component word contributes to the semantics of the expression. Multiword expressions have received a lot of attention from researchers [58]. However, there are relatively few resources, such as manually annotated corpora in various languages for multiword expressions, which makes working with them a "pain in the neck" for NLP [58]. To build a robust natural language processing technology, handling MWEs is very important [58] because knowledge of MWEs is essential in many downstream applications such as machine translation and information retrieval.

In machine translation, the multiword expression problem is that the system should

decide whether to translate an MWE word by word or consider it as a single unit. The story is the same for the information retrieval task. When a user is querying *hot dog* in a web search engine like Google, it is unusual to think that the user is actually looking for a *dog* that is *hot*. When the system sees an MWE like *hot dog* as a single unit, the system will give the user better results [1]. Addressing the challenges related to multiword expressions is therefore essential.

We now focus on verb–noun combinations (VNCs), which consist of a verb with a noun in its direct object position. VNCs are a common kind of MWE in English and also cross-lingually [25]. VNCs exhibit semantic idiosyncrasy, i.e., the meanings of VNCs are, to varying degrees, not transparent from the meanings of their component words. In the following example, *hit the road* has an idiomatic meaning corresponding roughly to 'start a journey':

1. The marchers had hit the road before 0500 hours and by midday they were limping back having achieved success on day one.

Furthermore, VNCs are often ambiguous with literal combinations whose meanings are transparent. In the following sentence, for example, *hit the road* is a literal combination, i.e., not an MWE, and the idiomatic meaning of 'start a journey' that this expression can have is not present:

2. Two climbers dislodged another huge block which hit the road within 18 inches of one of the estate's senior guides.[1]

The idiomatic interpretations of VNCs are typically lexico-syntactically fixed. For example, the idiomatic interpretation of *hit the road* generally is not accessible if the determiner is indefinite (e.g., *hit a road*), the noun is plural (e.g., *hit the roads*), or the voice is passive (e.g., *the road was hit*); in such cases typically only the literal interpretation is available.

[1]These example sentences are taken, with slight editing, from the VNC-Tokens dataset [20].

Several studies have considered automatic methods for distinguishing literal and idiomatic usages of VNCs. [27] apply k-means clustering to representations of VNC usages based on word embeddings [48], and then label each cluster as idiomatic or literal based on whether the majority of its instances are in a canonical or non-canonical form, respectively. [61] propose a supervised approach to predicting the token-level idiomaticity of VNCs based on training an SVM on skip-thoughts [35] representations of sentences containing VNCs. [33] achieve better results using a more straightforward sentence representation based on the average of word embeddings. Moreover, [33] show that adding a single binary feature to the sentence representation indicating whether the VNC occurs in a canonical form gives substantial improvements. [37] propose a supervised approach to classifying instances of potentially-idiomatic expressions, including VNCs, as idiomatic or literal, using representations of these instances based on contextualized embeddings.

In the current study, we propose a supervised approach using contextualized embeddings, specifically BERT [23] and RoBERTa [45], to distinguish idiomatic and literal usages of VNCs. We evaluate our work using accuracy in order to be able to compare our results to previous studies and baselines.

In this study, the main research question is *Does an approach to identifying VNC idioms that incorporates contextualized embeddings outperform prior approaches that do not use contextualized embeddings?* The results of our experiments in Chapter 5 demonstrate that the proposed supervised approach using contextualized embeddings is able to outperform the previous best approach of [33]. We utilize pre-trained BERT [23] and RoBERTa [45] language representation models and fine-tune them on our dataset for this supervised classification task. We also experiment with the effect of incorporating a feature indicating whether a given VNC instance is in a canonical form. Here we find that our approach using contextualized embeddings outperforms prior approaches that do not use contextualized embeddings. The second research

question that we answer in this study is *Is an approach to identifying VNC idioms that incorporates contextualized embeddings able to generalize to unseen expressions?* The results show that our proposed method can generalize to expressions that are unseen during the training stage. In order to examine this, we leave each VNC idiom and its corresponding sentences out of the training dataset, let the system label the instances of the VNC that is unseen during the training and repeat this for each VNC type. We also incorporate the canonical form feature in these experiments to see whether adding this knowledge improves the system. This is an interesting experimental setup as we may not have enough annotated training data for every VNC type. Our results indicate that the model has learned information about the idiomaticity of VNCs in general, and not just for specific expressions that are present in the training data. In other words our model is able to generalize to VNCs that are unseen during training.

The contribution of this thesis can be clearly listed as follows:

1. Proposing an approach to identifying VNC idioms as idiomatic or literal that incorporates pre-trained contextualized embeddings, and outperforms the previous state-of-the-art for this task.

2. Finding that contextualized embeddings are able to capture the linguistic knowledge encoded in the canonical form feature.

3. Proposing an approach to identifying VNC idioms that is able to generalize to unseen expressions.

The remainder of this thesis is structured as follows. Chapter 2 will discuss related works, beginning with different types of embeddings, from count-based embeddings to contextualized embeddings. We also review the background and related work in multiword expressions and specifically in VNCs. In Chapter 3, we present our proposed supervised classification model, which utilizes contextualized embeddings.

Chapter 4 describes the dataset we used in this study along with the vector representation models we use to train and evaluate our models. Chapter 5 presents the results of our experiments. Finally, in Chapter 6, we summarise the contributions of this thesis, and we briefly conclude our work and discuss potential future research directions.

Chapter 2

Related Work

This chapter summarizes previous research on word vector representations, including traditional count-based embeddings, static word embeddings, and contextualized word embeddings. After that, we review previous research on multiword expressions.

2.1 Embeddings

In order to find similarities, relatedness, and other relationships between words, we should be able to represent words in a way to apply mathematical operations to them. A common way to represent words is therefore as vectors, referred to as embeddings.

2.1.1 Count-based word embeddings

A simple approach to build vector representations of words is a *count-based* model based on a co-occurrence matrix. For a vocabulary V of size $|V|$, a co-occurrence matrix X of dimension $|V| \times |V|$ is created by looking over a corpus and populating the entries of the matrix in such a way that X_{jk} contains the number of occurrences of each word j in a specified window-size around the word k as seen in the given corpus. There is a downside to the use of co-occurrence counts as measures of association. Some word pairs may co-occur with high frequency solely due to the fact that one

of the words in the pair is frequent itself. If two words co-occur with words that are similar, using co-occurrence matrices does not necessarily give us any information about the similarity of these words. Also, such a matrix is highly sparse as most indices are zero. Figure 2.1 shows an example of the co-occurrence matrix with a window of size 3 for the corpus including the sentences "I enjoy flying.", "I like NLP.", and "I like deep learning".

$$X = \begin{array}{c} \\ I \\ like \\ enjoy \\ deep \\ learning \\ NLP \\ flying \\ . \end{array} \begin{array}{c} \begin{array}{cccccccc} I & like & enjoy & deep & learning & NLP & flying & . \end{array} \\ \begin{bmatrix} 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

Figure 2.1: Co-occurrence matrix with a window of size 3 [14].

2.1.2 Standard word-embeddings

To address the sparsity and other limitations discussed, word embeddings based on artificial neural networks were proposed [36, 53] and they achieved state-of-the-art or competitive results in many NLP tasks such as sentence completion [47]. By this language modelling and feature learning technique, we can represent a word in a context with dense vector representations [57]. We can use the embeddings as pre-trained features for representing a word [48, 53], context [47], sentence [35], or document [40] in many NLP tasks such as document classification [35], information retrieval [72], and semantic role labelling [17]. In these approaches, a word is assigned to a low-dimensional fixed size dense vector, which is learned by the distribution of

a word in a text. These representations can successfully capture the semantic and syntactic roles of a word in the text. Using these embeddings, words that are used in similar contexts are located close to each other in vector space. As we can see

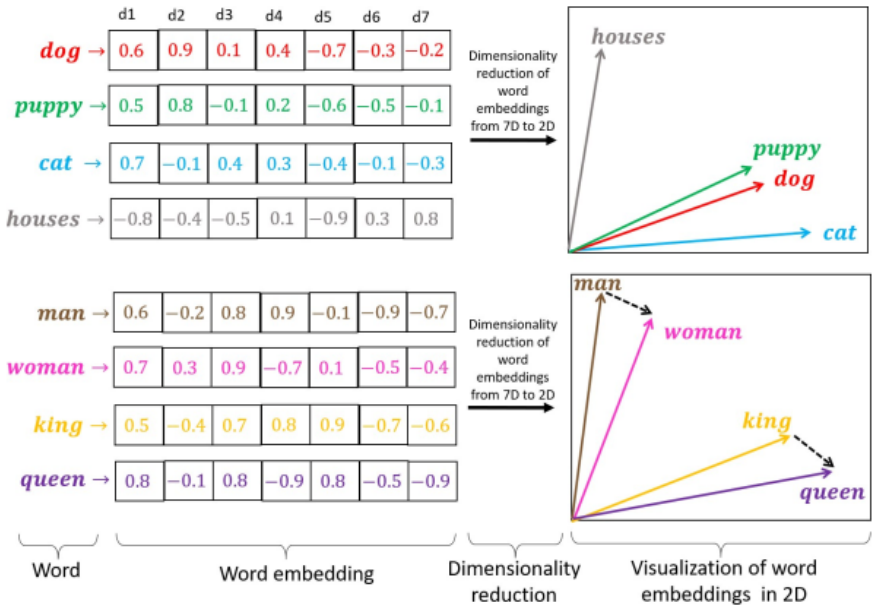

Figure 2.2: Example of 7-dimensional word embeddings, and their visualization in 2-dimensional space [57].

in Figure 2.2, the embeddings of the words *puppy* and *dog* are located close to each other, while the embedding of *houses* is far from them. Also, the word embeddings encode some relationships between words as well. For example:

$$v(king) - v(man) + v(woman) \approx v(queen)$$

The vector for *king* minus the vector for *man* plus the vector for *woman* will result in a vector that is close to the vector for *queen*. This shows that the model has learned to represent words with respect to a dimension that can capture gender and another dimension that can capture royalty.

At the word type level, the approach proposed by [48], referred to as word2vec, plays an important role in learning embeddings with neural networks that provide inter-

esting semantic relationships between words. Since this approach does not require matrix multiplication, the training of the word2vec model is very efficient [49]. Continuous bag-of-words (CBOW) and skip-gram are two different variations of neural network models we can use to learn word embeddings with word2vec. They use a shallow neural network to learn the word vectors. The purpose of the CBOW model is to predict the word representation of the target word $w(t)$ with respect to the words surrounding it in a fixed-size window (Fig 2.3). The CBOW model exploits a feedforward neural network language model in its architecture [8]. This prediction is, in fact, based on the sum of the vector representations of the words appearing in the window. The input to this model is the context words in the fixed-size symmetric context window $(w(t-i), w(t-i+1), ..., w(t-1), w(t+1), ..., w(t+i))$ encoded as a combination of one-hot vector representations of the context words of a target word, when i is the size of the window. The projection layer maps the input to lower dimensionality. After that, the distribution of all vocabulary words is learned and stored in a weight matrix. The purpose of the model is to maximize the probability of a target word in a given context with the help of the feedforward neural network. It tries to minimize the following loss function to learn the word representations, where w(t) is the target word:

$$E = -\log(p(w(t)|w(t-i), w(t-i+1), ..., w(t-1), w(t+1), ..., w(t+i))$$

In contrast, the Skip-gram model (Figure 2.4) takes a one-hot vector of the target word $w(t)$ as an input and tries to predict its context words $(w(t-i), w(t-i+1), ..., w(t-1), w(t+1), ..., w(t+i))$.

The document may include some information that the local context around the target word does not capture, and word2vec does not encode this in the word representations since word2vec only learns local context information. Global Vectors (GloVe), which was proposed by [53], considers all the global information in the training corpus in

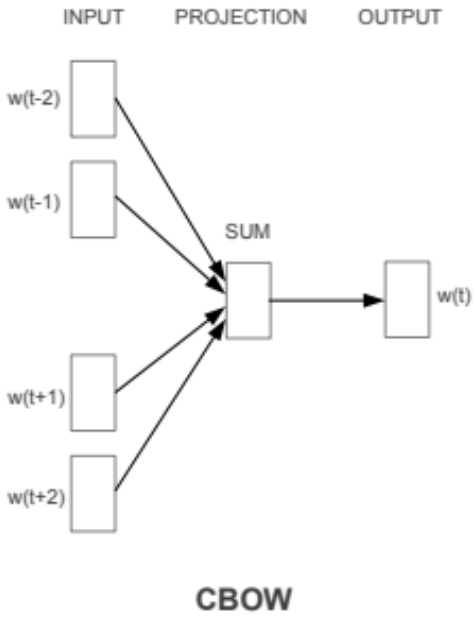

Figure 2.3: The CBOW model architecture proposed by [48].

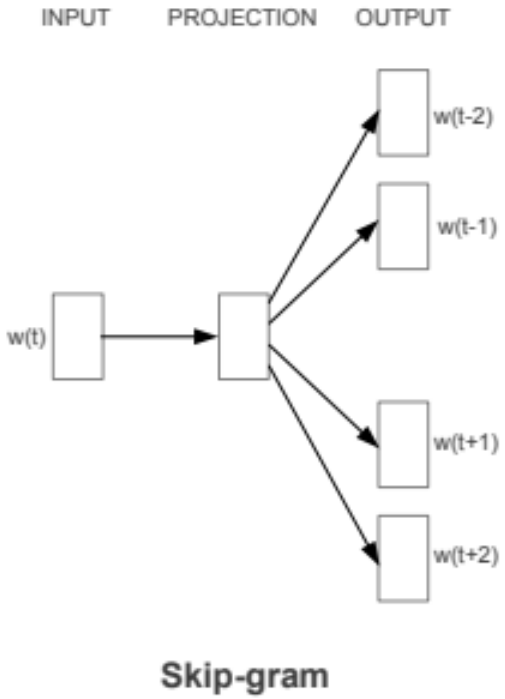

Figure 2.4: The Skip-gram model architecture proposed by [48].

the form of co-occurrences of words and their distance from the target word. For this purpose, we use a $V * V$ co-occurrence matrix X, in which V is the size of the vocabulary, and the element X_{ij} shows the number of times words in index i accompanied word in index j. When two words get used together often, word2vec does not know whether this is because one of the words is a common word or there is a strong linkage between them. In order to find whether two words such as i and j are similar to each other or not, using a co-occurrence matrix, we have to divide the number of times they have been seen with each other, X_{ij}, by the number of times the word i appeared in the corpus.

In recent years some complex approaches with the aim of improving word embeddings have been proposed including the work done by [41], which utilized dependency parse-trees and the work done by [73] which leveraged subword units.

To overcome this limitation, fastText [10] uses a bag of character n-grams (subwords) and extends the word2vec skip-gram model. In fastText, to obtain word embeddings of a word, we compute the average of the vectors associated with each of the character n-grams. For example to compute, the vector representation of the word *model* with the smallest n-gram = 3 and the largest n-gram = 6 we have to average the vectors for the n-grams "$<mo$", "mod", "$mode$", "$model$", "$model>$", "ode", "$odel$", "$odel>$" , "del", "$del>$", "$el>$" where "$<$" and "$>$" are special beginning and end of word markers. Representing each word as a bag of n-grams allowed the fastText model to better handle words that did not appear in the training corpus.

A further limitation of both word2vec and GloVe embedding models is that they assign one embedding for each word regardless of its sense. For example, we will have the same embedding for the word *watch* in the sentences *I lost my watch* and *Let's watch TV*. One limitation of both word2vec and GloVe is that they poorly estimate the word embeddings of rarely used words.

2.1.3 Contextualized word embeddings

The embedding methods we discussed until now assign each word a single global representation. Word2vec embeddings are learned based on the contexts in which words occur in. Context can have a significant impact on the word's meaning. In contrast contextualized embedding such as ELMO [47] and BERT [23] assign each token a context-dependent word representation. In these methods, each token is associated with a representation that is a function of an entire input sequence. These representations capture the semantic and syntactic properties of words in a specific context.

Mostly, contextualized word embeddings are pre-trained like a general-purpose language model on a large-scale unannotated corpus. These pre-trained embeddings can be used later as a representation layer in downstream NLP tasks. We can fine-tune these pre-trained embeddings to the task or use them as fixed embeddings.

ELMO (Embeddings from Language MOdels), proposed by [54] is one of the first contextualized embedding approaches that improved the results of many downstream NLP tasks. A two-layer bi-directional long short-term memory (LSTM) is associated with a language model (LM) objective and is trained on a large number of sentences of a huge corpus of text both forward, and backward [54]. The two language models that ELMO uses read each sentence in opposite directions. The system learns a linear combination of the layers stacked on top of each word, and we can derive the deep representations for each word, built from all internal layers of the bidirectional language model. ELMO can capture various types of syntactic and semantic information of a word in the context that it appears. ELMO embeddings can be concatenated to global word representations such as word2vec, and they can be used as an input to downstream NLP tasks without changing the architecture of their models. ELMO improves the state of the results of several NLP benchmarks including sentiment analysis [65], question answering [56] and named entity recognition

[67].

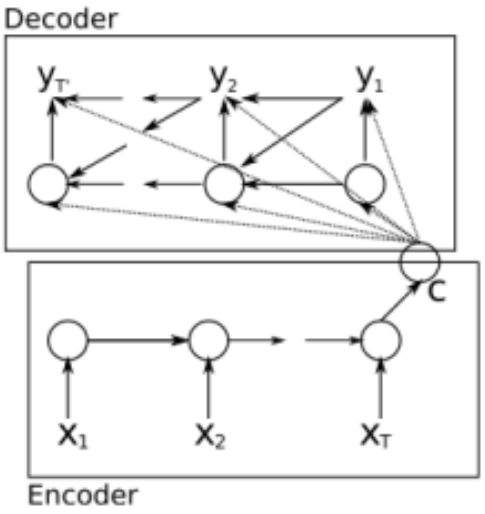

Figure 2.5: The model architecture of a simple encoder-decoder [16].

To focus on more recent contextualized word embeddings methods, we need to briefly explain the underlying principles of a new neural network architecture, transformers [70], and the concept of attention [3, 46]. In attention models, we mostly deal with encoders and decoders. The job of the encoder is to take the input sentence and represent the information of its context in a vector of continuous hidden values. After getting the entire input sequence, the encoder carries the context over to the decoder, and the decoder attempts to produce the output sequence one item at a time. You can see a simple visualization of the encoder-decoder model in Figure 2.5 in which x_1, x_2, x_T are the inputs, c is the summary of the input activations, and y_1, y_2, y_T are the outputs. The performance of the encoder-decoder architecture degrades in longer sentences [15]. Therefore, to better manage the context in long sentences a solution called "attention" has been proposed [3, 46] that helps the decoder to concentrate on the relevant parts of the input sequence by gathering information of each hidden unit separately. In the earlier encoder-decoder models referred to as sequence to sequence models, only the last hidden state of the encoding step (single

vector c in Figure 2.5) was sent to the decoder. However, in attention models, all the hidden states of the inputs (h) pass to the decoder. All the hidden states then will be scored based on their association with a certain word in the input sentence and will be multiplied by their softmaxed scores. The sum of the result of each hidden state will form a context vector for the decoder at each time step (Fig 2.6). At each time step in the decoder, the result of the concatenation of the context vector and hidden state of the decoder will be passed through the feed-forward neural network. The network indicates the output word of each time step.

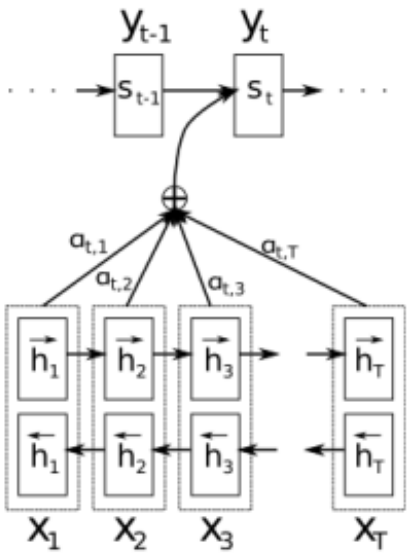

Figure 2.6: Illustration of the attention model proposed by [3] with y_t being the target word given a source sentence $x_1, x_2, ..., x_T$, a sequence of annotations $h_1, h_2, ..., h_T$, a_i the alignment model, and s_i RNN hidden state at time i. The alignment model determines the parts of the input sequence that are important to each output expression.

Transformers [70], with the help of an encoder-decoder model, try to model global dependencies within input and output tokens in a sequence with the help of an attention mechanism. They have noteworthy advantages over the conventional sequential models (RNN, LSTM, GRU, etc.), such as reducing the sequential computations which enables them to be trained much more efficiently. Transformers are made of

two components, which are the encoder stacks and decoder stacks. The encoder stacks take the inputs, and the decoder stacks consist of the same number of decoder blocks and produce the output tokens. Within each encoder and decoder component, there are also attention blocks that are referred to as multi-head self-attention layers, in combination with a regular feed-forward neural network. For each input sequence, these self-attention blocks are responsible for finding the relevant units (e.g. words) for each individual unit of the same sequence by looking at the entire sequence at once. Some language models only center on the decoder stacks of the transformer, such as Generative Pre-trained Transformer 2 (GPT-2) [55] and GPT-3 [11]. Also some models exist, such as BERT [23], which only deal with the encoder stacks of the transformers (Fig 2.7).

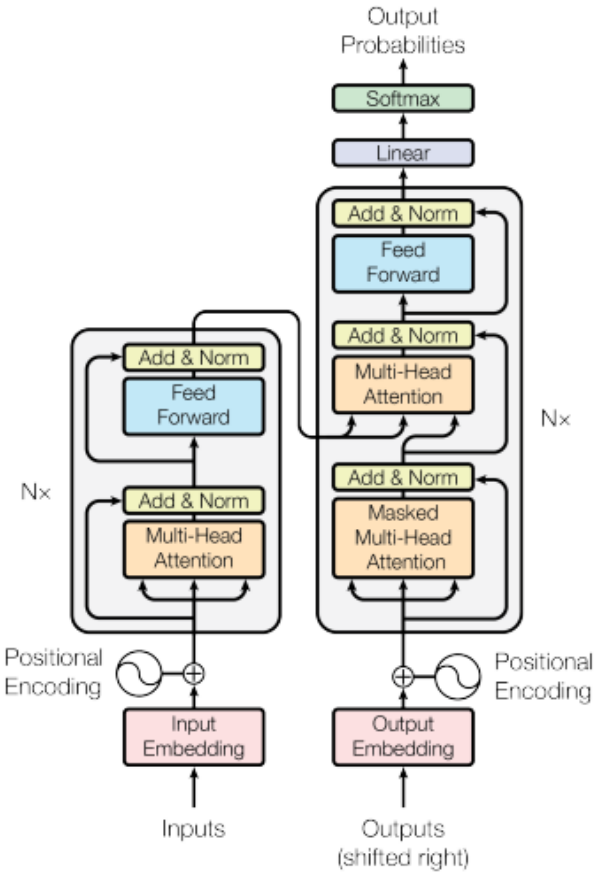

Figure 2.7: The model architecture of the transformers [70].

BERT (Bidirectional Encoder Representations from Transformers) introduced by [23] is the first contextualized embedding approach that, as mentioned earlier, only uses the encoder component of the transformers. BERT uses a technique called masked language modelling (MLM), which allows the model to see the entire sentence or text when predicting a token as opposed to the traditional recurrent neural network (RNN) methods that use only the previously seen tokens in the sentence in order to predict the next token. The BERT framework, which is unified across various NLP tasks, has two steps that are pre-training and fine-tuning (Fig 2.8). In the pre-training stage, the model is trained on a large corpus, and at the fine-tuning stage, these pre-trained parameters will initialize the BERT model, and then the parameters are fine-tuned in a downstream NLP task. The pre-training stage of BERT is done by two unsupervised tasks, which are depicted on the left side of the Figure 2.8. The first objective of BERT is called masked language modelling (MLM), in which 15% of the tokens of the input sequence are randomly masked, and the job of the system is to predict those masked tokens. To help the model to better generalize, the model randomly replaces some of these masked tokens with another token and tries to predict the correct word in that position, and some of the words remain unchanged from their original token. The other objective of BERT is a next-sentence prediction (NSP) objective, which is important in tasks that require an understanding of the relationship between a pair of sentences. When we pass two sentences to BERT as inputs, BERT predicts whether the second sentence is the actual next sentence of the first sentence. At the fine-tuning stage, by passing the appropriate input and output to the BERT model, we can fine-tune all the parameters end-to-end. We can use a shallow neural network at the top of the BERT as a classifier and achieve great results [23].

There are two model sizes of BERT published. The smaller one is referred to as $\text{BERT}_{\text{Base}}$, and the larger one is $\text{BERT}_{\text{Large}}$ which achieved state-of-the-art results

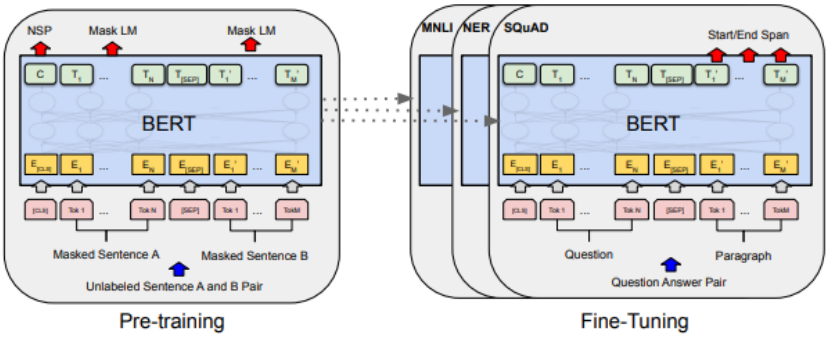

Figure 2.8: The pre-training and fine-tuning steps of the BERT model [23].

[23]. The size of $\text{BERT}_{\text{Base}}$ and $\text{BERT}_{\text{Large}}$ hidden layers are 768 and 1024, respectively. The input to the BERT model has a special format. We have to add a special [CLS] (classification) token to the begining of the first input sentence, and the final hidden state of this token can be used as a sentence representation in classification tasks. The words in a sentence are passed as an input to BERT, and the input flows up the stack of encoders. The BERT model architecture is designed in a way to be able to accept one sentence or pair of sentences. We separate the sentences with a special [SEP] token indicating the boundary of the two given sentences. If we have only one sentence, the [SEP] token is placed at the end of the given sentence. In order to feed the sentences to BERT, we have to add a learned embedding to each token, indicating whether they belong to the first or second sentence referred to as the segment embedding. The positional embeddings are learned vectors for every possible position which allow BERT to know the relative position of words in a sentence. By adding the current token, segment, and position embeddings, we will construct the input representation of each token (Fig 2.9).

RoBERTa (Robustly Optimized BERT Pretraining Approach) is a replication study of the BERT model [45]. The idea behind this model is that the BERT model is undertrained, and by using the new model, which the authors called RoBERTa, they achieved new state-of-the-art on downstream tasks like question answering (SQuAD

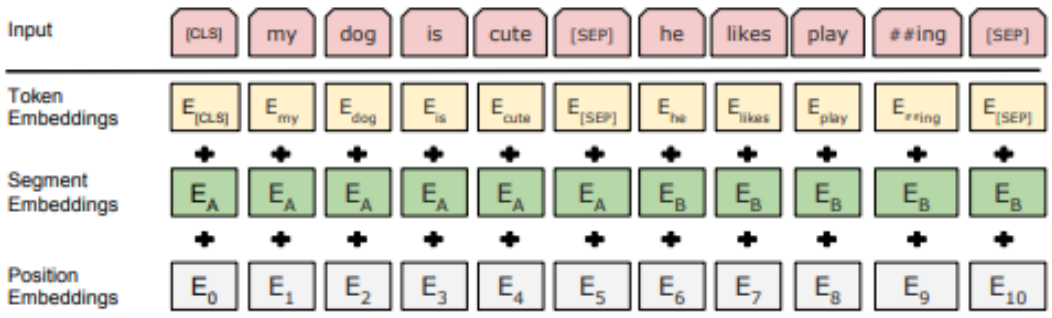

Figure 2.9: The input representation of the BERT model [23].

[56]) and reading comprehension (RACE [38]). The first modification of RoBERTa is allowing the model to train longer with bigger batches and more data. As we mentioned earlier, one of the objectives of the BERT pre-training stage is the next sentence prediction (NSP) task. As the second modification of the RoBERTa model, this objective was removed. The next modification is letting RoBERTa train on longer sequences, and instead of using the static masking of the BERT model, RoBERTa uses a dynamic masking pattern. This means that every time a sequence is passed into the model, a different masking pattern is created.

DistilBERT [62] is a distilled version of BERT with the same general architecture as BERT. In this model the token embeddings along with some other layers of the original BERT model are removed and the number of layers are half of the number of BERT layers. The next sentence prediction is removed for DistilBERT and it is trained on very large batches with dynamic masking. This model, which is a compressed version of BERT, is faster than BERT and it needs less computational resources. Another variant of BERT is BART [42] which is a denoising autoencoder that is responsible to map a corrupted document derived from an original document to the original document. This model achieved state-of-the results on a number of text generation tasks.

The power of the BERT and RoBERTa models is that by passing the proper task-specific inputs and outputs into the model, we do not need to modify the architecture

of the models for downstream tasks, and we can then fine-tune all the parameters end-to-end for a few epochs just by adding one additional task-specific output layer. In addition to the fine-tuning approach, BERT and RoBERTa can also be used as a form of feature extractor that can create contextualized embeddings, much like ELMo. In the feature-based approach, word representations can be extracted from the pre-trained model and they can be used as inputs to the architecture of the downstream tasks. This model has been shown to slightly underperform the fine-tuning approach [23, 54].

2.2 Multiword Expressions

In this section, we focus on the works done on multiword expressions (MWEs). MWEs consist of a combination of words, and their meanings may not be predictable from the meaning of each component word separately so that they may display some form of idiomaticity. This causes multiword expressions to be very difficult for NLP systems [71]. Previous works on MWEs have focused mainly on two subtasks, which are MWE extraction and MWE identification [4]. In MWE extraction, we are dealing with finding MWEs in text corpora, but in the identification task, we are working on annotating MWEs (tokens) in running text [18]. Machine translation [13], information retrieval [51], and opinion mining [9] can benefit from knowledge of multiword expressions.

Much research in NLP has also focused on identifying whether a multiword expression is compositional (e.g., *credit card*) or not. An MWE is non-compositional if its meaning cannot be predicted from the meaning of its component words separately. As an example, the meaning of the idiom *shoot the breeze* ("to chat") does not relate to the meanings of its component words, *shoot* and *breeze* [26]. The level to which the meaning of an MWE can be determined by combining the meanings of

its components shows the degree of compositionality of an MWE [4]. Some studies focus on this problem at the type level where we see a multiword expression out of context [5, 59, 60, 22] and some research considers MWE compositionality at the token level and sees instances of the MWE in context [25, 50].

Some previous works in this area look at linguistic signals such as the lexical fixedness of non-compositional MWEs [71], or the lexical flexibility of productive noun compounds [39] to classify MWEs at the type level. Various studies concentrate on detecting the compositionality of noun-noun MWEs like *ivory tower* [5], verb-particle constructions like *run away* [7, 21] and verb-noun pairs like *make a mistake* [32]. [26] studied the problem of MWE type classification for verb-noun combinations (VNCs). They studied the properties of four classes of VNCs and developed statistical measures to quantify these properties. After that, the measures were used as features to train a decision tree classifier.

[32] assumed in a supervised approach that we can determine whether an MWE is compositional or not with respect to its local linguistic context. In this study, the authors compute a representation for a multiword expression as the way it is used in a context and another representation of its meaning with respect to its component words. By measuring the similarity between the mentioned representations, they can distinguish compositional and non-compositional multiword expressions. At the token-level, much work has considered unsupervised and supervised approaches to predicting the idiomaticity of MWEs drawing on the context in which they occur. It is important for NLP applications such as machine translation [30] and sentiment analysis [74] to determine whether a particular MWE is used idiomatically with respect to its usage in text.

[19] utilized lexico-syntactic fixedness (i.e., canonical forms) of MWEs and the distributional hypothesis to perform token-level unsupervised classification of VNCs. Having information about the overall lexico-syntactic behaviour of an expression

can be used to automatically identify whether an MWE is used idiomatically or literally.

The unsupervised statistical approach of [25] classifies a particular type of multiword expression, verb-noun pairs, at the token level as idiomatic if it occurs in one of its pre-determined canonical forms for that expression and as literal otherwise. They identify canonical forms based on the lexico-syntactic patterns — with respect to the determiner, number of the noun, and voice of the verb — that a verb-noun idiomatic combination frequently occurs in. Their method shows that the idiomatic expression is more likely to appear in the canonical form, but this is not the case for literal expressions. In the following examples of *lose head*, you can see 1 is an idiomatic usage which is in the canonical form, in which the verb *lose* has active voice, the determiner is null and the noun is in the singular form. In contrast 2 is a literal combination that is not in the canonical form.[1]

1. I'm not going to lose my head and try and rush out at the first chance.

2. Within seconds her blonde head was lost in the crowds

[27] proposed another unsupervised approach based on k-means clustering, which clusters usages of verb-noun combinations (VNCs) which are represented based on word embeddings [48]. If the majority of the instances in a cluster are in canonical form, the label of the cluster is idiomatic, and if the instances are mainly in non-canonical form, the label of the cluster is literal. The results of this approach do not consistently outperform the unsupervised approach proposed by [25] based on canonical forms, which is a strong baseline.

[44] proposed a generalized model based on semantic compatibility to build an idiom usage recognizer. They train a continuous bag-of-words (CBOW) model without any supervision to identify whether an MWE instance is used figuratively or literally.

[1]These examples are taken from the VNC-Tokens dataset [20].

They treat each multiword expression as a single word and train a model on large text corpora to predict the semantic compatibility between the context and that single word. After that, they used the trained model to find the similarity between the context and the literal sense of the idiom.

Some works are further focused on a particular type of MWE. [24] proposed a supervised approach using a support vector machine (SVM) for classifying instances of multiword expression tokens focused on VNCs using various combinations of linguistically motivated features such as part-of-speech tags, lemma form, and named entities.

In the same vein as the former approach, [66] focus on supervised token-based identification of MWEs, again VNCs. This study showed that lexical and syntactic context representations derived from word embeddings, and the information gained from external arguments of the verb and the noun elements of the expression give excellent performance in the classification task.

[61] proposed that embedding the entire sentence that a VNC occurs in using skip-thoughts vector [35] representations serves as a good feature for classifying VNC instances as idiomatic or literal with SVM and k-nearest neighbours.

[33] proposed a model based on word embeddings to classify instances of VNCs as idiomatic or literal. In order to represent the context of an MWE, they used a simpler approach than [61] by averaging the embeddings of the surrounding words in that sentence. The results of their work showed substantial improvements when they incorporate the knowledge of lexico-syntactic fixedness based on the method of [25] with the embedding-based approach they propose. They trained word2vec's skip-gram [48] on a Wikipedia corpus with more than 2 billion tokens. They used the VNC-Tokens dataset [20] that consists of sentences from the British National Corpus [12] with instances of VNCs annotated as idiomatic or literal. They considered a supervised binary classification task using a support vector machine (SVM).

Recently contextualized embeddings have been applied to problems in multiword expressions. The work done by [64] is on multiword expressions using contextualized embeddings. They studied various representation methods for spans of tokens and found that contextualized embeddings can better capture the information needed for predicting various MWE properties, but do not consider VNC idiomaticity.

Other recent work has specifically focused on VNCs. [52] leveraged contextualized embeddings to represent context words and the target multiword expression in order to automatically recognize idiomatic tokens. Based on their hypothesis, the inner product of context word vectors with the vector representing a target expression in the literal case should be larger than for idiomatic expressions since vector representations of literal usages predict well local contexts, thereby distinguishing literal from idiomatic usages.

[28] propose a supervised method to classify multiword expressions based on their context as idiomatic or literal. In this approach, the lexicalized component words of VNC instances are merged, and they are represented using word and contextualized [47, 23] embeddings in order to take the context of each multiword expression into account. Merging the VNC component words shows some improvements, but their results do not outperform the results of [33].

The most similar work to ours is the approach proposed by [37] to represent the instances of potentially-idiomatic expressions (PIEs), including VNCs, based on the BERT [23] contextualized embeddings model. They investigate whether the information captured in these types of embeddings can assist in the task of classifying the usages of PIEs as idiomatic or literal by supervised and unsupervised classifiers. However, they do not consider evaluation on expressions that are unseen during training. As we may not have enough annotated training data for every VNC type, having a model that is able to generalize to VNCs that are unseen during training is important.

In our work, we focus on distinguishing literal from idiomatic instances of VNCs. Specifically, we also consider expressions that are unseen during the training stage in our experiments.

Chapter 3

Proposed Model

Verb-noun combinations (VNCs) are potentially idiomatic expressions whose meanings may not be directly related to their component words. The fact that VNCs are commonly used in language, and their meanings are often not predictable, makes them an essential topic of research for NLP [63]. This thesis proposes a supervised approach to distinguish idiomatic and literal usages of VNCs in a text based on contextualized representations, specifically BERT [23] and RoBERTa [45], which are powerful language models. We evaluate the performance of BERT and RoBERTa contextualized representations by fine-tuning both models to automatically identify whether an MWE is idiomatic or literal in a running text.

3.1 Contextualized Embedding

Before considering how to represent VNC instances with contextualized embeddings, we first discuss the BERT and RoBERTa tokenization models which affect how we do this. When tokenizing a word under the BERT [23] and RoBERTa [45] models, the tokenizer first tries to break the word into the largest possible subwords that exist in the vocabulary. After that, it will split the word into individual characters. So a given word can be split into multiple pieces. At the beginning of every sequence, a special

[CLS] token appears. This can be used as the aggregate sequence representation for downstream classification tasks.

In our first approach, we represent a VNC token instance using the [CLS] token for the sentence in which it occurs. The [CLS] representation, which is a 768-dimensional vector, is passed to the network we build on top of the pre-trained model. We refer to this approach as "CLS". The details of this model will be discussed in the next section.

After considering the [CLS] representation to represent each token instance of a VNC, we further consider the representation of separate component words of the VNC, i.e., the verb and the noun. In our next approach to represent a given VNC instance, we first form a representation of each of its verb and noun component words by averaging the embeddings of their pieces.

We then study two approaches to combining the representations of the verb and noun. Specifically, we consider averaging and concatenating the verb and noun representations. Each of the verb and noun representations is a 768-dimensional vector. When averaging the vectors of the verb and noun, as a result, we have a 768-dimensional vector to represent a VNC. We refer to this approach as "Average". When concatenating the verb and noun representations, we have a 1536-dimensional vector, and we refer to this approach as "Concat".[1]

We also considered other variants of BERT. DistilBERT [62] is a compressed version of BERT that only uses six transformer layers compared to 12 layers for $\text{BERT}_{\text{Base}}$, and has lower resource requirements. The preliminary results using DistilBERT were not as good as those for BERT and RoBERTa. We also considered BART [42], which uses an arbitrary noising function and learns a model to reconstruct the original text. Again the results did not improve over BERT and RoBERTa. We therefore do not

[1]In preliminary experiments on development data, we also considered concatenating and averaging these representations with the representation of the CLS token, but this did not give improvements, so we do not further discuss these approaches.

further consider distilBERT or BART.

3.2 Supervised Model

We aim to investigate the strength of the contextualized representations we formed for VNC token instances that we discussed earlier, at capturing information about the idiomaticity of VNCs. We test this in a supervised system for classifying the usages of a VNC as idiomatic or literal.

3.2.1 Fine-tuning

Our approach is based on fine-tuning pre-trained BERT and RoBERTa models for binary classification of VNC token instances. In this approach, a classification component is added to the pre-trained model, and all the parameters are fine-tuned on a downstream task. For our classification component, we use two fully-connected layers on top of either BERT or RoBERTa to classify instances of VNCs as idiomatic or literal.

The first layer of our network has the same dimensionality as the representation of the VNC (i.e., 768 for CLS and Average, and 1536 for Concat) and uses the ReLU activation function. The second layer has 512 dimensions and uses the softmax activation function.

Our model is illustrated in Figure 3.1 with x as input to the model which is the representation of a VNC instance and y_1 and y_2 are the labels idiomatic and literal.[2] The canonical form feature [25] we discussed in the previous chapter has been shown to be informative as to whether an instance of a VNC is idiomatic or literal [33]. We therefore also consider whether incorporating information about the lexico-syntactic

[2]We also experimented with a feature-based approach in which the representations of VNC instances are extracted from the pre-trained model and then used in an SVM classifier. Preliminary results showed relatively poor performance, and we therefore decided not to further consider this approach.

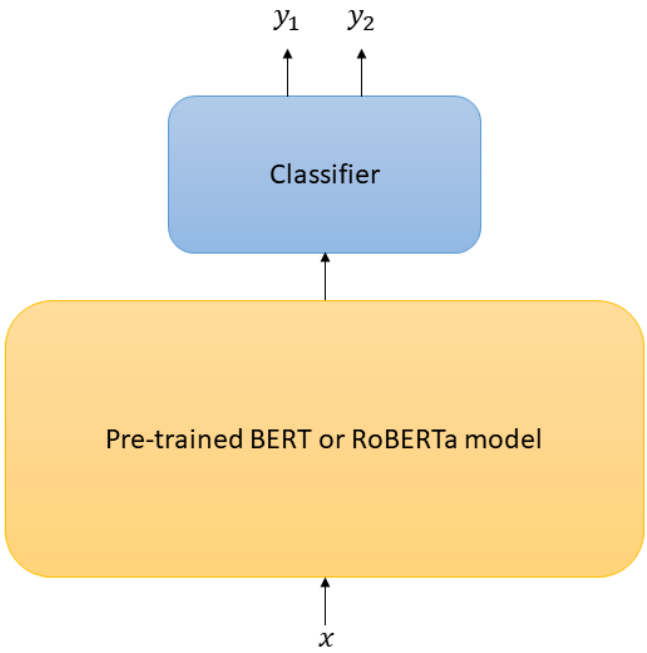

Figure 3.1: Our proposed model

fixedness of VNCs into our approach gives improvements. Specifically, we concatenate a single binary feature indicating whether a VNC usage is in a canonical form, referred to as CF, with each of the various representations of the VNC based on contextualized embeddings. In this case we represent a VNC with a 769-dimensional vector for the "CLS" and "Average" approaches, and a 1537-dimensional vector for the "Concat" approach.

The pre-trained BERT and RoBERTa models we use have 12 layers. In all the experiments we discuss, we use the representations of the last layer of the pre-trained model (i.e., 12-th layer). We then further study the effect of using different hidden layers of the BERT and RoBERTa models.

Chapter 4

Experimental Setup

In this chapter, we will discuss the datasets we used in our work. We will then discuss the metric we used for evaluating our models, and then the implementation of our models and parameter settings used.

4.1 Dataset and Evaluation

The dataset we used for our work is the VNC-Tokens dataset [20] which is the same dataset that is used by [33, 61]. This dataset contains 53 different VNC types, and their instances are extracted from the British National Corpus [12] and have been manually labelled at the token level as to whether they are literal or idiomatic usages. There are 53 expressions in the VNC-Tokens dataset, and they are divided into three subsets: DEV, TEST, and SKEWED. The SKEWED part of the dataset includes 25 expressions with many more instances in one class than the other. DEV and TEST include 14 expressions each that are more balanced between their idiomatic and literal usages, and we only consider DEV and TEST following previous work [61, 33]. The DEV and TEST parts of the dataset include a total of 594 and 613 VNC tokens, respectively, that are annotated as either literal or idiomatic usages.[1]

[1]Both DEV and TEST also contain instances that are annotated as "unknown"; following [25]

The details of the expressions and the number of instances annotated as idiomatic and literal in DEV and TEST are given in Table 4.1.

In our first experiment, we consider the same setup as [33], referred to as "all expressions" here. For each of DEV and TEST, we randomly partition the instances into training (roughly 75%) and testing (roughly 25%) sets, keeping the ratio of idiomatic to literal usages of each expression balanced across the training and testing sets. We repeat this random partitioning ten times.

We do not expect to have annotated instances of all VNC types, and this limits the applicability of models developed for the all expressions setup. Therefore, we are particularly interested in determining whether a supervised model is able to generalize to expressions that were unseen during training. Here we consider an experimental setup proposed by [27], referred to here as "unseen expressions". Here we hold out all instances of one VNC type for testing and train on all instances of the remaining types (within either DEV or TEST). We repeat this fourteen times for each of DEV and TEST, holding out each VNC type once for testing.

For both experimental setups — i.e., all expressions and unseen expressions — we train and test models on DEV for preliminary experiments and setting parameters. We then report the final results by training and testing models on TEST.

The idiomatic and literal classes are roughly balanced. Following [33] we therefore use accuracy as our evaluation metric. For the all expressions setup, we report average accuracy across the 10 runs. In the unseen expressions setup, we repeatedly hold out each expression until all instances of each expression (within either DEV or TEST) have been classified and then compute the accuracy.

---

we exclude these instances from our study.

Set	Expression	#Idiomatic	#Literal	Total
DEV	blow_trumpet	19	10	29
	find_foot	48	5	53
	get_nod	23	3	26
	hit_road	25	7	32
	hit_roof	11	7	18
	kick_heel	31	8	39
	lose_head	21	19	40
	make_face	27	14	41
	make_pile	8	17	25
	pull_leg	11	40	51
	pull_plug	45	20	65
	pull_weight	27	6	33
	see_star	5	56	61
	take_heart	61	20	81
	Total	362	232	594
TEST	blow_top	23	5	28
	blow_whistle	27	51	78
	cut_figure	36	7	43
	get_sack	43	7	50
	get_wind	13	16	29
	have_word	80	11	91
	hit_wall	7	56	63
	hold_fire	7	16	23
	lose_thread	18	2	20
	make_hay	9	8	17
	make_hit	5	9	14
	make_mark	72	13	85
	make_scene	30	20	50
	pull_punch	18	4	22
	Total	388	225	613

Table 4.1: The number of token instances in each class (i.e., idiomatic and literal) for each expression in the DEV and TEST datasets.

4.2 Vector representations

In order to prepare a sentence to be passed to BERT and RoBERTa, first, these models tokenize the sentence. In Figure 4.1 we show how BERT handles a single sentence in order to pass it to its pre-trained model. The BERT and RoBERTa models receive a fixed length of sentence as input. This length is set to the maximum length of sentences from the dataset. For sentences that are shorter than the maximum length, we add padding tokens to the sentences to make up the length. Then the BERT and RoBERTa tokenizers split the sentence into tokens which might split a word into multiple pieces as shown in Figure 4.1. In the next step, each token is replaced with its id as present in the BERT and RoBERTa tokenizer vocabularies. We mentioned earlier in Section 3.1 that the BERT and RoBERTa tokenizers can break words into pieces. Therefore, to represent a word, we average the embeddings for its pieces.

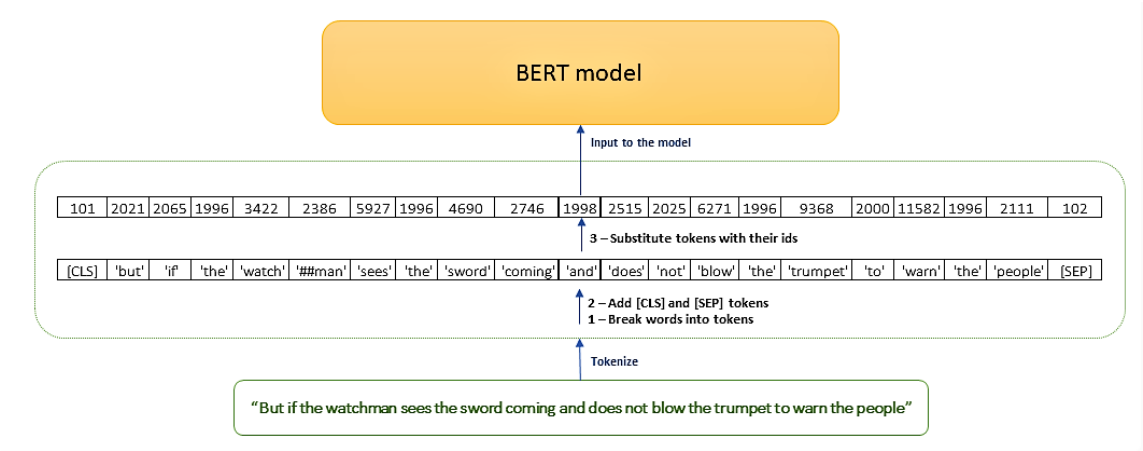

Figure 4.1: The tokenization process of BERT to prepare a sentence to pass it to the BERT model [2].

We consider twelve different representations for VNC instances. We consider BERT and RoBERTa, in 3 different approaches, optionally incorporating information about lexico-syntactic fixedness. The first approach is to represent each token instance of a VNC with the [CLS] token and use the 768-dimensional vector representation

corresponding to this token. We refer to this approach as "CLS". Then we pass this vector representation to the classification component of the system to classify an instance as idiomatic or literal. We further consider concatenating a single binary feature indicating whether a VNC usage is in a canonical form, referred to as CF, with each of the representations of the VNC based on the CLS token to find out whether incorporating information about the lexico-syntactic fixedness of VNCs into our approach gives improvements.

Next, we consider representing an instance of a VNC using the representations of VNC component words, i.e., the verb and noun. Here, we represent a VNC by averaging the representations of its verb and noun component words to form a 768-dimensional vector representation. Moreover, we again consider concatenating the CF feature to this representation to form a 769-dimensional vector representation which incorporates lexico-syntactic fixedness information. In the last approach, we represent a VNC by concatenating the representations of its verb and noun component words to form a 1536-dimensional vector representation. We again consider concatenating the CF feature to this representation to form a 1537-dimensional vector representation. Then we pass each of these representations to the classification component to classify the VNC instance as idiomatic or literal. In Table 4.2 a summary of the approaches we discussed here is given.

4.3 Implementation and Parameter Settings

We use Huggingface [75] implementations of BERT and RoBERTa. For BERT, we use bert-base-uncased, which is pre-trained on lower-cased English text. This model has 12 layers and a hidden layer size of 768 with 110M parameters overall. For RoBERTa, we use roberta-base, which has the same number of hidden layers, and hidden layer size, as bert-base-uncased.

Model	Representation	Canonical Form
BERT	CLS	-CF
		+CF
	Concat	-CF
		+CF
	Average	-CF
		+CF
RoBERTa	CLS	-CF
		+CF
	Concat	-CF
		+CF
	Average	-CF
		+CF

Table 4.2: A summary of the twelve approaches we experiment with can be seen here."-CF" means we do not incorporate the canonical form feature and "+CF" means we do use this feature.

In order to fine-tune pre-trained BERT and RoBERTa models for binary classification of VNC token instances, a classification component is added to the pre-trained model, and all the parameters are fine-tuned on a downstream task. As we discussed in Section 3.2.1 our classification component has two fully connected layers that take the context representations of VNCs as input and classify them as either literal or idiomatic.

We train our models using Adam optimizer [34] to minimize the cross-entropy loss. We use the default dropout of 0.5 for the network layers, which are on top of either BERT or RoBERTa. For fine-tuning, we use the parameters recommended by [23]. We use batch sizes of 8, 16, or 32; epochs between 2 and 4; and a learning rate of 2e-5, 3e-5, or 5e-5. We perform a grid search over these parameter settings on DEV. We report results for the best parameter settings on DEV and then use only these parameter settings for experiments on TEST. We repeat the experiments 10 times with different random seeds and report the mean accuracy and standard deviation over the runs.

Chapter 5

Results

In this section we compare our proposed approach discussed in Section 3.2 based on contextualized embeddings against the unsupervised approach of [25], which is based on canonical forms, and the supervised approach proposed by [33].

At first, we consider results for the all expressions experimental setup in which the model is tested on instances of expression observed during training. Results are given in Table 5.1. On each dataset, our proposed method based on contextualized embeddings outperforms all of the baselines. This finding indicates that contextualized embeddings are able to better capture knowledge of the idiomaticity of VNCs than previous approaches. On the DEV set, the best results are achieved using BERT with CLS. On TEST, the RoBERTa model using the Concat representation without the CF feature performs best of approaches that do not use this feature, while BERT using the CLS approach with the CF feature outperforms all other approaches. Overall, it looks like BERT using CLS performs well on both datasets. Although we see that Concat on TEST with RoBERTa and without the CF feature has a higher accuracy here, given the standard deviations that we observe, there does not appear to be a significant difference between approaches, and all the methods are quite similar.

Representation	Model	DEV		TEST	
		-CF	+CF	-CF	+CF
CLS	BERT	**90.7** ±0.53	**90.8** ±0.51	89.3 ±1.11	**89.8** ±0.71
	RoBERTa	88.3 ±0.96	89.9 ±0.66	88.6 ±0.87	89.0 ±0.48
Concat	BERT	89.8 ±0.90	90.4 ±0.80	89.2 ±0.74	89.2 ±0.63
	RoBERTa	89.3 ±0.78	89.7 ±0.71	**89.6** ±0.59	87.8 ±0.89
Average	BERT	90.1 ±0.54	90.5 ±0.75	89.4 ±0.50	89.7 ±0.49
	RoBERTa	89.6 ±0.85	89.6 ±0.86	89.4 ±0.74	88.51 ±1.27
Most Frequent Baseline		63.4	63.4	62.9	62.9
Unsupervised CForm [25]		75.0	75.0	71.1	71.1
Supervised word2vec [33]		82.5	85.6	81.5	84.7

Table 5.1: % accuracy and standard deviation for the all expressions experimental setup on DEV and TEST, for BERT and RoBERTa, for each approach to representing instances with and without the canonical form (CF) feature. % accuracy for the baselines is also shown. The best accuracy for each experimental setup, on each dataset, with and without the CF feature, is shown in boldface.

Next, we analyze the impact of the CF feature. Overall, the results are not consistent over the different representations. In some cases incorporating this feature improves performance, while in other cases the performance is similar or worse. Focusing on the method that performs best, BERT with CLS, using the canonical feature gives a small improvement on both DEV and TEST. However, this improvement is substantially smaller (i.e., in terms of percentage points) than the improvement obtained by [33] when using the CF feature. These findings suggest that contextualized embeddings can better capture the linguistic knowledge encoded in this feature than conventional word embeddings, which [33] use to represent VNC instances.

Turning to consider the different ways of representing VNC instances, Concat and Average do not improve the CLS approach on TEST, indicating that sentence-level representations are able to capture knowledge of VNC idiomaticity.

We now consider results for the unseen expressions experimental setup in which models are tested on instances that were not observed during training. Table 5.2 shows the results. On DEV, we get the best results using BERT with CLS, however, the accuracy for this approach drops substantially on TEST. RoBERTa with CLS per-

Representation	Model	DEV		TEST	
		-CF	+CF	-CF	+CF
CLS	BERT	**83.5** ±0.97	**83.4** ±0.65	78.6 ±1.78	79.8 ±1.55
	RoBERTa	81.8 ±1.60	82.4 ±1.20	**82.3** ±1.76	**80.6** ±2.35
Concat	BERT	83.2 ±1.07	83.1 ±0.74	78.7 ±1.47	78.1 ±2.70
	RoBERTa	81.6 ±1.19	80.8 ±2.42	79.2 ±1.70	77.5 ±2.64
Average	BERT	81.7 ±1.02	82.4 ±2.56	79.3 ±3.01	77.2 ±2.65
	RoBERTa	82.6 ±1.90	82.7 ±1.26	81.3 ±2.48	79.7 ±2.27
Most Frequent Baseline		60.9	60.9	63.3	63.3
Unsupervised CForm[25]		73.6	73.6	70.0	70.0
Supervised word2vec [33]		72.3	76.4	74.6	77.8

Table 5.2: % accuracy and standard deviation for the unseen expressions experimental setup on DEV and TEST, for BERT and RoBERTa for each approach to representing instances, with and without the CF feature. % accuracy for the baselines is also shown. The best accuracy for each experimental setup, on each dataset, with and without the CF feature, is shown in boldface.

forms more consistently across DEV and TEST and performs best on TEST. Further analyzing the impact of incorporating the CF feature for both BERT and RoBERTa on DEV we do not see a clear improvement when considering the standard deviation across runs. Focusing then on results on TEST for BERT and RoBERTa using CLS, without using the CF feature, both models outperform the baselines, including the approach of [33] which incorporates the CF feature, although this difference does not appear to be significant for BERT. Given the substantial improvements over the unsupervised CForm baseline [25] and the most-frequent class baseline (63.3% for TEST), these findings suggest that the classifiers (including the approach of [33]) have learned information about the idiomaticity of VNCs, that is not restricted to specific expressions, as in the case of the all expressions setup.

In an effort to understand why BERT using CLS performed relatively poorly on TEST, we compared the performance of the BERT and RoBERTa models using CLS representation on TEST without incorporating the CF feature in Table 5.3 based on accuracy across the various expressions. However, these results do not show any clear difference between the performance of the models on these expressions. Therefore,

Multiword Expression	BERT	RoBERTa
blow_top	75.4	76.8
blow_whistle	76.7	85
cut_figure	72.3	73
get_sack	73	85.6
get_wind	74.5	73.8
have_word	81.4	84.9
hit_wall	84.3	94.4
hold_fire	79.6	82.2
lose_thread	69	63.5
make_hay	84.7	86.5
make_hit	70	65.7
make_mark	86.4	86.7
make_scene	70.6	74.2
pull_punch	90	80.9

Table 5.3: % accuracy across various expressions in the TEST dataset for the unseen experimental setup for both the BERT and RoBERTa models using CLS representation without the CF feature.

we can conclude that there is not a specific expression that causes the drop in BERT performance on TEST data using the CLS representation.

Finally, we projected the computed vector from the hidden layer of the classification component discussed in Section 3.2.1 into two-dimensional space using t-SNE [69] for the unseen expressions experiment with RoBERTa using CLS without incorporating the CF feature. The reason we focus on the RoBERTa model using CLS is because it is the simplest model and as it performs roughly as well as, or better than, the more complex models (Concat and Average).

Figure 5.1 shows an example VNC type in the TEST dataset and shows that our model separates idiomatic and literal instances. We can see a separation of the idiomatic instances, which are blue dots in the upper right corner of the picture, and literal instances, which are red dots in the bottom left corner of the picture. This shows that the classifier is able to separate the idiomatic and literal representations in this 512-dimensional space which is projected down to two-dimensional space,

however, this separation is not perfect.

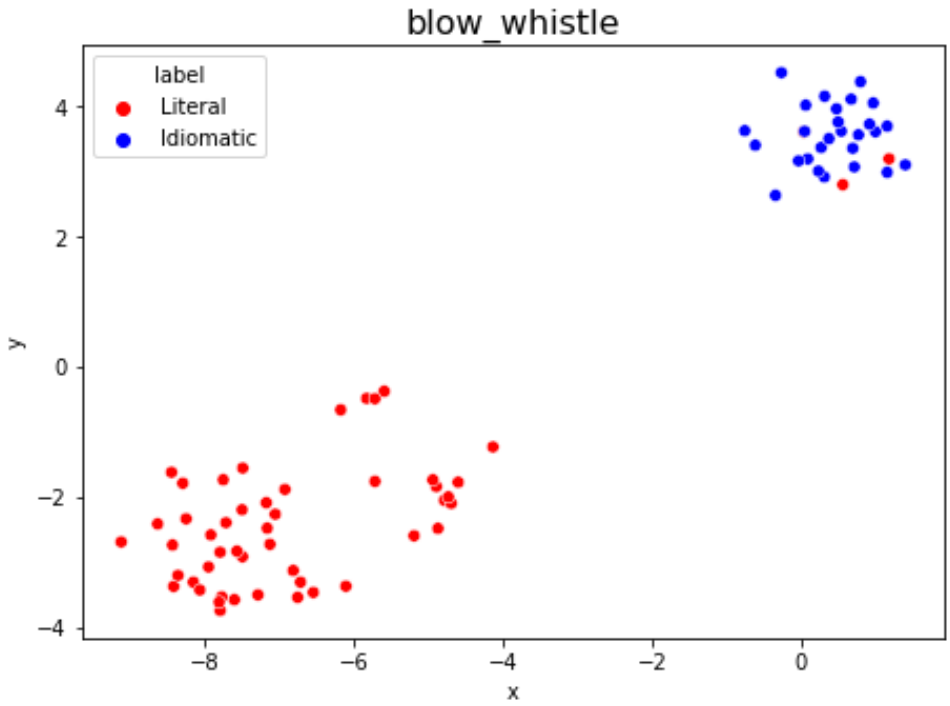

Figure 5.1: t-SNE projection of 512 hidden units in the classification component of the RoBERTa model using CLS in the unseen expression setup without incorporating the CF feature for the VNC, *blow whistle*, from the TEST set. Blue points are idiomatic VNC instances and red points are literal VNC instances.

We further projected the vector representations of the hidden layer of the classification component for the unseen expression without incorporating the canonical form feature for all the VNC instances of all the VNCs in TEST using t-SNE (Figure 5.2). We can conclude from this figure that we cannot see global separation between the idiomatic and literal instances, but rather the separation is much more local between idiomatic and literal instances.

In experiments, until now we have used representations from the final layer of BERT and RoBERTa. We now consider the effect of using different hidden layers, focusing on the unseen expressions setup in both the BERT and RoBERTa models using CLS. The reason behind exploring these approaches is that the best performance for the unseen expressions setup is achieved using one of BERT or RoBERTa using CLS

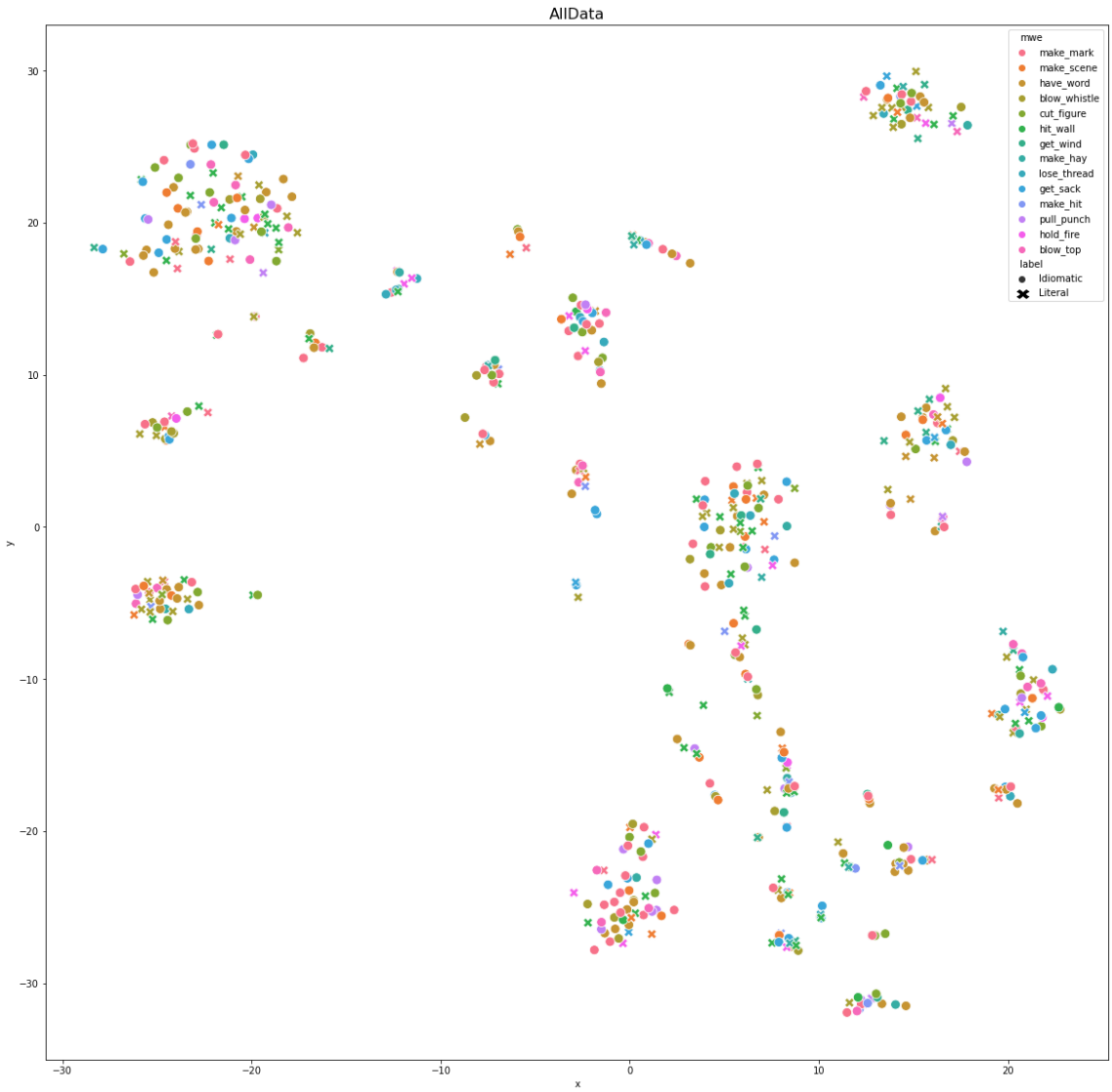

Figure 5.2: t-SNE projection of 512 hidden units in the classification component of VNC instances for all the instances in TEST for the unseen expressions setup with RoBERTa using CLS, without incorporating the CF feature. The circles represent the idiomatic VNC instances and the ×s represent the literal VNC instances.

which is the simplest approach. Results are shown in Table 5.4.[1] In all cases, except for BERT on TEST, the final layer performs best. This is in line with the findings of [31] that BERT can capture structural properties and that the upper layers of BERT encode semantic information. For BERT, where accuracy was low on TEST

[1]Results are only shown for layers 9–12. The overall trend for other layers is that lower layers achieve lower accuracy.

Model	Dataset	Layer			
		9	10	11	12
BERT	DEV	82.0	82.2	82.6	**83.5**
BERT	TEST	79.2	79.8	**80.2**	78.6
RoBERTa	DEV	75.6	78.2	79.8	**81.8**
RoBERTa	TEST	71.8	77.7	79.5	**82.3**

Table 5.4: % accuracy and standard deviation for the unseen expressions experimental setup on DEV and TEST using BERT and RoBERTa with CLS representations from the indicated layers. The best results for each model and dataset are shown in boldface.

relative to DEV in Table 5.2, on TEST the second last layer performs best.

Overall, considering the qualitative and quantitative results we can conclude that the approaches using contextualized embeddings give improvements over approaches that do not use contextualized embeddings. Using the CF feature does not improve the results and it appears that contextualized embeddings are able to capture the linguistic knowledge encoded in this feature. Moreover, our results outperform the baselines [25, 33] on the unseen expressions experiment, indicating that the model has learned information about the idiomaticity of VNCs generally, that is not restricted to the expressions it was trained on.

Chapter 6

Conclusion

MWEs have varying degrees of compositionality and they have received a lot of attention from researchers. However, there are relatively few resources, such as manually annotated corpora in various languages for MWEs, which makes working with them challenging. To build a robust natural language processing technology, handling MWEs is very important as knowledge of MWEs is essential in many downstream applications such as machine translation and information retrieval. In this thesis, we focus on VNCs, which consist of a verb with a noun in its direct object position. VNCs are a common kind of MWE in English and also cross-lingually.

We have studied two different experimental setups for classifying VNCs as idiomatic and literal. In the first experimental setup, "all expressions", we train and test on instances of the same VNC types. However, as we are particularly interested in determining whether a supervised model is able to generalize to expressions that were unseen during training, we consider another experimental setup, referred to as "unseen expressions". In this experimental setup, we hold out all instances of one VNC type for testing and train on all instances of the remaining types.

This study proposes a supervised approach to distinguish idiomatic and literal usages of VNCs in a text based on contextualized representations, specifically BERT

and RoBERTa, powerful language models. We represent a VNC token instance with three different approaches with these two models, including "CLS", "Concat", and "Average". In the "CLS" approach we use the representation of the [CLS] token of the BERT or RoBERTa model to represent a VNC token instance. For "Average" and "Concat" we combine the representations of the verb and noun by averaging and concatenating them, respectively. We also incorporate a canonical form feature to the contextualized embeddings to see whether adding this feature improves the performance or not. By adding this feature we incorporate information about the lexico-syntactic fixedness of VNCs. We evaluate the performance of BERT and RoBERTa contextualized representations by fine-tuning both models to automatically identify whether an MWE is idiomatic or literal in a running text.

The main research question of this study is *Does an approach to identifying VNC idioms that incorporates contextualized embeddings outperform prior methods that do not use contextualized embeddings?* We showed that the proposed supervised approach, in the "all expressions" experimental setup using contextualized embeddings, is able to outperform the previous best approach. We also experiment with incorporating the canonical form feature into our system, and we showed that contextualized embeddings capture this information, and that adding this feature to the contextualized embedding does not add extra information about VNC idiomaticity. Prior works in this area incorporate this feature with representations from standard word embedding methods and have obtained substantial performance increases by doing so.

The second research question that we answer in this study is *Is an approach to identifying VNC idioms that incorporates contextualized embeddings able to generalize to unseen expressions?* The results show that our proposed method in the "unseen expressions" experimental setup, can generalize to expressions that are unseen during the training stage. In order to do that, we leave each VNC idiom and its corre-

sponding sentences out of the training dataset, let the system label the instances of the VNC that is unseen during the training and repeat this for each VNC type. We also incorporate the canonical form feature to see whether adding this feature improves the system. This is an interesting experimental setup because we may not have enough annotated training data for every VNC type. Our findings indicate that the model has learned information about the idiomaticity of VNCs more generally, not just for specific expressions that are present in the training data. In other words, our model is able to generalize to VNCs that are unseen during training.

The contributions of this thesis can be clearly listed as follows:

1. Proposing an approach to identifying VNC idioms as idiomatic or literal that incorporates pre-trained contextualized embeddings.

2. Finding that contextualized embeddings are able to capture the linguistic knowledge encoded in the canonical form feature.

3. Proposing an approach to identifying VNC idioms that is able to generalize to unseen expressions.

For future work, different lines of research can be taken. First, in this study, we used BERT and RoBERTa as contextualized embeddings. We intend to explore other variants of BERT to see whether they improve our results. For instance, we use $\text{BERT}_{\text{Base}}$ as we mentioned in Section 4.3 and in the future we could use $\text{BERT}_{\text{Large}}$ as it is a larger and more complex pre-trained model than $\text{BERT}_{\text{Base}}$, we may get better results. In order to fine-tune BERT and RoBERTa for our classification task, we add a classification component on top of their models. We could consider a different neural network architecture for the classification component. Adding more layers or modifying the number of hidden units may improve the results. Other than these, we can also explore the BERT and RoBERTa layers more thoroughly as different layers of the BERT model capture different information [31]. We already explored

the performance of the individual layers of BERT and RoBERTa and reported their results in Table 5.4. However, we can also use the combination of representations of various layers by concatenating or averaging them. In this way, we can have a thorough analysis to see where the information about the idiomaticity of VNCs, and more generally MWEs, are encoded within BERT and RoBERTa.

Another future direction that we want to consider is to experiment with XLNet [76]. This model uses an improved training model and more training data compared to BERT. In this thesis, we are dealing with a particular text classification problem. We classify a VNC instance as idiomatic or literal. XLNet uses a permutation language modelling objective, in which all the tokens are predicted in random order, instead of a masked language modelling objective, in which only 15% of the tokens are masked and predicted, that is used in BERT and RoBERTa as discussed in Section 2.1.3. The reason behind using XLNet is that this generalized autoregressive pretraining method outperforms BERT in many tasks such as text classification [76].

We can also build a system that works cross-lingually in future work. In our model, we showed that we could train our model on English VNCs, and with that, we can classify the English instances of other VNCs that are not present during the training using BERT. Interesting future work would be to see as BERT is encoding knowledge of English VNCs that are not particular to some specific expression, would a model like multilingual BERT (mBERT) encode knowledge of idioms that isn't particular to any language? mBERT provides sentence representations for various languages, which are beneficial for many multi-lingual tasks [43]. For instance, we can train a model on English, German, French, and Spanish idioms and then evaluate our model on Dutch idioms.

The last future direction that we intend to consider is to conduct further research on a system that can be used for all types of multiword expressions and that is not just restricted to a specific type of multiword expression. We have only considered VNCs

but there are other common types of multiword expressions such as verb-particle constructions. In this case, the goal of this system would be to identify spans of tokens that are multiword expressions.

Bibliography

[1] Otavio Acosta, Aline Villavicencio, and Viviane Moreira, *Identification and treatment of multiword expressions applied to information retrieval*, Proceedings of the Workshop on Multiword Expressions: from Parsing and Generation to the Real World (Portland, Oregon, USA), Association for Computational Linguistics, June 2011, pp. 101–109.

[2] Jay Alammar, *A Visual Guide to Using BERT for the First Time*, `http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/`, 2019, Accessed: 2021-04-29.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Y. Bengio, *Neural machine translation by jointly learning to align and translate*, ArXiv **1409** (2014).

[4] Timothy Baldwin, *Compositionality and multiword expressions: Six of one, half a dozen of the other?*, Proceedings of the Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties (Sydney, Australia), Association for Computational Linguistics, July 2006, p. 1.

[5] Timothy Baldwin, Colin Bannard, Takaaki Tanaka, and Dominic Widdows, *An empirical model of multiword expression decomposability*, Proceedings of the ACL 2003 Workshop on Multiword Expressions: Analysis, Acquisition and

Treatment (Sapporo, Japan), Association for Computational Linguistics, July 2003, pp. 89–96.

[6] Timothy Baldwin and Su Nam Kim, *Multiword expressions*, Handbook of Natural Language Processing (Nitin Indurkhya and Fred J. Damerau, eds.), CRC Press, Boca Raton, USA, 2nd ed., 2010.

[7] Colin Bannard, Timothy Baldwin, and Alex Lascarides, *A statistical approach to the semantics of verb-particles*, Proceedings of the ACL 2003 Workshop on Multiword Expressions: Analysis, Acquisition and Treatment (Sapporo, Japan), Association for Computational Linguistics, July 2003, pp. 65–72.

[8] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin, *A neural probabilistic language model*, The journal of machine learning research **3** (2003), 1137–1155.

[9] Gábor Berend, *Opinion expression mining by exploiting keyphrase extraction*, Proceedings of 5th International Joint Conference on Natural Language Processing (Chiang Mai, Thailand), Asian Federation of Natural Language Processing, November 2011, pp. 1162–1170.

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

[11] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al., *Language models are few-shot learners*, arXiv preprint arXiv:2005.14165 (2020).

[12] Lou Burnard, *The British national corpus users reference guide*, Oxford University Computing Services, 2000.

[13] Marine Carpuat and Mona Diab, *Task-based evaluation of multiword expressions: a pilot study in statistical machine translation*, Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Los Angeles, California), Association for Computational Linguistics, June 2010, pp. 242–245.

[14] Francois Chaubard, Michael Fang, Guillaume Genthial, Rohit Mundra, and Richard Socher, *Lecture Notes: Part I Word Vectors I: Introduction, SVD and Word2Vec*, `https://web.stanford.edu/class/cs224n/readings/cs224n-2019-notes01-wordvecs1.pdf`, 2019, Accessed: 2021-04-29.

[15] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio, *On the properties of neural machine translation: Encoder–decoder approaches*, Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (Doha, Qatar), Association for Computational Linguistics, October 2014, pp. 103–111.

[16] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio, *Learning phrase representations using RNN encoder–decoder for statistical machine translation*, Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) (Doha, Qatar), Association for Computational Linguistics, October 2014, pp. 1724–1734.

[17] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa, *Natural language processing (almost) from scratch*, Journal of Machine Learning Research **12** (2011), 2493–2537.

[18] Mathieu Constant, Gülşen Eryiğit, Johanna Monti, Lonneke Van Der Plas, Carlos Ramisch, Michael Rosner, and Amalia Todirascu, *Multiword expression processing: A survey*, Computational Linguistics **43** (2017), no. 4, 837–892.

[19] Paul Cook, Afsaneh Fazly, and Suzanne Stevenson, *Pulling their weight: Exploiting syntactic forms for the automatic identification of idiomatic expressions in context*, Proceedings of the workshop on a broader perspective on multiword expressions, 2007, pp. 41–48.

[20] ———, *The vnc-tokens dataset*, Proceedings of the LREC Workshop Towards a Shared Task for Multiword Expressions (MWE 2008), 2008, pp. 19–22.

[21] Paul Cook and Suzanne Stevenson, *Classifying particle semantics in English verb-particle constructions*, Proceedings of the Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties (Sydney, Australia), Association for Computational Linguistics, July 2006, pp. 45–53.

[22] Silvio Cordeiro, Carlos Ramisch, Marco Idiart, and Aline Villavicencio, *Predicting the compositionality of nominal compounds: Giving word embeddings a hard time*, Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2016, pp. 1986–1997.

[23] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, *BERT: Pre-training of deep bidirectional transformers for language understanding*, Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers) (Minneapolis, Minnesota), Association for Computational Linguistics, June 2019, pp. 4171–4186.

[24] Mona Diab and Pravin Bhutada, *Verb noun construction mwe token classification*, Proceedings of the Workshop on Multiword Expressions: Identification, Interpretation, Disambiguation and Applications (MWE 2009), 2009, pp. 17–22.

[25] Afsaneh Fazly, Paul Cook, and Suzanne Stevenson, *Unsupervised type and token identification of idiomatic expressions*, Computational Linguistics **35** (2009), no. 1, 61–103.

[26] Afsaneh Fazly and Suzanne Stevenson, *Distinguishing subtypes of multiword expressions using linguistically-motivated statistical measures*, Proceedings of the Workshop on A Broader Perspective on Multiword Expressions, 2007, pp. 9–16.

[27] Waseem Gharbieh, Virendrakumar Bhavsar, and Paul Cook, *A word embedding approach to identifying verb-noun idiomatic combinations*, Proceedings of the 12th Workshop on Multiword Expressions, 2016, pp. 112–118.

[28] Reyhaneh Hashempour and Aline Villavicencio, *Token level identification of multiword expressions using contextual information*, Proceedings of the The Fourth Widening Natural Language Processing Workshop, 2020, pp. 23–25.

[29] W John Hutchins, *Machine translation: A brief history*, Concise history of the language sciences, Elsevier, 1995, pp. 431–445.

[30] Pierre Isabelle, Colin Cherry, and George Foster, *A challenge set approach to evaluating machine translation*, Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (Copenhagen, Denmark), Association for Computational Linguistics, September 2017, pp. 2486–2496.

[31] Ganesh Jawahar, Benoît Sagot, and Djamé Seddah, *What does BERT learn about the structure of language?*, Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (Florence, Italy), Association for Computational Linguistics, July 2019, pp. 3651–3657.

[32] Graham Katz and Eugenie Giesbrecht, *Automatic identification of non-compositional multi-word expressions using latent semantic analysis*, Proceed-

ings of the Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties (Sydney, Australia), Association for Computational Linguistics, July 2006, pp. 12–19.

[33] Milton King and Paul Cook, *Leveraging distributed representations and lexico-syntactic fixedness for token-level prediction of the idiomaticity of English verb-noun combinations*, Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) (Melbourne, Australia), Association for Computational Linguistics, July 2018, pp. 345–350.

[34] DP Kingman and J Ba, *Adam: A method for stochastic optimization. conference paper*, 3rd International Conference for Learning Representations, 2015.

[35] Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler, *Skip-thought vectors*, Advances in Neural Information Processing Systems 28 (C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds.), Curran Associates, Inc., 2015, pp. 3276–3284.

[36] Stefan Kombrink, Tomáš Mikolov, Martin Karafiát, and Lukáš Burget, *Recurrent neural network based language modeling in meeting recognition*, Twelfth annual conference of the international speech communication association, 2011.

[37] Murathan Kurfalı and Robert Östling, *Disambiguation of potentially idiomatic expressions with contextual embeddings*, Proceedings of the Joint Workshop on Multiword Expressions and Electronic Lexicons (online), Association for Computational Linguistics, December 2020, pp. 85–94.

[38] Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy, *RACE: Large-scale ReAding comprehension dataset from examinations*, Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing

(Copenhagen, Denmark), Association for Computational Linguistics, September 2017, pp. 785–794.

[39] Mirella Lapata and Alex Lascarides, *Detecting novel compounds: The role of distributional evidence*, 10th Conference of the European Chapter of the Association for Computational Linguistics, 2003.

[40] Quoc Le and Tomas Mikolov, *Distributed representations of sentences and documents*, Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32, ICML'14, JMLR.org, 2014, p. II–1188–II–1196.

[41] Omer Levy and Yoav Goldberg, *Dependency-based word embeddings*, Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) (Baltimore, Maryland), Association for Computational Linguistics, June 2014, pp. 302–308.

[42] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer, *BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension*, Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (Online), Association for Computational Linguistics, July 2020, pp. 7871–7880.

[43] Jindřich Libovickỳ, Rudolf Rosa, and Alexander Fraser, *How language-neutral is multilingual bert?*, arXiv preprint arXiv:1911.03310 (2019).

[44] Changsheng Liu and Rebecca Hwa, *A generalized idiom usage recognition model based on semantic compatibility*, Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 6738–6745.

[45] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov, *Roberta: A robustly optimized bert pretraining approach*, arXiv preprint arXiv:1907.11692 (2019).

[46] Thang Luong, Hieu Pham, and Christopher D. Manning, *Effective approaches to attention-based neural machine translation*, Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (Lisbon, Portugal), Association for Computational Linguistics, September 2015, pp. 1412–1421.

[47] Oren Melamud, Jacob Goldberger, and Ido Dagan, *context2vec: Learning generic context embedding with bidirectional LSTM*, Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning (Berlin, Germany), Association for Computational Linguistics, August 2016, pp. 51–61.

[48] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean, *Efficient estimation of word representations in vector space*, Proceedings of Workshop at the International Conference on Learning Representations, 2013 (Scottsdale, USA), 2013.

[49] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean, *Distributed representations of words and phrases and their compositionality*, Advances in neural information processing systems **26** (2013), 3111–3119.

[50] Grace Muzny and Luke Zettlemoyer, *Automatic idiom identification in wiktionary*, Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, 2013, pp. 1417–1421.

[51] David Newman, Nagendra Koilada, Jey Han Lau, and Timothy Baldwin, *Bayesian text segmentation for index term identification and keyphrase extrac-*

tion, Proceedings of COLING 2012 (Mumbai, India), The COLING 2012 Organizing Committee, December 2012, pp. 2077–2092.

[52] Jing Peng, Anna Feldman, and Hamza Jazmati, *Classifying idiomatic and literal expressions using vector space representations*, Proceedings of the International Conference Recent Advances in Natural Language Processing (Hissar, Bulgaria), INCOMA Ltd. Shoumen, BULGARIA, September 2015, pp. 507–511.

[53] Jeffrey Pennington, Richard Socher, and Christopher Manning, *GloVe: Global vectors for word representation*, Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) (Doha, Qatar), Association for Computational Linguistics, October 2014, pp. 1532–1543.

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

[55] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever, *Language models are unsupervised multitask learners*, OpenAI blog **1** (2019), no. 8, 9.

[56] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang, *SQuAD: 100,000+ questions for machine comprehension of text*, Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (Austin, Texas), Association for Computational Linguistics, November 2016, pp. 2383–2392.

[57] David Rozado, *Wide range screening of algorithmic bias in word embedding models using large sentiment lexicons reveals underreported bias types*, PloS one **15** (2020), no. 4, e0231189.

[58] Ivan A. Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger, *Multiword expressions: A pain in the neck for nlp*, Computational Linguistics and Intelligent Text Processing (Berlin, Heidelberg) (Alexander Gelbukh, ed.), Springer Berlin Heidelberg, 2002, pp. 1–15.

[59] Bahar Salehi and Paul Cook, *Predicting the compositionality of multiword expressions using translations in multiple languages*, Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity, 2013, pp. 266–275.

[60] Bahar Salehi, Paul Cook, and Timothy Baldwin, *Detecting non-compositional mwe components using wiktionary*, Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1792–1797.

[61] Giancarlo Salton, Robert Ross, and John Kelleher, *Idiom token classification using sentential distributed semantics*, Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (Berlin, Germany), Association for Computational Linguistics, August 2016, pp. 194–204.

[62] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf, *Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter*, arXiv preprint arXiv:1910.01108 (2019).

[63] Agata Savary, Manfred Sailer, Yannick Parmentier, Michael Rosner, Victoria Rosén, Adam Przepiórkowski, Cvetana Krstev, Veronika Vincze, Beata Wójtowicz, Gyri Smørdal Losnegaard, et al., *Parseme–parsing and multiword expressions within a european multilingual network*, 7th Language & Technol-

ogy Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics (LTC 2015), 2015.

[64] Vered Shwartz and Ido Dagan, *Still a pain in the neck: Evaluating text representations on lexical composition*, Transactions of the Association for Computational Linguistics **7** (2019), 403–419.

[65] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts, *Recursive deep models for semantic compositionality over a sentiment treebank*, Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (Seattle, Washington, USA), Association for Computational Linguistics, October 2013, pp. 1631–1642.

[66] Shiva Taslimipoor, Omid Rohanian, Ruslan Mitkov, and Afsaneh Fazly, *Investigating the opacity of verb-noun multiword expression usages in context*, Proceedings of the 13th Workshop on Multiword Expressions (MWE 2017), 2017, pp. 133–138.

[67] Erik F. Tjong Kim Sang and Fien De Meulder, *Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition*, Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003, 2003, pp. 142–147.

[68] AM Turing, *Computing machinery and intelligence*, Mind **59** (1950), 433–460.

[69] Laurens Van der Maaten and Geoffrey Hinton, *Visualizing data using t-sne.*, Journal of machine learning research **9** (2008), no. 11.

[70] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin, *Attention is all you need*, arXiv preprint arXiv:1706.03762 (2017).

[71] Joachim Wermter and Udo Hahn, *Paradigmatic modifiability statistics for the extraction of complex multi-word terms*, Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, 2005, pp. 843–850.

[72] Jason Weston, Samy Bengio, and Nicolas Usunier, *Large scale image annotation: learning to rank with joint word-image embeddings*, Machine learning **81** (2010), no. 1, 21–35.

[73] John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu, *Charagram: Embedding words and sentences via character n-grams*, Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (Austin, Texas), Association for Computational Linguistics, November 2016, pp. 1504–1515.

[74] Lowri Williams, Christian Bannister, Michael Arribas-Ayllon, Alun Preece, and Irena Spasić, *The role of idioms in sentiment analysis*, Expert Systems with Applications **42** (2015), no. 21, 7375–7385.

[75] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al., *Huggingface's transformers: State-of-the-art natural language processing*, arXiv preprint arXiv:1910.03771 (2019).

[76] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le, *Xlnet: Generalized autoregressive pretraining for language understanding*, arXiv preprint arXiv:1906.08237 (2019).

Vita

Candidate's full name: Samin Fakharian
University attended (with dates and degrees obtained): K. N. Toosi University of Technology, Iran, Bachelor of Computer Software Engineering, 2014-2018
Publications:
Samin Fakharian, Paul Cook, *Contextualized Embeddings Encode Knowledge of English Verb–Noun Combination Idiomaticity*, submitted to Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP 2021), 2021