Department of Computing and Electronic Systems, University of Essex

Automatic Detection of Idiomatic Language

Submitted as part of the requirements for:

CE902 Professional Practice and Research Methodology

Name: Jose Juan Zavala Iglesias

Supervisor: Vahid Abolghasemi / Aline Villavicencio / Ansgar Scherp

Date: 19 August 2019

Abstract. Idiomatic use of Verb-Noun Combinations (VNCs) has pose a challenge, particularly for the tasks of text translation and semantic parsing, for NLP researchers since these may also pose literal meaning under certain contexts. Since idiomatic VNC are known to exhibit high degrees of lexico-syntactic fixedness, we ran tests for extraction of idiomatic VNC Candidates using these metrics alone with significant results. We then propose that the addition of these fixedness measures to feature vectors used on past research, consisting of embeddings from Word2Vec, Siamese CBOW, and Skip-Thoughts models for the task of supervised classification of idiomatic vs literal use may increase performance, but we could not find a significant increase over the original vector. To improve on past research, we a) Test the performance of embeddings trained on the target Corpora; and b) Test the usage of ELMo embeddings, a new embeddings model using bi-LSTMs that claims to carry semantic context; both tests were successful with significant positive results. This success of classifying sentence embeddings using Support Vector Machines also motivated the use of unsupervised clustering algorithms to form idiomatic and literal clusters for classification using a random seed, which shows results that compete with state-of-the-art models such as CForm when using a high number of clusters. Lastly, we propose the use of the Cosine Similarity between the VNC and the original target sentence to detect semantic differences that may help to classify literal vs idiomatic usage; this proposal fell short of the performance of previous, unsupervised, models such as CForm. We perform all these experiments over the VNC-Tokens Dataset.

Keywords: *Multiword Expressions, Verb-Noun Combinations, Idiom Detection, Word Embeddings, Support Vector Machines, k-Means Clustering, Pointwise Mutual Information, Cosine Similarity, BNC-XML Corpora, VNC-Tokens Dataset*

Table of Contents

1 Introduction

In recent years, the field of Natural Language processing has been strongly focused on the tasks of machine translation and semantic parsing in order to more effectively understand human language for analysis and information retrieval. However, idiomatic use of phrases has always been a problem in the field, since its appearance in text is challenging in the sense that it encapsulates meaning that is not easily disambiguated. Usage of idiomatic language in text and conversation comes natural for individuals with enough context of the phrases used, however, for a computer system these are nearly impossible to detect since idiomatic phrases have a meaning unrelated to the individual meanings of the conforming tokens; syntactically-idiomatic phrases can also lead to parsing problems, due to non-conformance with common patterns [1].

Idiomatic phrases fall under the study of Multiword Expressions (MWEs), which are idiosyncratic combinations of multiple words whose interpretation crosses token boundaries (such as spaces), which means that their use has little to no variation across texts due to their inherent fixedness [1] [2] [3] [4]. In particular, there has been interest of MWEs in the form of Verb-Noun Idiomatic Combinations (VNICs), which consist of a single verb with a noun in its direct object position, with usually a determiner in between (e.g. Break a leg) [2] [4]. Research on idiomatic phrases focuses on VNICs since they are a common type of semantically-idiomatic MWE across several languages [3] [4]. Note that non-literal usage is not restricted to VNICs solely (e.g. *Time flies* when you are having fun), but research has found significant correlations among its usage with different phrases, which makes them a great target for exploratory research [3].

To demonstrate the complexity, and importance, of VNIC detection, take for example the VNC <*break, leg*> on the following two sentences, in which the first one has idiomatic usage while the second one has literal meaning:

1. Go to the stage and break a leg.
2. John fell from the stage and broke a leg.

This has motivated previous research to frame the problem as a supervised binary classification problem [2] [5] [6]. Unsupervised methods have also seemed some degree of success on the past [3] [7], but their reliability falls short under the performance of the supervised methods. Both implementations differ from the way they create the feature vectors used from classification of VNCs for idiomatic and literal usage, however, we have observed that some of these methods may complement each other for the emergence of more powerful models.

Another relevant task is the one of automatically identifying new potential VNC that are subject to idiomatic use over time [3]. Language is vast and its use changes across generations, so the ability of retrieving emerging idiomatic phrases as soon as possible by quickly analysing tons of text data and using a small amount of human resources is a necessary, if not crucial, task for machine translation and semantic parsing.

It is of interest to know whether if measures of a VNC's Lexical and Syntactical Fixedness can by themselves be successfully utilized to extract potential idiomatic phrases from a vast Corpora. As well, how this syntactic and semantic information is carried to

the popular word-embeddings models for the task of classifying a sentence based on the presence of an idiomatic VNC among its tokens.

This project aims to first demonstrate the power of unsupervised language models that work over the inherent Lexical and Syntactical Fixedness of VNICs working alone over classical algorithms (e.g. PMI) in the task of detecting idiomatical language from a manually unreviewed Corpora [3]. This will also aim to reflect the relevance of the syntactical and lexical fixedness present in VNICs, which will be explained in earlier sections of this dissertation. Next, we will show how word-embedding models are capable to detect the degree of fixedness and semantical ambiguity on these target sentences and serve as ideal feature vector candidates for a binary-classification solution to this problem using algorithms that can discriminate over boundaries found in vector space, such as supervised-learning Support Vector Machines and semi-supervised k-Means Clustering with a random seed. Lastly, we push these embeddings trying to identify idiomatic use by calculating the cosine similarity between the context sentence and the VNIC present in said sentence.

We evaluate these claims on the VNC-Tokens Dataset, which contains almost 3,000 English sentences that use one of 53 VNCs. This dataset was constructed over the BNC-XML dataset, which will be used to extract the required syntactical and lexical information for all the VNCs available.

This thesis presents a few improvements over previous methods used by researchers over which we worked upon. Our proposed approaches reached a great degree of success. The VNIC extraction metrics showed a significant increase over the baseline as expected, based solely on the scores achieved without considering filtering any particular verbs for idiomatic phrases as previously researched [3]. On the idiomatic vs literal problem, the supervised method using the proposed sentence embeddings also showed a significant increase over previous research, demonstrating that embeddings models trained on the particular used context (i.e. Word2Vec trained on BNC-XML), as well as models that weight heavily the context in which an expression is being used (i.e. ELMo) have a significant improvement over other, more generic, embedding models.

2 Background

In this section we explain key concepts to understand the development of this dissertation. Most importantly, this section should explain the meaning of what we mean as "idiomatic" use of a phrase and important characteristics present in most idiomatic word combinations.

2.1 Multiword Expressions

Formally, Multiword Expressions are lexical items that: (a) can be decomposed into multiple lexemes; and (b) display lexical, syntactic, semantic, pragmatic and/or statistical idiomaticity [1] [4]. Regarding property (b), *idiomaticity* refers to the deviation from the basic properties of the individual lexemes of the MWE [4]. This deviation can occur at any of the following levels according to [4]:

- Lexical: One or more components of an MWE are not part of the conventional English lexicon, e.g. *ad hoc*.
- Syntactic: The syntax of the MWE is not derived from that of its components, e.g. *by and large* is used as an adverb instead of a preposition (*by*) and adjective (*large*).
- Semantic: The meaning of the MWE cannot be explicitly derived from its lexemes, e.g. *kick the bucket* refers to the action of *dying* instead of the literal act of kicking a container.
- Pragmatic: A MWE being associated with a fixed set of scenarios or a particular context. These are often ambiguous with literal translations, e.g. *Good morning* as a greeting or as an actually "*good*" (or pleasant) *morning*.
- Statistical: Refers to when a particular combination of lexemes occurs with high-frequency, relative to the use of its components.

The deviation mentioned causes, as it can be observed from the examples, that the conjunct use of the lexemes creates a new meaning to the expression that is not obviously extracted from the individual meanings.

2.2 Types of Multiword Expressions

There are several categories of MWEs based on its component tokens, and these vary across languages. For this report we are focusing on defining only English MWEs, but keep in mind that other categories of MWEs exists for other languages.

The three main categories of MWEs are: **Nominal**, **Verbal**, and **Prepositional** [4]. Nominal MWEs are the most common types based on token and type frequency [4]. *Noun Compounds* (NCs) are the primary type of Nominal MWEs, and these refer to the combination of **modifier** and **head** nouns, such as *golf club* and *computer science department*, in which "golf" and "computer science" are modifier nouns while "club" and "department" are the head nouns [4]. Other types of Nominal MWEs allow the head

noun to be deverbal (e.g. *investor hesitation*), while also allow the modifiers to be verbs or adjectives (e.g. *connecting flight* and *open secret* respectively) [4]. The next category to be discussed is that of Preposition MWEs, which consists of *Determinerless-Prepositional Phrases* (PP-Ds) and *Complex Prepositions* [4]. The former, PP-Ds, are made up of a preposition and a singular noun without a determiner and are highly diverse (e.g. *on [table] top, by car/bus/foot/…,* and *at [eye] level*), the later refers to complex prepositions (e.g. *on top of, in addition to*), and other forms of complex markers [4]. Lastly, and the focus of our investigation, are the Verbal MWEs, which consist of 4 major types: *Verb-Particle Constructions, Prepositional Verbs, Light-Verb Constructions,* and *Verb-Noun Idiomatic Combinations* [4]. From this last group, we are particularly interested in the *Verb-Noun Idiomatic Combinations* (VNICs), which are composed of a verb and noun in direct object position and are at least semantically idiomatic [4]. We explain the relevance of this particular type of MWE in the following sections.

2.3 Verb-Noun Idiomatic Combinations

Verb-Noun Idiomatic Combinations (VNICs) are often categorized into two groups, based on their **semantic decomposability** [1] [4]. **Non-decomposable** instances of VNCs are not subject to syntactic variability by modification of the idiomatic phrase (i.e. passitivization); instead the only types of lexical variation they can undergo are inflection (e.g. *kick the bucket* → *kicked the bucket*) and variation in reflexive form (*wet himself* → *wet herself*) [1] [4]. On the other side, **decomposable** idioms allow to extract the meaning by analysing the semantically linked parts (e.g. *spill the beans* => *spill* → *reveal* + *the beans* → *secrets*) and are syntactically flexible to some degree (e.g. *spill the beans* →*the beans were spilled*), however, variation is highly unpredictable [1] [4]. This lexical and syntactical variation deters research from merely utilizing a "words-with-spaces" approach for VNIC analysis, but stablishes this variability as a corner stone for understanding the idiomatic use of these phrases [1] [3].

Recent research of non-literal usage of MWEs focuses on VNIC analysis because of their crosslingual occurrence, and have high lexical and semantic variability in comparison to other kinds of MWEs [1] [3] [4]. First, as mentioned previously, is the observation that most idiomatic VNICs exhibit **lexico-syntactic fixedness** [2] [3]; i.e. the phrase *see stars* often presents idiomatic meaning when the verb has active voice, the determiner is null, and the noun is in plural form, as in *see stars* or *seeing stars*; while usages with a determiner (*see the stars*), singular noun form (*see a star),* or passive voice (*stars where seen*) often have literal interpretation [2].

Lexical-fixedness of idiomatic phrases means that the substitution of a near synonym for a constituent does not preserve the idiomatic meaning of the expression [3] (i.e. *see stars* and *observe stars*). Even if some idioms allow lexical variations which generate closely related meanings, these are usually highly unpredictable substitutions that can't be considered as a rule [3].

Syntactic-fixedness means that many idiomatic VNCs cannot undergo syntactic variations while retaining their idiomatic interpretation (i.e. the punch let him *seeing stars* / *seeing the star*); however, it is relevant to note that idiomatic VNCs differ with respect to their degree of tolerance to semantic operations (**syntactic flexibility**) [3].

Following on the concept of lexico-syntactic fixedness, a corpus-based study by [8] showed that "in contrast to nonidiomatic combinations of words-idioms have strongly preferred canonical form, but at the same time the occurrence of idiom variation is too common to be ignored" [8]. This sounds redundant, as it says that idiomatic VNC identification must be on the lookout for any form of VNC variation since they can all be idiomatic. However, it also establishes the **canonical form**, which is the standard form of a VNC, also known as its "preferred usage" [3]. Subsequent research works on the assumption that idiomatic VNCs are more likely to appear on canonical form than non-idiomatic phrases [3].

Phrasal idioms have also been found to involve a certain degree of semantic idiosyncrasy, which means that the idiom is hard to determine without special context or previous exposure even if the meaning of the component words is clear [3] [6]. Also, although it is traditionally believed that idioms are completely non-compositional, linguists and psycholinguists claim that they show some degree of semantic compositionality [3]. This suggests that many idioms have internal semantic structure, without ignoring the fact that they are non-compositional in a traditional sense, which opens the field for the introduction of terms such as **semantic decomposability** and/or **semantic analyzability** [3]. To say that an idiomatic VNC is semantically analysable means that the constituents contribute by their independent meanings to the idiomatic interpretation; so, the more semantically analyzable an idiom is, the easier it is to interpret the idiomatic meaning from its constituents [3].

3 Literature Review

Much research on MWE identification focuses on the task of detecting specific kinds on MWEs, such as English VNCs [2] [3] [9], while other authors focus on multilingual detection of MWEs [10]. Another great distinction within idiomatic use of MWEs is between the tasks of *idiom type* and *idiom token* classification; while *idiom type* classification is the task of identifying expression with possible idiomatic interpretations, *idiom token* classification focuses on distinguishing between idiomatic and literal usages of potentially idiomatic phrases [6].
I this research, we propose to focus on the task of detecting idiomatic usage of token-level (*idiom token* classification) English VNCs. What distinguishes idiomatic and literal VNCs is the fact that an idiom has a different meaning than the resulting from the simple composition of the meaning of its component words [3]. To detect if a VNC presents idiomatic usage previous research has made use of supervised and unsupervised methods for learning underlying patters in idiomatic VNC formation, making use of the sentence context, the lexical and syntactic fixedness of the phrase in the corpora, and feature extraction with Word2Vec and Sent2Vec methods.

3.1 Unsupervised Methods

Unsupervised models' approach to detect idiomatic phrases uses the assumptions mentioned in Section 2.3.

Pattern No.	Pattern Signature			Example
1	v_{act}	det:NULL	n_{sg}	*give money*
2	v_{act}	det:*a/an*	n_{sg}	*give a book*
3	v_{act}	det:*the*	n_{sg}	*give the book*
4	v_{act}	det:DEM	n_{sg}	*give this book*
5	v_{act}	det:POSS	n_{sg}	*give my book*
6	v_{act}	det:NULL	n_{pl}	*give books*
7	v_{act}	det:*the*	n_{pl}	*give the books*
8	v_{act}	det:DEM	n_{pl}	*give those books*
9	v_{act}	det:POSS	n_{pl}	*give my books*
10	v_{act}	det:OTHER	$n_{sg,pl}$	*give many books*
11	v_{pass}	det:ANY	$n_{sg,pl}$	*a/the/this/my book/books was/were given*

Table 1 - VNC Patterns [3]. A pattern signature is composed of a verb v in active (v_{act}) or passive (v_{pass}); a determiner (det) that can be NULL, indefinite (*a/an*), definite (*the*), demonstrative (DEM), or possessive (POSS); and a noun n that can be singular (n_{sg}) or plural (n_{pl})

An approach by [3] calculates the lexical and syntactic fixedness in numerical values to generate a degree of fixedness, which is useful since idiomatic use of VNCs is believed to be both lexically and semantically more fixed than literal verb+noun combinations. To measure lexical fixedness of a pair, [3] calculates its association strength for the target pair and its variants using Pointwise Mutual Information (Equation 1). All verb-noun combination pairs are those which correspond to one of the patterns in Table 1.

$$PMI(v_r, n_t) = \log \frac{P(v_r, n_t)}{P(v_r)P(n_t)} = \log \frac{N_{v+n} f(v_r, n_t)}{f(v_r,*) f(*, n_t)}$$

Equation 1 - PMI for Verb+Noun Combinations in [3]

Where $\langle v_r, n_t \rangle \in \{\langle v, n \rangle\} \cup S_{sim}(v, n)$, in which $S_{sim}(v, n) = \{\langle v_i, n \rangle | 1 \leq i \leq K_v\} \cup \{\langle v, n_j \rangle | 1 \leq j \leq K_n\}$, being K_v a parameter for the number of closest verbs to target v and K_n the number of closest nouns to target n [3]; N_{v+n} is the total number of verb-object pairs in the corpus; $f(v_r, n_t)$, $f(v_r,*)$, and $f(*, n_t)$ are the frequency counts of the target verb+noun pair, the target verb with any other noun and the target noun with any other verb respectively.

$$Fixedness_{lex}(v, n) = \frac{PMI(v, n) - \overline{PMI}}{s}$$

Equation 2 - Degree of Lexical Fixedness of Verb-Noun Combination in [3]

Equation 2 calculates a degree of lexical-fixedness for a verb-noun combination under the assumption that the target pair is lexically fixed to the extent that its PMI deviates from the average PMI of its variants [3]. The higher the degree, the more lexically fixed the pair is, thus $Fixedness_{lex}(v, n) \in [-\infty, +\infty]$. In Equation 2, $\overline{PMI}$ and s are the mean and standard deviation of the following sample: $\{PMI(v_r, n_t) | \langle v_r, n_t \rangle \in \{\langle v, n \rangle\} \cup S_{sim(v,n)}\}$.

The author then proceeds to explain the process of calculating the Syntactic Fixedness, under the assumption that idiomatic VNCs appear in more restricted syntactic forms [3]. To quantify this value, they first identify relevant syntactic patters to distinguish idiomatic from literal usage, to then translate the frequency distribution of the target pair in the identified patterns to measure syntactic fixedness.
The identified syntactic patterns were:

- Passivization: Idiomatic VNCs often do not undergo passivization due to the non-referential status of the noun constituent in most idiomatic verb-noun pairs.
- Determiner type: There's a strong correlation between the flexibility of the determiner preceding the noun in a VNC and the overall flexibility of the phrase. Idiomatic VNCs are expected to appear with one type of determiner.
- Pluralization: Even if the verb constituent of idiomatic VNCs is morphologically flexible, the non-referential noun constituent of the pair is expected to mainly appear in just one of the singular or plural forms.

The step of *devising a statistical measure* that quantifies the degree of syntactic fixedness using the proposed set of patterns proposes a measure that compares the syntactic behaviour of the target pair with that of a "typical" verb-noun pair. The syntactic behaviour of a typical pair is defined as the prior probability distribution over the selected patterns (Equation 3), where V is the set of all instances of transitive verbs in the corpus, and N is the set of all instances of nouns as direct objects of the verb.

$$P(pt) = \frac{\sum_{v_i \in V} \sum_{n_j \in N} f(v_i, n_j, pt)}{\sum_{v_i \in V} \sum_{n_j \in N} \sum_{pt_k \in P} f(v_i, n_j, pt_k)} = \frac{f(*,*,pt)}{f(*,*,*)}$$

Equation 3 - Syntactic Behaviour of Typical Verb-Noun Pair in [3]

For the target pairs $\langle v, n \rangle$, the syntactic behaviour is defined as the posterior probability distribution over the patterns given the pair, as shown in Equation 4.

$$P(pt|v,n) = \frac{f(v,n,pt)}{\sum_{pt_k \in P} f(v,n,pt_k)} = \frac{f(v,n,pt)}{f(v,n,*)}$$

Equation 4 - Syntactic Behaviour of Target Verb-Noun Pair in [3]

Using these two equations, the degree of syntactic fixedness for a target verb-noun pair is estimated the divergence of its syntactic behaviour from the typical syntactic behaviour, which is formulated in Equation 5 using Kullback Leibler (KL-) divergence [11]. Thus $Fixedness_{syn}(v,n) \in [0, +\infty]$.

$$Fixedness_{syn}(v,n) = D(P(pt|v,n)||P(pt)) = \sum_{pt_k \in P} P(pt_k|v,n) \log \frac{P(pt_k|v,n)}{P(pt_k)}$$

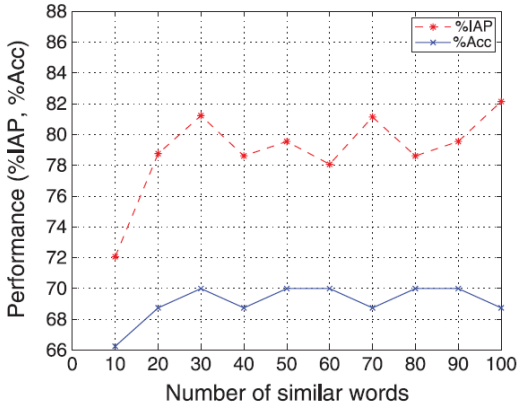
Equation 5 - Degree of Syntactic Fixedness for a Target Verb-Noun pair in [3]

The authors of [3] hypothesize that idiomatic VNCs are both lexically and syntactically more fixed than literal verb-noun combinations, thus they propose Equation 6 to measure overall fixedness of a given pair, rescaling the syntactic and lexical fixedness degrees under the range [0,1], so the overall fixedness falls in the range $Fixedness_{overall}(v,n) \in [0,1]$.

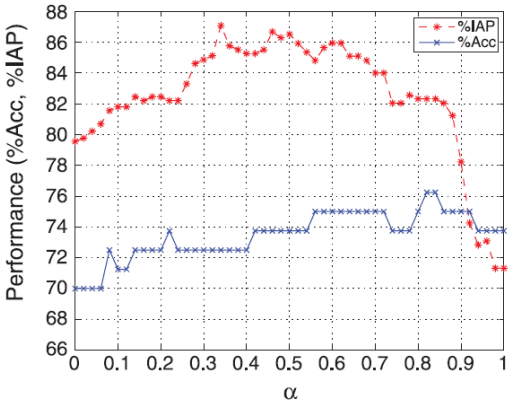
$$Fixedness_{overall}(v,n) = \alpha Fixedness_{syn}(v,n) + (1-\alpha) Fixedness_{lex}(v,n)$$

Equation 6 - Overall Fixedness for Target Verb-Noun pair in [3]

To measure the performance, the median score of the fixedness was determined as the threshold for separating idiomatic from literal VNCs, being pairs with an overall fixedness degree over the threshold classified as idiomatic. The results of accuracy, relative error rate reduction (ERR), the precision-recall curves, and the interpolated three-point average precision (IAP) are shown in Figure 1.



(a) Performance of $Fixedness_{lex}$ as a function of K.



(b) Performance of $Fixedness_{overall}$ as a function of α.

Figure 1 - %IAP and %Acc of Fixedness_lex and Fixedness_overall over development data [3]

Canonical Forms

Based on the assumption that VNICs are syntactically fixed, it is suggested that they will have a small set of Canonical Forms ("preferred forms") [3] [8]. An unsupervised

method by [3] aims to find this set of Canonical Forms C of a target pair $\langle v, n \rangle$ with the following equation:

$$C(v,n) = \{pt_k \in P | z(v,n,pt_k) > T_z\}$$

Equation 7 - Algorithm to find set of Canonical Forms by [3]

where P is the set of possible verb/noun patterns defined in Table 1, $z(v,n,pt_k)$ is the static z-score in Equation 8, and T_z is a predefined threshold.

$$z(v,n,pt_k) = \frac{f(v,n,pt_k) - \overline{f}}{s}$$

Equation 8 - z-score for Pattern counts by [3]

where $f(v,n,pt_k)$ is the frequency of a given verb/noun pair appearing in a particular pattern (pt_k), $\overline{f}$ is the sample mean, and s the sample standard deviation.

3.2 Further Unsupervised Methods

Fazly et al. (2009) success on discriminating idiomatic use of VNCs with their metrics plus the high performance of SVMs with linear kernels (more on this in the following sections) have demonstrated that word vector representations of target sentences can be linearly separated given a correct set of features. Previous research on clustering methods for idiomatic vs literal use discrimination also works under the assumption that a target sentence is attracted to a collection of *seed sentences* (training examples) to which it shows the highest similarity; two sentences are similar if they contain similar words and two words are similar if they are contained in similar sentences [12]. This motivates the use of clustering algorithms that can separate idiomatic phrases from their literal use. In order to backup this claim, we propose the use of the **k-Means** algorithm to find these clusters on diverse sentence embbedings based on previous research using this tool by [9].

k-Means Clustering

The **k-Means** algorithm places all the example feature vectors in the sample space and starts by initializing *k* number of random cluster centroids (hence the name) [13]. Examples are then assigned to the closest centroid based on a distance function and after all examples have been assigned, the centroids are repositioned to the mean position of their currently tagged examples, repeating the process until centroid convergence [13]. This can be better observed in Figure 2, where $\|x^t - m_i\|$ can be any distance function:

$$Initialize\ m_i, i =, \dots, k$$
$$Repeat:$$
$$\quad For\ all\ x^t \in X:$$
$$\quad\quad b_i^t \leftarrow \begin{cases} 1, if\ \|x^t - m_i\| = min_j \|x^t - m_j\| \\ 0, otherwise \end{cases}$$
$$\quad For\ all\ m_i, i = 1, \dots, k:$$
$$\quad\quad m_i \leftarrow \frac{\sum_t b_i^t x^t}{\sum_t b_i^t}$$
$$Until\ m_i\ converge$$

Figure 2 - k-Means Algorithm from [13]

k-Means is capable of grouping different sets of data in the problem vector space. The proposal of implementing it for this problem comes from the success of Linear Kernel SVMs, which suggest linear separation of the task vector space. **K-Means** was also chosen over more complex clustering algorithms (e.g. DBSCAN [14]) due to its simplicity of implementation and its capacity for generalizing the classification of new examples. However, if time constraints allow it, we could expand research for different unsupervised clustering algorithms.

Cosine Similarity

Some research on the matter has aimed to use similarity-based models: between the target VNIC and the context(s) in which it is being used; between the VNIC the definition used to describe the non-literal meaning of the idiom; between the contexts in which the VNIC is used [15] [16].

Particularly, we are interested in previous research that makes use of the word embeddings under the assumption that they capture the 'meaning' of the sentences' context and their respective VNICs [15]. Research focused on identifying paraphrasing of idiomatic targets ('Cloud Nine' vs 'Seventh Heaven') determines the semantic similarity between two idioms as the cosine similarity between the averaged word embeddings of the tokens used to define each of the target idioms [15]. We then assume that the cosine similarity between sentence embeddings is a good determined about the semantic similarity between two sentences.

To avoid any sort of confusion, we refer to the following as the formula for cosine similarity between two vectors v_1 and v_2:

$$cosSim(v_1, v_2) = \frac{v_1 \cdot v_2}{|v_1| * |v_2|}$$

Equation 9 - Cosine Similarity

3.3 Supervised Methods

Supervised model training for idiomatic phrase detection focuses on identifying if a given excerpt of a sentence is of idiomatic or literal meaning. This approach usually tackles token-level identification of VNCs as a supervised binary classification problem, classifying the use of a VNC as idiomatic of literal [2] [6].

Recent research makes use of classifier models such as k-Nearest Neighbours (k-NNs) and Support Vector Machines (SVMs) with linear and polynomial kernels, since these algorithms have been proven to work for binary classification problems [2] [6]. However, the pre-processing of known VNCs and feature creation is the task in which most research focuses on. In an attempt to exploit the knowledge on lexical and syntactic patterns presented in Section 3.1, researchers have made use of unsupervised feature encoders to train the classifiers [2] [9] [6].

One approach tries to train a Linear SVM Classifier with data obtained by using **Word Embeddings** from Word2Vec [17]. This implementation proposes the creation of two vectors that contain both the representation of the VNC and its context, called $\vec{e}$ and $\vec{c}$ respectively [9]. The former is created by averaging the word embedding vectors of the lemmatized component words of the VNC, while the latter is the averaging of two other vectors that represent the context of the verb and noun components [9]. Once $\vec{e}$ and $\vec{c}$ are obtained they are subtracted into a feature vector, which is then appended a Boolean feature that determines if the VNC occurs in its Canonical Form (CForm) or not, as described in [3]. The resultant feature vector is then used to train an SVM with linear kernel. The results of this approach are shown on Table 2, comparing them to the models presented in [3].

Window	Dimensions	Dev	Test
1	50	87.3	85.9
	100	**88.2**	85.5
	300	86.3	**88.3**
2	50	86.4	84.2
	100	86.7	84.2
	300	86.5	86.7
5	50	86.0	83.4
	100	85.9	84.2
	300	87.3	85.7
8	50	85.5	84.3
	100	85.6	85.9
	300	85.8	86.3
Baseline		62.1	61.9
Fazly et al. (2009) CFORM		72.3	73.7
Fazly et al. (2009) Supervised		80.1	82.7

Table 2 - Accuracy Score for supervised word2vec approach by [9]

Other such approach is that of **Skip-Thoughts Vectors** (Sent2Vec) [18], used originally by [6] and then used as a base of comparison by [2]. This model uses the continuity of text from books to train an encoder-decoder model that aims to reconstruct the surrounding sentences of an encoded passage, so sentences with similar semantic and syntactic properties are mapped to similar vector representations [18]. This results in an encoder that can produce highly generic sentence representations [18]. Sent2Vec was first used for idiom detection on the assumption that, in a real-world application, target phrases won't have access to a surrounding context [6]; which motivated the exploration of distributed compositional semantic models to produce reliable estimates of idiom token classification [6]. Utility found in the Sent2Vec model is that it is possible to infer properties of the surrounding context only from the input sentence [6] [18], which allows the classifier to learn lexical and syntactic patterns without complex methods. Salton et al. uses the resulting encodings to train three SVM classifiers with the VNC-Tokens Dataset [6] [19]: Linear Kernel with C=1.0, Polynomial Kernel of degree = 2 and C = 1000, and Linear Kernel trained using Stochastic Gradient Descent with a learning rate of 0.0001. Results on the classifiers (Table 3) show an improvement on the baseline set by the authors, which used entire context extracted from several paragraphs.

Expressions	Linear-SVM-GE			Grid-SVM-GE			SGD-SVM-GE		
	P.	R.	F1	P.	R.	F1	P.	R.	F1
BlowTop	0.91	0.96	0.94	0.91	0.93	0.94	0.80	0.98	0.88
BlowTrumpet	0.98	0.88	0.93	0.98	0.88	0.93	0.89	0.93	0.90
BlowWhistle*	0.84	0.67	0.75	0.84	0.68	0.75	0.67	0.59	0.63
CutFigure	0.91	0.85	0.88	0.89	0.85	0.87	0.86	0.85	0.86
FindFoot	0.96	0.93	0.94	0.97	0.93	0.95	0.85	0.90	0.87
GetNod	0.98	0.91	0.95	0.98	0.91	0.95	0.91	0.91	0.91
GetSack	0.87	0.89	0.88	0.86	0.88	0.87	0.81	0.89	0.84
GetWind	0.86	0.82	0.84	0.92	0.85	0.88	0.69	0.81	0.75
HaveWord	0.99	0.89	0.94	0.99	0.89	0.94	0.95	0.91	0.93
HitRoad	0.86	0.98	0.92	0.89	0.98	0.93	0.83	0.98	0.90
HitRoof	0.88	0.88	0.88	0.92	0.88	0.90	0.80	0.83	0.82
HitWall	0.74	0.58	0.65	0.74	0.58	0.65	0.74	0.45	0.56
HoldFire	1.00	0.63	0.77	1.00	0.63	0.77	0.82	0.67	0.74
KickHeel	0.92	0.96	0.94	0.92	0.99	0.95	0.89	0.92	0.91
LoseHead*	0.78	0.66	0.72	0.75	0.64	0.69	0.75	0.67	0.71
LoseThread	1.00	0.88	0.93	1.00	0.86	0.92	0.81	0.85	0.83
MakeFace	0.70	0.83	0.76	0.69	0.76	0.72	0.62	0.81	0.70
MakeHay	0.81	0.78	0.79	0.81	0.84	0.82	0.73	0.76	0.75
MakeHit	0.10	0.54	0.70	0.10	0.54	0.70	0.85	0.55	0.67
MakeMark	0.99	0.92	0.95	0.98	0.91	0.94	0.93	0.93	0.93
MakePile	0.84	0.67	0.74	0.84	0.70	0.76	0.74	0.70	0.72
MakeScene*	0.92	0.84	0.88	0.92	0.81	0.86	0.78	0.81	0.79
PullLeg	0.79	0.71	0.75	0.82	0.72	0.77	0.75	0.70	0.72
PullPlug	0.91	0.91	0.91	0.91	0.91	0.91	0.90	0.92	0.91
PullPunch	0.85	0.87	0.86	0.87	0.87	0.87	0.70	0.85	0.77
PullWeight	1.00	0.96	0.98	1.00	0.96	0.98	0.89	0.93	0.93
SeeStar	0.17	0.13	0.15	0.17	0.13	0.15	0.17	0.17	0.17
TakeHeart*	0.94	0.79	0.86	0.94	0.80	0.86	0.86	0.80	0.83
Total	0.84	0.80	0.83	0.84	0.80	0.83	0.79	0.79	0.78

Table 3 - Precision (P.), Recall (R.), and F1-Score (F1) results on Generic Classifiers by [6]

The last reviewed research for the supervised learning methods is that developed by King and Cook, which also used a Linear SVM kernel but experiments with three different feature encodings for the VNCs and their context [2]. First, they use Word2Vec's **Skip-Gram** model [17], similarly to the approach taken by [9]; however, instead of encoding the VNC and context in different vectors and then subtracting them, their implementation averages the normalized word embeddings for each word in the sentences containing a target VNC [2]. Secondly, they use the **Siamese CBOW** model [5] since it "learns word embeddings that are better able to represent a sentence through averaging that conventional word embeddings such as skip-gram or CBOW" [2]; the word embeddings produced for this model for a target sentence are averaged as in the Skip-Gram implementation. Lastly, they replicate the skip-thoughts model approach taken by [6] to use as a strong baseline for comparison. As an extra feature, they append the CForm [3] Boolean feature as described previously with the approach taken by [9]. The accuracy score results for the different word embeddings methods used with and without the added CForm are shown on Table 4.

Model	DEV		TEST	
	−CF	+CF	−CF	+CF
CForm	-	0.721	-	0.749
Word2vec	**0.830**	**0.854**	**0.804**	**0.852**
Siamese CBOW	0.763	0.774	0.717	0.779
Skip-thoughts	0.803	0.827	0.786	0.842

Table 4 - Accuracy Score for supervised word2vec, siamese cbow and skip-thoughts approaches by [2]

3.4 Further Word Embedding Models

Aside from Word2Vec, Siamese CBOW, and Skip-Thoughts models for word embeddings used in previous idiomatic VNC detection research, we expand this paper to include the recent **Embeddings from Language Models** (ELMo) embeddings to evaluate performance [20].

3.5 ELMo

$$ELMo_k^{task} = E(R_k; \theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} h_{kj}^{LM}, j = 0, .., L$$

Equation 10 - ELMo Layer Computation by [20]

ELMo word representations are computed on top of two-layer Bidirectional Language Models (biLM) as a linear function of the internal network states, represented as h_{kj}^{LM} in Equation 10, where h_{k0}^{LM} is the token layer [20]. Each biLM consist of a Forward and a Backward Linear Model $h_{kj}^{LM} = \left[\overrightarrow{h_{kJ}^{LM}}; \overleftarrow{h_{kJ}^{LM}}\right]$, each one computing a context-independent representation of the target token, repeated L times in Equation 10 [20]. The output of these biLMs goes through the ELMo layer, which passes its enhanced representations to the task in hand. s_j^{task} are softmax-normalized weights and the scalar parameter γ^{task} allows the task model to scale the entire ELMo vector [20]. According to [20], the representations provided by the ELMo Layer over the word embeddings outputted by the biLSTMs enrich the models, since higher-level LSTM states capture context-dependent aspects of word meaning, while lower-level states cover aspects of syntax, which are two highly relevant features for our task according to previous research on idiom detection [3]. The representations provided by ELMo layer over word embeddings have been proven to increase performance of methods that use biLSTMs on its original, as it can be seen in Table 5. This motivates us to add ELMo to this task in the exploratory section of development, to compare with the previous described methods as a baseline.

Task	Previous SOTA		Our baseline	ELMo + baseline	Increase (absolute/ relative)
SQuAD	Liu et al. (2017)	84.4	81.1	85.8	4.7 / 24.9%
SNLI	Chen et al. (2017)	88.6	88.0	88.7 ± 0.17	0.7 / 5.8%
SRL	He et al. (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coref	Lee et al. (2017)	67.2	67.2	70.4	3.2 / 9.8%
NER	Peters et al. (2017)	91.93 ± 0.19	90.15	92.22 ± 0.10	2.06 / 21%
SST-5	McCann et al. (2017)	53.7	51.4	54.7 ± 0.5	3.3 / 6.8%

Table 5 - Test set comparison of ELMo enhanced neural models with state-of-the-art model baselines across six benchmark NLP tasks by [20]

4 Methods

This section aims to describe the implementation of the previously described algorithms for the purposes of this dissertation.

4.1 Fixedness Metrics

These metrics were implemented as described previously in Section 3.1, based on previous work aiming on unsupervised detection of VNCs [3].

Verb-Noun Combination Pattern Counts

We consider a correct VNC Pattern count if it follows one of the signatures described in Table 1.
To count the instances of VNC Patterns in a Corpora, we implemented a method that takes two lists, one containing the lemmatized tokens of a given sentence while the other contains the respective Part-of-Speech (PoS) tags. The algorithm returns a list of all the VNCs that occur within a given window along with their respective Pattern Number.

Since this project works using the BNC-XML Corpora [21] (more details in Section 5.1), we take into account the C5 Tagset for the PoS Tags. With this in mind, here are the PoS tags that correspond for each element described in Table 1, along with extra tokens used for determining the validity of patterns.

Patterns 1 to 10 are implemented exactly as described in Table 1, since their appearance in text is straight-forward. However, since Pattern 11's signature describes VNCs in passive-form, we use the following pattern to our purposes:

$$det: ANY \quad n_{sl}|s_{pl} \quad BE \quad v_{pass}$$

Where *BE* is any form of the verb *to be*.

One the found patterns are returned, they are stored in a dictionary, in which the key is a tuple of the Verb and the Noun, and the value is a list in which the index corresponds to one of [3]'s patterns. Index 0 corresponds to its frequency. This dictionary is stored in disk for later use.

When extracting the VNC Pattern Counts, the only hyperparameter is that of **Window Size**. The standard value we use for out experiments is 7, but more on that in Section 5.3.

Pattern Element	*C5 Tags*	*Additional Comments*
DET	AT0, DT0, DTQ, DT0-CJT	This element also takes into account if the token is either *A/AN* or *THE*, since both fall under the AT0 tag but correspond to different signatures.
POSS	DPS	
v_{act}	VBB, VBD, VBG, VBI, VBN, VBZ, VDB, VDD, VDG, VDI, VDN, VDZ, VHB, VHD, VHG, VHI, VHN, VHZ, VM0, VVB, VVD, VVG, VVI, VVN, VVZ, VVB-NN1, VVD-AJ0, VVD-VVN, VVG-AJ0, VVN-AJ0, VVN-VVD, VVZ-NN2	
v_{pass}	VBN, VDN, VHN, VVN, VVN-AJ0, VVN-VVD	
n_{sg}	NN0, NN1, NP0, NN1-AJ0, NN1-NP0, NN1-VVB, NN1-VVG, NP0-NN1, VVB-NN1, AJ0-NN1	
n_{pl}	NN0, NN2, NN2-VVZ, VVZ-NN2	
BE	VBB, VBD, VBG, VBI, VBN, VBZ	Used for Pattern 11, since VNC in passive voice must include a form of the BE verb between them.
PUNC	PUN	Punctuation tokens (such as '.') will stop a VNC from forming. Tokens '-' and ',' are excluded when using this PoS, since they are valid occurrences.

Table 6 - C5 Tag Equivalences for VNC Pattern Elements.

Pointwise Mutual Information

Our Pointwise Mutual Information (PMI) implementation is that described in Equation 1.

Recalling the mentioned equation, we take into account the following values for the different elements of PMI for VNC v_r, n_t:

- N_{v+n}: Total number of captured VNCs (in this case, the length of the dictionary described earlier in this section).
- $f(v_r, n_t)$: Frequency of the target VNC (in this case, the position 0 of the stored list).
- $f(v_r, *)$: Total frequency of v_r in all extracted patterns (in this case, the sum of all pattern frequencies in which v_r appears).
- $f(*, n_t)$: Total frequency of n_t in all extracted patterns (in this case, the sum of all pattern frequencies in which n_t appears).

We use base 2 for all logarithmic calculations.

Lexical Fixedness

The implementation of the Lexical Fixedness metric follows the description observed in Equation 2 [3].
Its implementation is pretty straight-forward, since it only requires to calculate the PMI of the target VNC, and the mean and standard deviation of the PMIs of those VNCs that fall in the following sample: $\{PMI(v_r, n_t) | \langle v_r, n_t \rangle \in \{\langle v, n \rangle\} \cup S_{sim(v,n)}\}$, in which $S_{sim(v,n)} = \{\langle v_i, n \rangle | 1 \leq i \leq K_v\} \cup \{\langle v, n_j \rangle | 1 \leq j \leq K_n\}$, being K_v and K_n the number of the most similar verbs and nouns to those in the VNC respectively.

To get the K_v most similar verbs and the K_n most similar nouns, we use one of two options: Lin's Thesaurus following implementation by [3] or a subset of most similar tokens using WordNet expanded by the most similar elements found by Word2Vec. The former is an already tested implementation for obtaining Lexical Fixedness, while the latter may provide a set of more accurate similar tokens since the Word2Vec implementation was trained with the target corpora (more on that in Section 4.3) curated by hand-reviewed similar tokens contained in WordNet.

Our described implementation is affected by the following hyperparameters:

- Word Window in Pattern Counts
- K_v
- K_n
- Use Lin's Thesaurus or WordNet+Word2Vec for similar Verb/Nouns

Syntactic Fixedness

Our implementation of Syntactical Fixedness follows Equation 3, Equation 4, and Equation 5 by using the Pattern Counts obtained.

Equation 3, which is the Maximum Likelihood Estimate (MLE), for a given Pattern is implemented by diving the total counts of a given pattern across all observed VNCs by the total frequency of all patterns across all VNCs. In our data structure for VNC Pattern Counts, this is calculated by:

- $f(*,*,pt)$: Accumulated frequency for a given pattern, that is a given index in the stored list, across all keys in our VNC dictionary.
- $f(*,*,*)$: Accumulated frequency of all appearances of the stored keys, that is the sum of all counts stored in index 0.

Equation 4, which is the conditional probability of a pattern occurring given a VNC, is implemented by dividing the frequency a VNC appeared in a given pattern by its total frequency. Using our stored VNCs, this was calculated by:

- $f(v,n,pt)$: Index 1-11 from the list of pattern counts given a key (VNC).
- $f(v,n,*)$: Index 0 from the list of pattern counts given a key (VNC).

Equation 5 is the Kullback Leibler (KL-) divergence between the probability of a pattern occurring given the target VNC and the probability of it occurring at all. For a given VNC, we take the prior two calculated equations for all 11 Patterns and store the results.

This equation has no hyperparameters.

Overall Fixedness

The Overall Fixedness of a target VNC is obtained using Equation 6, which calculates a weighted mean between the Lexical Fixedness and Syntactic Fixedness degrees for a target VNC with a factor $\propto \in [0,1]$, being the larger the value, the higher the weight of Syntactical Fixedness at viceversa [3].

As described, this equation requires the following hyperparameters:

- $\propto$

As well, since it is affected by the Lexical Fixedness measure, it also requires:

- Word Window in Pattern Counts
- K_v
- K_n
- Use Lin's Thesaurus or WordNet+Word2Vec for similar Verb/Nouns

Canonical Forms

To find the Canonical Forms of the VNCs in our project, we implement Equation 7 and Equation 8 [3].
For Equation 8, we use the stored frequencies from out Verb-Noun Pattern counts:

- $f(v, n, pt_k)$: Index 1-11 of the list of pattern counts given a key (VNC).
- $\overline{f}$: Mean value stored in indexes 1-11 from the list of pattern counts given a key.
- s: Standard Deviation of values stored in indexes 1-11 from the list of pattern counts given a key.

For Equation 7, we store the set of patterns for those which z-score surpasses the defined threshold T_z.

Our described implementation is affected by the following hyperparameters:

- Word Window in Pattern Counts
- T_z

4.2 Potential VNICs Extraction

We propose a method for extracting potentially idiomatic VNICs based on the fixedness between the verb-noun components [3]. Our aim is to extract all the best VNIC candidates that appear in a given corpora based on the assumption that the higher the scores obtained in the metrics described in Section 3.1, the more likely a VNC has some sort of idiomatic usage [3].

We perform this action by applying all the formulas implemented in the previous section (4.1) to the list of found VNCs in the test.

Frequency Considerations

While extracting the candidate VNICs we discovered that the top scoring VNCs were those of very specific Verb-Noun pairs with overall low frequency, which resulted in a poor selection of candidates that reflected phrases with small, contextual use, on the corpora which contained them.
To fix this, we set a threshold of **150** minimum instances for a candidate VNC pair to be considered.

Metrics

To define the top VNICs candidates we decided to use the PMI, Syntactical Fixedness, Lexical Fixedness, and Overall Fixedness scores. In order to achieve this, we calculated these scores for all found VNCs in the Corpora.
To define the most relevant candidates, we generated four lists that rank in a descending order the scores of the VNCs:

- PotentialVNICs_PMI.csv
- PotentialVNICs_LEX.csv
- PotentialVNICs_SYN.csv
- PotentialVNICs_OVA.csv

We use the Top 20 scored VNCs in each of the ranked lists as our top VNICs candidates.

Instance Extraction

Having obtained the Top 20 scored VNICs, we proceed to generate a dataset similar to VNC-Tokens Dataset [19] called 'VNC-Tokens_candidates'.
In order to achieve this, we search for all the instances in which the top scoring VNICs candidates appear within a predefined Word Window (with the same length as that used for VNC extraction). We store the found instances in the same format as the one used in the VNC-Tokens Dataset, containing first the classification ('Q' in all instances since we have no way of determining if usage was Literal 'L' or Idiomatic 'I'), followed by the Verb/Noun combinations concatenated with the '_' token, then the location of the file in the BNC XML Corpora, and lastly the sentence id.

4.3 Word Embeddings

Word Embeddings are used for experiments following research by [2] [6] [9]. For our purposes, we generate word embeddings for the target sentence which contains usage of VNICs and the substring that possess the VNIC as it appears in the target sentence. I.e., for target sentence containing the VNC <kick, bucket>: "After all those years of smoking, he finally **kicked the bucket**.", we generate an embedding for the whole sentence and an embedding for the VNIC "*kicked the bucket*" verbatim and lemmatized.

In this section, we will explain the implementation, training, and embedding generation for the selected embedding models.

Word2Vec

Implementation of Word2Vec [17] model was made with the publicly available library **gensim**. Training was performed using the available BNC XML Corpora, with the parameters used by [2]:

- Window Size: ±8
- Vector Dimension: 300
- Epochs: 5

Since this model does not offer sentence embeddings, we follow [2]'s implementation by averaging the embeddings of all tokens in the target sentence, including stopwords. However, normalization to the tokens embeddings to have unit length was not performed.

Siamese CBOW

The Siamese CBOW [5] model was implemented using Tom Kenter's implementation of the algorithm[1]. We used a pre-trained model trained on the Wikipedia (INEX) Corpus, following the approach by [2]. This model provides embeddings of a dimension of 300.
Similar to the Word2Vec's sentence embeddings generation, we average the embeddings for all tokens in the target sentence.

Skip-Thoughts

The Skip-Thoughts [18] model was implemented using the code made available in Kiros' public github repository[2]. We use the available Comb-Skip model, which is a concatenation of the Uni-Skip and Bi-Skip models trained using the BookCorpus [22], similar to the approaches by [2] and [6]. The generated embeddings have 4800 dimensions (2400 from Uni-Skip + 2400 from Bi-Skip).
Unlike the previous two models, Skip-Thoughts embeddings store data from all tokens in the sentence.

ELMo

ELMo [20] embeddings were implemented using the TesorFlow Hub available model[3] with embeddings trained on the 1 Billion Word Benchmark. Unlike the previous models, ELMo uses character-based word representations and bidirectional LSTMs, which should provide more contextual information about the idiomatic use of VNCs.
The implemented model generates 1024 dimensional vectors.

---

[1] https://bitbucket.org/TomKenter/siamese-cbow/src/master/

[2] https://github.com/ryankiros/skip-thoughts

[3] https://tfhub.dev/google/elmo/2

4.4 Supervised Idiomatic Use Detection

Supervised classification of idiomatic vs non-idiomatic use is performed using a Support Vector Machine following experiments by previous researchers [2] [6] [9]. We replicate the performed experiments while adding the character-based embeddings generated by ELMo. We assume that the representations generated by ELMo using bidirectional LSTMs will outperform the other methods since they claim to be able to model both the characteristics of word use and how these uses vary across linguistic contexts [20].

SVM Classification

For our SVM Classifier, we used the Scikit-Learn's SVC packet[4] [23]. This implementation allows for easy parameter tuning, as well as compatibility with other Scikit-Learn packets for simple implementation of tools such as k-Fold Cross-Validation and Classification Reports to calculate metrics such as accuracy and F1-Scores.

4.5 Semi-Unsupervised Idiomatic Use Detection

We propose to use of three semi-unsupervised techniques for detection of idiomatic of target VNCs. The first one is the implementation of the k-Means clustering algorithm for detecting clusters of sentence embeddings, under the assumption that close vectors present a similar level of idiomaticity [9]. The second one is the use of Cosine Similarity between a target VNC and the sentence in which it is being used, under the assumption that VNICs are more semantically distinct from a given sentence than a non-idiomatic target [15] [16]. The third one is a weighted average between the Cosine Similarity of a VNC and the CForm value defined by Fazly et al. (2009), this works over the previously defined assumption, giving more weight if the VNC is in its Canonical Form [3] [15] [16].

k-Means Clustering

For the k-Means algorithm, we also used Scikit-Learn's implementation on the KMeans packet[5] [23]. This implementation was selected since it allows for quick parameter tuning for a different number of initial centroids.
We consider this to be a semi-supervised approach since we use a random sample of manually curated VNICs, tagged as idiomatic or literal, to determine the classification of each cluster based on the representative class of each cluster when classifying new unannotated sentences.

[4] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[5] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

Cosine Similarity

The Cosine Similarity metric was implemented by calculating Equation 9. The dot product and vector norms where calculated using NumPy[6] package for Python.
For this experiment, those VNC uses that score a value lower than a threshold are classified as idiomatic. We consider this to be a semi-supervised model since we find the optimal threshold using a set of manually curated VNICs, tagged as idiomatic or literal.

Cosine Similarity + CForm

Based on Fazly et al. (2009) CForm model, we propose a joint model using Cosine Similarity and CForm. Our assumption is that idiomatic use detected by CForm may be further tuned by taking into account the semantic difference between the target sentence and the VNC. The equation for this joint model is presented below, where $CForm(vnc)$ has a value of 1 if the target VNC occurs in its Canonical Form and 0 otherwise.

$$JointMetric(vnc, sent, \beta) = \beta(1 - cosSim(vnc, sent)) + (1 - \beta)CForm(vnc)$$

Equation 11 - Cosine Similarity + CForm

For this joint model, those VNC instances that score a value higher than a threshold are classified as idiomatic. We train this semi-unsupervised model by tuning for the optimal threshold value and the β parameter with a tagged VNC set.

[6] https://numpy.org/

5 Apparatus

This section aims to explain the data and parameters used for the proposed experiments, as well as the followed procedures and the metrics used to evaluate the results.

5.1 Dataset

BNC XML Corpora

The BNC XML Corpora [21] is a wide collection of texts that covers a large range of topics, including extracts from regional and national newspapers to popular fiction and academic books, among other kinds of texts, both from spoken and written language. The Corpora contains information about the lemmatized form of the tokens, as well as the POS Tags using the C5 tagset.

Preprocessing of the dataset was performed by removing all tags unnecessary to our purposes, leaving only the following tags:

- <s>: Indicates each sentence in the dataset. Its attribute indicates the sentence's id.
- <w>: Indicates tokens that correspond to words. Its attributes indicate the tokens's lemmatized form and POS Tag.
- <c>: Indicates tokens that correspond to punctuation signs. Its attribute indicates the token's POS Tag.

VNC-Tokens Dataset

The VNC-Tokens Dataset [19] is a list of instances of a set of manually curated VNICs that appear in the BNC XML Corpora. This dataset was selected following its previous use by [2], [3], and [9], which will provide a baseline for the performance of our classifiers.

The dataset contains 53 VNICs expression, collecting a total of roughly 3000 tagged instances that show if their use was "Literal" (L), "Idiomatic" (I), or "Unknown" (Q). Entries are tagged as follows:

ANNOTATION VERB_NOUN FILENAME SENTENCE-NUM

For example:

I blow_top A/A7/A7N 1279

Which means that the phrase "Blow Top" has idiomatic usage in sentence 1279 from the corpus A/A7/A7N. This corpus corresponds to an entry in the BNC XML Dataset.

Preprocessing of the dataset was performed by removing all VNIC instances that occur in a word window higher than 7 since this was problematic for the execution of our experiments. More on that in further sections.

5.2 Experimental Procedure

In order to test the performance of the ELMo character-based embeddings and test the performance of the proposed semi-supervised models, the following pipeline must be followed:

1. Data Pre-Processing
2. Sentence Embeddings Model Training
3. VNC Pattern Count
4. VNIC Candidates Extraction
5. Generate Sentence Embeddings
6. Experiment 1 – Support Vector Machines
7. Experiment 2 – k-Means Clustering
8. Experiment 3 – Cosine Similarity

Data Pre-Processing

Pre-processing of the BNC-XML Corpora is performed as mentioned in Section 5.1. After removing the tags, we store the indexed sentences for quick loading when performing the experiments, removing any sentence that shows inconsistencies (e.g. missing id number). We store each Corpora (from A to K) as three files: Original text, Lemmatized Text, and POS Tags.

Sentence Embeddings Model Training

We proceed to train the Word2Vec sentence embedding models with the extracted sentences for 5 epochs. Following King and Cook (2018) we then proceed to load the pre-trained models for the Siamese CBOW and Skip-Thoughts models [2]. We load the pre-trained ELMo model provided by TensorFlow Hub.

VNC Pattern Count

We proceed to process the extracted lemmatized sentences and their POS-Tags to get the VNC-Pattern counts for each Verb-Noun Combination found that follows any of the patterns included on Table 1 [3]. We set a **Max Window size of 7**, which means that we count any VNC that contains a maximum number of 5 tokens (including stop-words) in between. This number was set empirically after being able to capture a great number of patterns in the VNC-Dataset without being too broad.

VNIC Candidates Extraction

We take all the VNCs found and calculate their PMI, Lexical Fixedness, Syntactical Fixedness, and Overall Fixedness measures as described in Section 3.1 [3]. For the last three of these measures we need to consider the following parameters:

- K
- Alpha
- Lin's Thesaurus or Word2Vec's Most Similar

We set the **K** and **Alpha** to the values that showed the best-performing results for Fazly et. al (2009)., being **50** and **0.6** respectively [3]. We decide to use **Word2Vec's Most Similar** to find the most similar set of tokens for the verb and nouns required by the Lexical Fixedness measure since it should return a set of tokens more closely related to the actual use in the current Corpora.

Unlike Fazly et al. (2009), who only used VNCs that contain a verb of a pre-defined list of **Basic Verbs** [3], we want to explore the capacity of the measures to detect any sort of VNC with idiomatic capabilities without prior-knowledge of the Corpora used.

Also, since PMI has been demonstrated to work poorly with low-frequency samples [3], we set a minimum frequency threshold of **150** for any VNC to be considered.

We extract the **Top 20** ranked VNCs for each metric mentioned as our VNICs-Candidates Dataset, which is stored in the same format of the VNC-Tokens Dataset. To determine the success of the Lexical, Syntactic, and Overall Fixedness metrics in the task of recalling VNCs with idiomatic usage from a Corpora, we use the success of the PMI metric as our baseline, following Fazly et al. (2009) experiment [3].

Generate Sentence Embeddings

Sentence and VNC embeddings are generated using the described models in Section 4.3. We generate the embeddings for all sentences and VNCs that include the examples contained in the VNCs in the VNC-Tokens Dataset and the VNICs-Candidates Dataset. We do this for both lemmatized and un-lemmatized versions of the tokens.

Experiment 1 – Support Vector Machines

This first experiment aims to reproduce the results achieved by King and Cook (2018) on their tests on the use of Support Vector Machines to classify the idiomatic or literal use of the embeddings of sentences contained in the VNC-Tokens Dataset [2].
This experiment consists of measuring the performance of an SVM on classifying the sentence embeddings as idiomatic or literal. In addition to this, we add the rest of the mentioned metrics to the feature vector to see their impact on the accuracy of the model. This way, our feature vectors will consist of: **Sentence Embeddings**, **Sentence Embeddings + CForm**, and **Sentence Embeddings + CForm + Lexical Fixedness + Syntactic Fixedness + Overall Fixedness**.
We perform this experiment for three major purposes:

- Evaluate the performance of the original models (Word2Vec, Siamese CBOW, and Skip-Thoughts) when appending the extra features of Lexical Similarity, Syntactic Similarity, and Overall Similarity to the sentence embeddings vectors.
- Compare the performance of the ELMo character-based embeddings over the previously used models.
- Obtain the best hyper-parameters to train the best performing model to serve as an oracle to generate a silver-standard target set for the untagged VNICs from the VNICs-Candidates Dataset. This will be used to measure the semi-unsupervised approaches classification performance on this dataset.

When training this model, we will tune the **C** parameter of the SVM Classifier.

Experiment 2 – k-Means Clustering

The second round of experiments aims to observe the performance of the k-Means algorithm in generating clusters of sentences with VNCs that share idiomatic or literal use. We test this model for different number of clusters, in which a cluster is categorized as idiomatic or literal by the majority class of a "seed" sample extracted from the VNC-Tokens Dataset. Clusters are initialized randomly in each test with a set random seed

for replicability. Feature vectors used will be the same as those described for the **Experiment 1**.
As mentioned earlier, we will tune the **k** (cluster number) parameter of the k-Means algorithm.

Experiment 3 – Cosine Similarity

The third and last round of experiments tests the performance of the Cosine Similarity between the VNC and the sentence its being used in, as well as the weighted average of this measure and the CForm Model. For both of these metrics we first search for the optimal hyperparameter using the tagged VNC-Tokens Dataset to then test the performance on the untagged dataset. This parameter is a threshold that discriminates between idiomatic and literal cases.
Cosine Similarity: We test the Cosine Similarity metric (Equation 9) by determining those elements closer to a value of 0 idiomatic and those closer to 1 as literal. Thus, elements that score **lower** than a threshold are classified as idiomatic.
Cosine Similarity + CForm: We test the weighted average of Cosine Similarity and CForm (Equation 11) values by determining those close to 0 as literal and those with higher values idiomatic. This change occurs since we invert the value of the Cosine Similarity since a CForm value of 1 (i.e. True) indicates idiomaticity. Thus, elements that score **higher** than a threshold are classified as idiomatic.

5.3 Hyperparameters

Our training targets will be to find the best hyperparameters for the supervised and semi-supervised models. The following table shows the parameters to be tuned and the tested values for each experiment:

Model	**Parameter**	**Values**
Support Vector Machine Classifier	C	0.01, 0.1, 1, 10, 100
	Kernel Type	Linear
k-Means Clustering	Number of Clusters	2, 4, 6, 8, 10
Cosine Similarity	Classification Threshold	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9
Cosine Similarity + CForm	Classification Threshold	0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0
	Beta	0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0

Table 7 - Hyperparameter Tuning

5.4 Evaluation Metrics

To evaluate the performance of the classifiers, we will make use of the **Accuracy**, **F-1 Score**, **Precision**, and **Recall** metrics for binary classification. We consider these metrics to compare results between classifiers but will base the hyperparameter tuning decision on the **F-1 Score** metric.

We calculate these scores using the following formulas [13]:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Equation 12 - Accuracy Score Calculation

$$Precision = \frac{TP}{TP + FP}$$

Equation 13 - Accuracy Score Calculation

$$Recall = \frac{TP}{TP + FN}$$

Equation 14 - Accuracy Score Calculation

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Equation 15 - Accuracy Score Calculation

Where TP stands for True Positive, TN for True Negative, FP for False Positive, and FN for False Negative. In the following sections, we will be using the terms "Idiomatic/Literal" Precision, Recall, and F-1. When talking about the "Idiomatic" version of the metrics, we refer to idiomatic classified examples as the Positive and literal as the Negative, and vice versa for the "Literal" version of the metrics. We show these different metrics to provide a better insight of the power of the classifiers detecting both idiomatic and literal classifications; this is necessary, since a "Idiomatic" Recall of 1 may look as an optimal performance at first, but if it is accompanied by a "Literal" Recall of 0, this means the model may be classifying all examples as Idiomatic, which is not ideal.

6 Results

In this section we show the raw results for all our Literal vs Idiomatic classifiers.

6.1 Top-Scoring VNCs

Here we show the top-scoring VNICs based on the different metrics previously described.

Verb	Noun	PMI	Lex Fix	Syn Fix	Ova Fix	Freq
beg	**pardon**	**10.15622**	**3.982182**	**4.01588**	**4.002401**	**306**
purse	lip	9.110071	4.535113	5.354303	5.026627	160
remand	custody	8.915351	4.039665	1.894668	2.752667	291
grit	**tooth**	**8.79039**	**3.261212**	**5.303729**	**4.486722**	**170**
sen-tence	imprison-ment	8.006574	3.559355	1.847481	2.53223	265
sow	**seed**	**7.956956**	**2.583653**	**1.084038**	**1.683884**	**189**
bridge	**gap**	**7.822305**	**4.704997**	**1.256108**	**2.635664**	**211**
resist	temptation	7.485981	3.517369	1.083538	2.057071	237
bite	**lip**	**7.379781**	**2.810308**	**3.926534**	**3.480044**	**397**
kiss	cheek	7.311899	2.891318	1.396893	1.994663	255
lick	**lip**	**7.294396**	**2.661184**	**4.481111**	**3.75314**	**187**
smoke	cigarette	7.204077	2.062322	0.948518	1.39404	247
cele-brate	anniversary	7.168973	2.526501	1.465614	1.889969	307
shrug	shoulder	7.130682	4.424221	4.923199	4.723608	274
ratify	treaty	7.024577	2.802296	1.028802	1.7382	152
thank	**goodness**	**6.975595**	**2.394373**	**1.976647**	**2.143737**	**227**
re-search	grant	6.90542	4.959151	2.048731	3.212899	598
light	candle	6.842358	1.880882	0.43882	1.015645	159
burst	**tear**	**6.726277**	**3.540254**	**3.065902**	**3.255643**	**267**
ring	**bell**	**6.700443**	**2.099002**	**0.514403**	**1.148243**	**435**

Table 8 - Top 20 Scoring VNCs based on PMI (min frequency 150)

Verb	Noun	PMI	Lex Fix	Syn Fix	Ova Fix	Freq
score	try	6.68454	4.979805	0.693563	2.40806	322
re-search	grant	6.90542	4.959151	2.048731	3.212899	598
bridge	**gap**	**7.822305**	**4.704997**	**1.256108**	**2.635664**	**211**
glance	watch	6.664504	4.607793	4.029118	4.260588	272
purse	lip	9.110071	4.535113	5.354303	5.026627	160
shrug	shoulder	7.130682	4.424221	4.923199	4.723608	274
list	engage-ment	5.886745	4.42318	5.611822	5.136365	178
clear	**throat**	**5.63101**	**4.415943**	**4.010826**	**4.172873**	**281**
answer	guide	4.850938	4.167017	2.048731	2.896045	265
bring	boil	4.871689	4.117909	2.041157	2.871857	186
test	hypothe-sis	5.154724	4.08295	0.696914	2.051329	183
remand	custody	8.915351	4.039665	1.894668	2.752667	291
beg	**pardon**	**10.15622**	**3.982182**	**4.01588**	**4.002401**	**306**
date	century	4.865884	3.973631	1.431305	2.448235	276
vote	favour	4.278743	3.937476	1.583884	2.52532	187
let's	look	4.674024	3.90147	2.508218	3.065519	511
record	verdict	4.825872	3.825996	1.716195	2.560115	158
act	behalf	3.73109	3.796432	1.547675	2.447178	216
re-search	project	5.290824	3.754459	1.784355	2.572397	441
obtain	deception	5.139777	3.739019	1.252407	2.247052	155

Table 9 - Top 20 Scoring VNCs based on Lexical Fixedness (min frequency 150)

Verb	Noun	PMI	Lex Fix	Syn Fix	Ova Fix	Freq
list	engage-ment	5.886745	4.42318	5.611822	5.136365	178
purse	lip	9.110071	4.535113	5.354303	5.026627	160
grit	**tooth**	**8.79039**	**3.261212**	**5.303729**	**4.486722**	**170**
close	eye	3.416051	2.555601	4.989781	4.016109	1225
shrug	shoulder	7.130682	4.424221	4.923199	4.723608	274
will	engage-ment	0.823521	1.560609	4.791427	3.4991	195
roll	**eye**	**2.144891**	**1.884097**	**4.781811**	**3.622725**	**159**
shut	**eye**	**3.034754**	**2.414575**	**4.75417**	**3.818332**	**244**
narrow	eye	4.193732	2.916959	4.654824	3.959678	153
stop	**track**	**2.312636**	**2.132338**	**4.524485**	**3.567626**	**167**
open	**eye**	**1.802786**	**1.501028**	**4.483167**	**3.290311**	**762**
lick	**lip**	**7.294396**	**2.661184**	**4.481111**	**3.75314**	**187**
wash	**hand**	**2.494843**	**2.037479**	**4.459294**	**3.490568**	**311**
rise	**foot**	**2.420396**	**2.051098**	**4.396217**	**3.458169**	**404**
raise	eye	-0.11289	0.351049	4.262079	2.697667	216
list	official	3.976095	3.406567	4.107599	3.827186	179
shake	**head**	**5.65355**	**3.02429**	**4.055366**	**3.642936**	**3781**
glance	watch	6.664504	4.607793	4.029118	4.260588	272
beg	**pardon**	**10.15622**	**3.982182**	**4.01588**	**4.002401**	**306**
clear	**throat**	**5.63101**	**4.415943**	**4.010826**	**4.172873**	**281**

Table 10 - Top 20 Scoring VNCs based on Syntactical Fixedness (min frequency 150)

Verb	Noun	PMI	Lex Fix	Syn Fix	Ova Fix	Freq
list	engage-ment	5.886745	4.42318	5.611822	5.136365	178
purse	lip	9.110071	4.535113	5.354303	5.026627	160
shrug	shoulder	7.130682	4.424221	4.923199	4.723608	274
grit	**tooth**	**8.79039**	**3.261212**	**5.303729**	**4.486722**	**170**
glance	watch	6.664504	4.607793	4.029118	4.260588	272
clear	**throat**	**5.63101**	**4.415943**	**4.010826**	**4.172873**	**281**
close	**eye**	**3.416051**	**2.555601**	**4.989781**	**4.016109**	**1225**
beg	**pardon**	**10.15622**	**3.982182**	**4.01588**	**4.002401**	**306**
narrow	eye	4.193732	2.916959	4.654824	3.959678	153
list	official	3.976095	3.406567	4.107599	3.827186	179
shut	**eye**	**3.034754**	**2.414575**	**4.75417**	**3.818332**	**244**
lick	**lip**	**7.294396**	**2.661184**	**4.481111**	**3.75314**	**187**
shake	**head**	**5.65355**	**3.02429**	**4.055366**	**3.642936**	**3781**
roll	**eye**	**2.144891**	**1.884097**	**4.781811**	**3.622725**	**159**
lose	**temper**	**4.375711**	**3.125611**	**3.922125**	**3.603519**	**247**
stop	**track**	**2.312636**	**2.132338**	**4.524485**	**3.567626**	**167**
will	engage-ment	0.823521	1.560609	4.791427	3.4991	195
wash	**hand**	**2.494843**	**2.037479**	**4.459294**	**3.490568**	**311**
bite	**lip**	**7.379781**	**2.810308**	**3.926534**	**3.480044**	**397**
rise	**foot**	**2.420396**	**2.051098**	**4.396217**	**3.458169**	**404**

Table 11 - Top 20 Scoring VNCs based on Overall Fixedness (min frequency 150)

Method	**Accuracy**
PMI (Baseline)	45%
Lexical Fixedness	15%
Syntactic Fixedness	55%
Overall Fixedness	65%

Table 12 - Accuracy for VNIC Extraction

We searched for the VNIC candidates in the Online Cambridge Dictionary[7] to determine if they have idiomatic usage or not. You can find in Table 8, Table 9, Table 10, and Table 11 in **bold** the idiomatic VNCs found in text. Syntactical and Overall Fixedness metrics are the most successful, with a 55% and a 65% accuracy, while the lexical fixedness has a remarkably low performance, with a mere 15% accuracy. Because of this, for the semi-unsupervised models we will only focus on the performance results that were trained on the "Overall Fixedness" portion of VNIC-Candidates Dataset.

[7] https://dictionary.cambridge.org/us/

6.2 Experiment 1

Here are the results on the tests performed using the supervised Support Vector Machine model on different feature vectors across all different sentence embeddings. All tables show the results of the 10-Fold Cross Validation tests for the best performing C parameter based on the **F1-Score** metric, shown in Table 13 and Table 14.

We show both Accuracy and F1-Score for Idiomatic and Literal use prediction, in order to show a full picture of the performance of the classifier.

Unlemmatized Sentences

	C		
Model	-	+CF	+CF+Fix
Word2Vec	0.1	0.1	0.1
Siamese CBOW	1	1	1
Skip-Thoughts	1	1	1
ELMo	1	1	1

Table 13 - Best C Parameters for Unlemmatized Sentence Embeddings

Lemmatized Sentences

	C		
Model	-	+CF	+CF+Fix
Word2Vec	0.1	0.1	0.1
Siamese CBOW	1	1	100
Skip-Thoughts	1	10	1
ELMo	1	1	1

Table 14- Best C Parameters for Lemmatized Sentence Embeddings

For each table, find in **bold** the best performing model for each feature vector proposed.

Accuracy Score – Unlemmatized Sentences

	Training			Testing		
Model	-	+CF	+CF+Fix	-	+CF	+CF+Fix
Word2Vec	0.92551	0.93507	0.93788	0.9	0.902033	0.906910569
Siamese CBOW	0.77816	0.77816	0.77816	0.787	0.786992	0.78699187
Skip-Thoughts	0.911826	0.923697	0.92478	0.881	0.891057	0.89796748
ELMo	**0.960942**	**0.965247**	**0.96828**	**0.903**	**0.906504**	**0.912601626**

Table 15 - Mean accuracy scores using Unlemmatized Sentence Embeddings (SVM)

F1-Score – Idiomatic Prediction – Unlemmatized Sentences

	Training			Testing		
Model	-	+CF	+CF+Fix	-	+CF	+CF+Fix
Word2Vec	0.953419	0.959177	0.960905	0.937639	0.938788	0.941785471
Siamese CBOW	0.875241	0.875241	0.875241	0.880688	0.880688	0.880687675
Skip-Thoughts	0.946037	0.952844	0.953563	0.92832	0.933407	0.937593065
ELMo	**0.975326**	**0.977909**	**0.97982**	**0.938818**	**0.940793**	**0.944542948**

Table 16 - Mean F1-Scores for Idiomatic Prediction task using Unlemmatized Sentence Embeddings (SVM)

F1-Score – Literal Prediction – Unlemmatized Sentences

Model	Training			Testing		
	-	+CF	+CF+Fix	-	+CF	+CF+Fix
Word2Vec	0.81413	0.841422	0.84907	0.74648	0.753131	0.766561036
Siamese CBOW	0	0	0	0	0	0
Skip-Thoughts	0.75909	0.800196	0.802177	0.65304	0.698845	0.717123716
ELMo	0.90622	0.918684	0.92593	0.76335	0.776658	0.792504795

Table 17 - Mean F1-Scores for Literal Prediction task using Unlemmatized Sentence Embeddings (SVM)

Accuracy Score – Lemmatized Sentences

Model	Training			Testing		
	-	+CF	+CF+Fix	-	+CF	+CF+Fix
Word2Vec	0.924513	0.932895	0.93525	0.895	0.904472	0.905284553
Siamese CBOW	0.77816	0.77816	0.83761	0.787	0.786992	0.778861789
Skip-Thoughts	0.909651	0.960942	0.92397	0.879	0.895935	0.893902439
ELMo	**0.95333**	**0.96203**	**0.96375**	**0.899**	**0.906098**	**0.907723577**

Table 18- Mean accuracy scores using Lemmatized Sentence Embeddings (SVM)

F1-Score – Idiomatic Prediction – Lemmatized Sentences

Model	Training			Testing		
	-	+CF	+CF+Fix	-	+CF	+CF+Fix
Word2Vec	0.952702	0.957904	0.959355	0.934193	**0.940433**	0.940847652
Siamese CBOW	0.875241	0.875241	0.90243	0.880688	0.880688	0.867897249
Skip-Thoughts	0.944811	0.975517	0.953088	0.926791	0.935373	0.935390237
ELMo	**0.970613**	**0.97592**	**0.976987**	**0.936355**	0.940298	**0.941518089**

Table 19 - Mean F1-Scores for Idiomatic Prediction task using Lemmatized Sentence Embeddings (SVM)

F1-Score – Literal Prediction – Lemmatized Sentences

Model	Training			Testing		
	-	+CF	+CF+Fix	-	+CF	+CF+Fix
Word2Vec	0.812979	0.834664	0.841004	0.735952	0.75808	0.761405168
Siamese CBOW	0	0	0.515382	0	0	0.313085378
Skip-Thoughts	0.751166	0.90348	0.799514	0.646918	0.731466	0.701843888
ELMo	**0.886667**	**0.910267**	**0.914541**	**0.756688**	**0.778976**	**0.779789198**

Table 20 - Mean F1-Scores for Literal Prediction task using Lemmatized Sentence Embeddings (SVM)

As expressed in previous sections, ELMo embeddings outperform all the other metrics both in the idiomatic and literal prediction tasks in both Accuracy and F1-Score metrics. Interestingly, there is not a significant difference between the use of lemmatized over unlemmatized sentences as it was initially expected.

6.3 Experiment 2

We show the results for our semi-supervised k-Means model using different number of centroids.

Experiments on VNC-Tokens Dataset

The tables below show the Accuracy and F1-Scor for the different word embeddings and feature vectors. The results are coloured, with data in green showing the best values across the whole grid.

Accuracy Score – Unlemmatized Sentences

		Clusters				
Embedding Model	Added Features	2	4	6	8	10
Word2Vec	-	0.779046	0.779046	0.83245	0.82552	0.833673
	cForm	0.779046	0.779046	0.830004	0.825112	0.833673
	cForm + Fixedness	0.779046	0.779046	0.829189	0.819405	0.836527
Siamese CBOW	-	0.779046	0.779046	0.779046	0.784753	0.784753
	cForm	0.779046	0.779046	0.779046	0.784753	0.778638
	cForm + Fixedness	0.779046	0.779046	0.788015	0.788015	0.799022
Skip-Thoughts	-	0.779046	0.779046	0.779046	0.779046	0.802691
	cForm	0.779046	0.779046	0.779046	0.779046	0.778638
	cForm + Fixedness	0.779046	0.779046	0.779046	0.779046	0.779046
ELMo	-	0.779046	0.779046	0.779046	0.779046	0.803098
	cForm	0.779046	0.779046	0.779046	0.779046	0.779046
	cForm + Fixedness	0.779046	0.779046	0.779046	0.779046	0.779046

Table 21 - Mean accuracy scores using Unlemmatized Sentence Embeddings (k-Means)

F1-Score – Idiomatic Prediction – Unlemmatized Sentences

		Clusters				
Embedding Model	Added Features	2	4	6	8	10
Word2Vec	-	0.875802	0.875802	0.900508	0.891315	0.898608
	cForm	0.875802	0.875802	0.898614	0.89142	0.898051
	cForm + Fixedness	0.875802	0.875802	0.898375	0.886961	0.899322
Siamese CBOW	-	0.875802	0.875802	0.875802	0.878621	0.878621
	cForm	0.875802	0.875802	0.875802	0.878621	0.86727
	cForm + Fixedness	0.875802	0.875802	0.877762	0.877762	0.87788
Skip-Thoughts	-	0.875802	0.875802	0.875802	0.875802	0.885525
	cForm	0.875802	0.875802	0.875802	0.875802	0.870807
	cForm + Fixedness	0.875802	0.875802	0.875802	0.875802	0.875802
ELMo	-	0.875802	0.875802	0.875802	0.875802	0.880475
	cForm	0.875802	0.875802	0.875802	0.875802	0.875802
	cForm + Fixedness	0.875802	0.875802	0.875802	0.875802	0.875802

Table 22 - Mean F1-Scores for Idiomatic Prediction task using Unlemmatized Sentence Embeddings (k-Means)

F1-Score – Literal Prediction – Unlemmatized Sentences

Embedding Model	Added Features	Clusters				
		2	4	6	8	10
Word2Vec	-	0	0	0.469677	0.557851	0.537415
	cForm	0	0	0.474149	0.550785	0.548673
	cForm + Fixedness	0	0	0.464879	0.551165	0.565547
Siamese CBOW	-	0	0	0	0.05036	0.05036
	cForm	0	0	0	0.05036	0.333742
	cForm + Fixedness	0	0	0.202454	0.202454	0.432681
Skip-Thoughts	-	0	0	0	0	0.286136
	cForm	0	0	0	0	0.227596
	cForm + Fixedness	0	0	0	0	0
ELMo	-	0	0	0	0	0.441618
	cForm	0	0	0	0	0
	cForm + Fixedness	0	0	0	0	0

Table 23 - Mean F1-Scores for Literal Prediction task using Unlemmatized Sentence Embeddings (k-Means)

We can see in Table 21, Table 22, and Table 23 that the classifier's performance increases with the number of clusters available. We assume that this is because the random sample we extract from the VNC-Tokens Dataset has a bigger representation of idiomatic sentences, which in turn causes the clusters to follow this representation until the examples are more finely discriminated.

Accuracy Score – Lemmatized Sentences

Embedding Model	Added Features	Clusters				
		2	4	6	8	10
Word2Vec	-	0.779046	0.782715	0.844272	0.844272	0.830412
	cForm	0.779046	0.779046	0.845088	0.83775	0.833265
	cForm + Fixedness	0.779046	0.783123	0.841826	0.84468	0.84468
Siamese CBOW	-	0.779046	0.779046	0.779046	0.784753	0.784753
	cForm	0.779046	0.779046	0.779046	0.779046	0.780677
	cForm + Fixedness	0.779046	0.779046	0.788015	0.788015	0.790461
Skip-Thoughts	-	0.779046	0.779046	0.803098	0.790461	0.800652
	cForm	0.779046	0.779046	0.779046	0.779046	0.796168
	cForm + Fixedness	0.779046	0.779046	0.779046	0.779046	0.788015
ELMo	-	0.779046	0.779046	0.779046	0.779046	0.816551
	cForm	0.779046	0.779046	0.779046	0.779046	0.779046
	cForm + Fixedness	0.779046	0.779046	0.779046	0.779046	0.816959

Table 24 - Mean accuracy scores using Lemmatized Sentence Embeddings (k-Means)

F1-Score – Idiomatic Prediction – Lemmatized Sentences

Embedding Model	Added Features	Clusters				
		2	4	6	8	10
Word2Vec	-	0.875802	0.85933	0.907237	0.904691	0.896414
	cForm	0.875802	0.875802	0.907498	0.900599	0.899878
	cForm + Fixedness	0.875802	0.859705	0.905181	0.904391	0.905623
Siamese CBOW	-	0.875802	0.875802	0.875802	0.878621	0.878621
	cForm	0.875802	0.875802	0.875802	0.875802	0.872512
	cForm + Fixedness	0.875802	0.875802	0.877762	0.877762	0.873086
Skip-Thoughts	-	0.875802	0.875802	0.883305	0.87791	0.88382
	cForm	0.875802	0.875802	0.875802	0.870397	0.882959
	cForm + Fixedness	0.875802	0.875802	0.875802	0.875802	0.878391
ELMo	-	0.875802	0.875802	0.875802	0.875802	0.89083
	cForm	0.875802	0.875802	0.875802	0.875802	0.875802
	cForm + Fixedness	0.875802	0.875802	0.875802	0.875802	0.892249

Table 25 - Mean F1-Scores for Idiomatic Prediction task using Lemmatized Sentence Embeddings (k-Means)

F1-Score – Literal Prediction – Lemmatized Sentences

Embedding Model	Added Features	Clusters				
		2	4	6	8	10
Word2Vec	-	0	0.522829	0.515228	0.57461	0.532584
	cForm	0	0	0.52381	0.558758	0.501827
	cForm + Fixedness	0	0.522442	0.523342	0.586319	0.561565
Siamese CBOW	-	0	0	0	0.05036	0.05036
	cForm	0	0	0	0	0.215743
	cForm + Fixedness	0	0	0.202454	0.202454	0.399533
Skip-Thoughts	-	0	0	0.370274	0.261494	0.298422
	cForm	0	0	0	0.251381	0.211356
	cForm + Fixedness	0	0	0	0	0.174603
ELMo	-	0	0	0	0	0.42602
	cForm	0	0	0	0	0
	cForm + Fixedness	0	0	0	0	0.392422

Table 26 - Mean F1-Scores for Literal Prediction task using Lemmatized Sentence Embeddings (k-Means)

Table 25, Table 26, and Table 29 show the results for the k-Means models using sentence embeddings of the lemmatized version of the dataset. Performance stays roughly the same than in the unlemmatized version, being the most significant improvement the classification of literal use of VNCs for a small number of clusters for embeddings made using the Word2Vec model.

Experiments on VNIC-Candidates Dataset

These are the results on tests carried out on the VNIC-Tokens Dataset using the best performing number of clusters. Take these results with a grain of salt, since original

targets are not manually curated but a silver standard created using the best performing SVM model.
As previously stated, we are showing the results on the **top scoring VNICs using the Overall Fixedness metric** portion of the dataset.

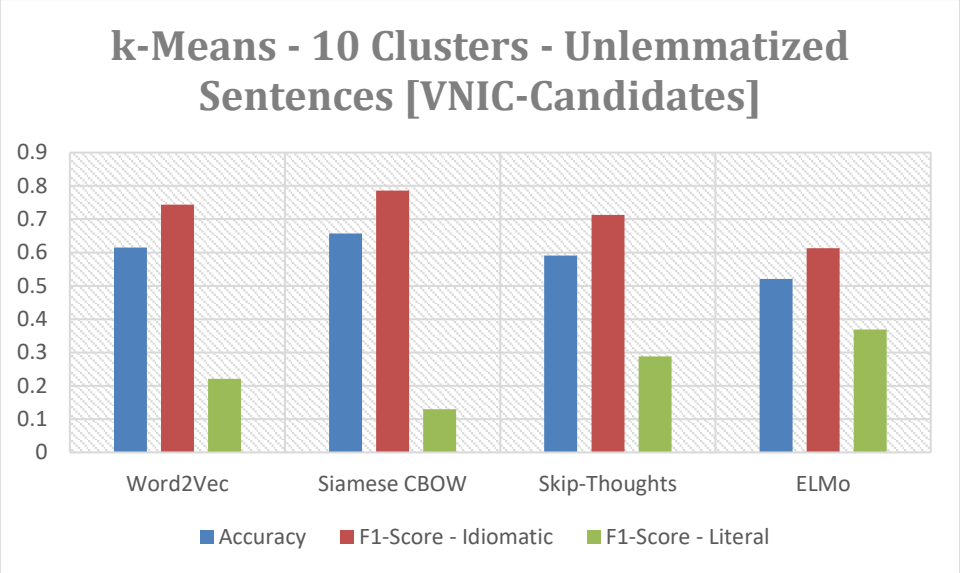


Figure 3 - kMeans Clustering on Unlemmatized Sentence Embeddings [VNIC-Candidates "OVA" Dataset]

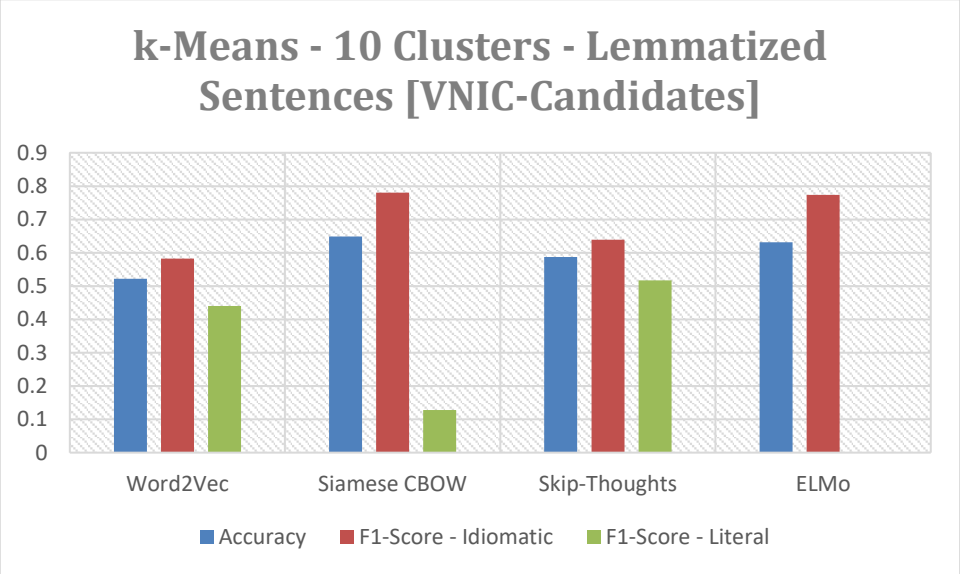


Figure 4 - kMeans Clustering on Lemmatized Sentence Embeddings [VNIC-Candidates "OVA" Dataset]

The unlemmatized sentences results show a high accuracy for the Word2Vec, Siamese CBOW, and Skip-Thoughts but, as it can be seen with the F1-Score metric for literal predictions, the clusters formed with these embeddings tend to overfit. Contrary to the experiments on the VNC-Tokens Dataset, the use of lemmatized sentences creates an increase in performance for the clusters formed with Word2Vec and Skip-Thoughts embeddings, while making the ELMo embeddings trained model to overfit.

6.4 Experiment 3

These are the results on both semi-supervised metrics proposed: Cosine Similarity and the weighted average between Cosine Similarity and Fazly's CForm Model.

Experiments on VNC-Tokens Dataset

First, we present the results on the Cosine Similarity metric on the tagged VNC-Tokens Dataset. For the next set of results, remember that we classify as "Idiomatic" those sentences that score lower values than the threshold for Cosine Similarity.

CForm (Baseline) - Accuracy and F1-Score

Model	Metric		
	Accuracy	F1-Score (I)	F1-Score (L)
CForm	0.7684468	0.85177453	0.47113594

Table 27 - Accuracy and F1-Scores for CForm Model

Cosine Similarity - Accuracy and F1-Score – Unlemmatized Sentences

Model	Metric	Threshold								
		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Word2Vec	Accuracy	0.23726	0.284142	0.351406	0.459845	0.590705	0.71545	0.770077	0.776192	0.778638
	F1 (I)	0.041005	0.15087	0.304937	0.500565	0.681674	0.81446	0.863965	0.872355	0.875373
	F1 (L)	0.366836	0.381254	0.39205	0.41190	0.42694	0.38986	0.25789	0.09256	0.01093
Siamese CBOW	Accuracy	0.742356	0.775785	0.775785	0.775785	0.775785	0.775785	0.775785	0.775785	0.775785
	F1 (I)	0.850662	0.873563	0.873563	0.873563	0.873563	0.873563	0.873563	0.873563	0.873563
	F1 (L)	0.062315	0.010791	0.010791	0.010791	0.010791	0.010791	0.010791	0.010791	0.010791
Skip-Thoughts	Accuracy	0.370974	0.609865	0.753771	0.774154	0.773746	0.772523	0.772931	0.774969	0.775785
	F1 (I)	0.369432	0.716276	0.855847	0.871819	0.871854	0.871251	0.871689	0.873045	0.873563
	F1 (L)	0.3725	0.3757	0.1564	0.0514	0.0348	0.0245	0.0142	0.0108	0.0108
ELMo	Accuracy	0.220954	0.220954	0.220954	0.232776	0.365267	0.693844	0.7766	0.779046	0.779046
	F1 (I)	0	0	0	0.029897	0.341649	0.80148	0.873965	0.875802	0.875802
	F1 (L)	0.361937	0.361937	0.361937	0.365475	0.387249	0.331256	0.017921	0	0

Table 28 - Accuracy and F1-Scores for Cosine Similarity Model using Unlemmatized Sentence Embeddings

Cosine Similarity - Accuracy and F1-Score – Lemmatized Sentences

Model	Metric	Threshold								
		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Word2Vec	Accuracy	0.228292	0.258051	0.312271	0.394619	0.496127	0.609865	0.730534	0.766408	0.777415
	F1 (I)	0.018663	0.090909	0.215714	0.383049	0.560768	0.718778	0.833794	0.865525	0.874483
	F1 (L)	0.364125	0.373278	0.387659	0.405762	0.409178	0.363273	0.288482	0.111628	0.017986
Siamese CBOW	Accuracy	0.741133	0.775785	0.775785	0.775785	0.775785	0.775785	0.775785	0.775785	0.775785
	F1 (I)	0.849348	0.873505	0.873563	0.873563	0.873563	0.873563	0.873563	0.873563	0.873563
	F1 (L)	0.081042	0.014337	0.010791	0.010791	0.010791	0.010791	0.010791	0.010791	0.010791
Skip-Thoughts	Accuracy	0.395842	0.610273	0.748064	0.774154	0.773746	0.772523	0.772931	0.774969	0.775785
	F1 (I)	0.408619	0.714286	0.852082	0.8717	0.871854	0.871251	0.871689	0.873045	0.873563
	F1 (L)	0.3825	0.387179	0.151099	0.057823	0.034783	0.024476	0.014159	0.010753	0.010791
ELMo	Accuracy	0.220954	0.220954	0.220954	0.224623	0.308194	0.61435	0.768854	0.779046	0.779046
	F1 (I)	0	0	0	0.011435	0.226879	0.725319	0.868354	0.875802	0.875802
	F1 (L)	0.361937	0.361937	0.361937	0.362173	0.374032	0.352941	0.053422	0	0

Table 29 – Accuracy and F1-Scores for Cosine Similarity Model using Lemmatized Sentence Embeddings

Tables show that accuracy and idiomatic F1-Score increase with the threshold, which suggests that most idiomatic examples lie in the lower values. This increase starts to halt around the **0.6** mark.

For a more complete look, see the Precision vs Recall plots below:

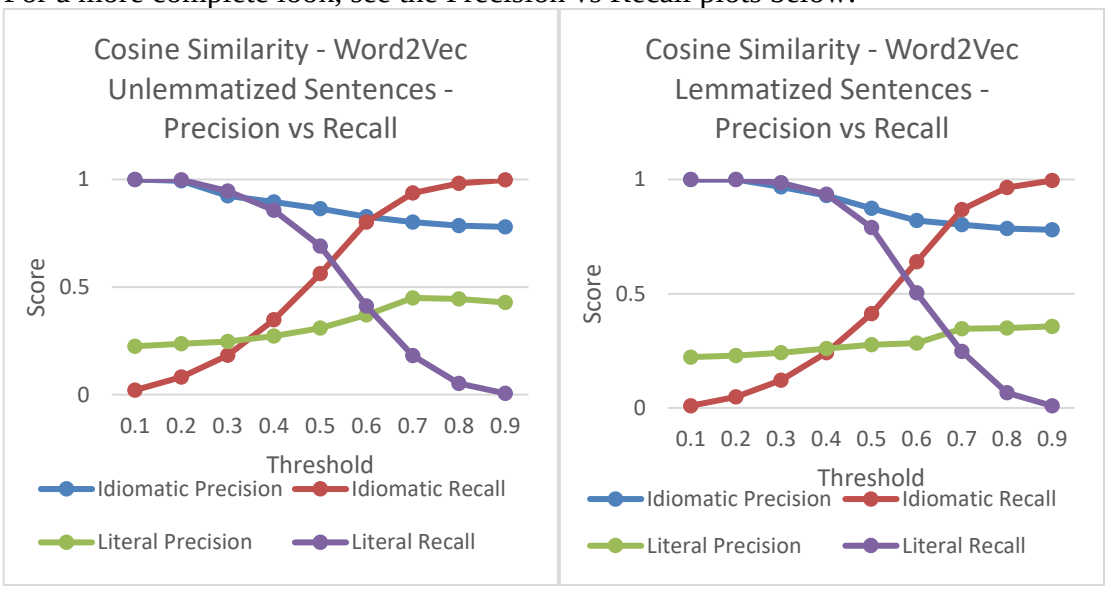


Figure 5 - Precision v Recall Scores for CosSim Model using Word2Vec Embeddings

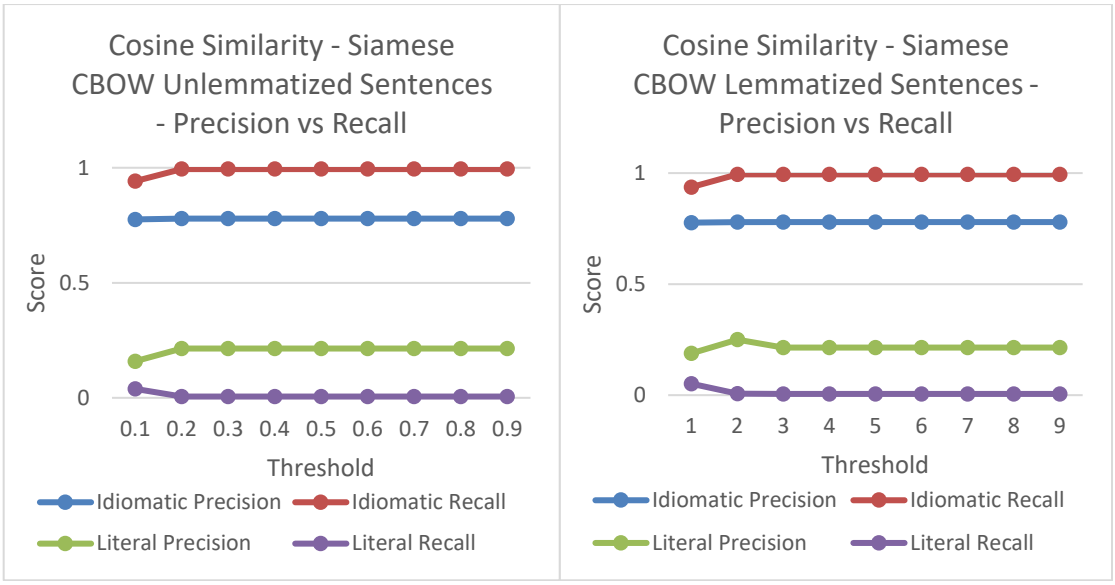


Figure 6 - Precision v Recall Scores for CosSim Model using Siamese CBOW Embeddings

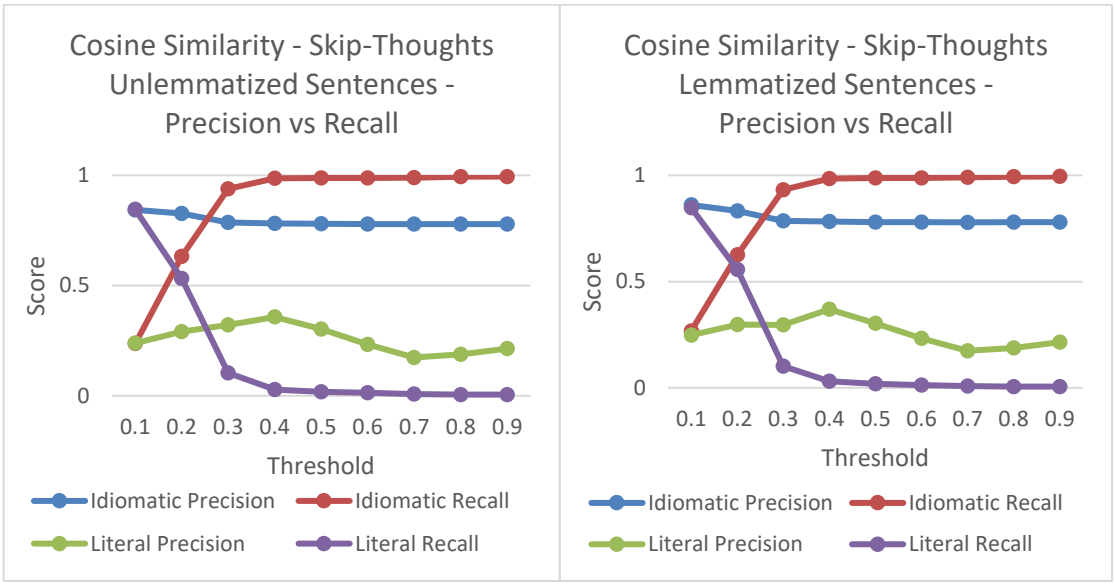


Figure 7 - Precision v Recall Scores for CosSim Model using Skip-Thoughts Embeddings

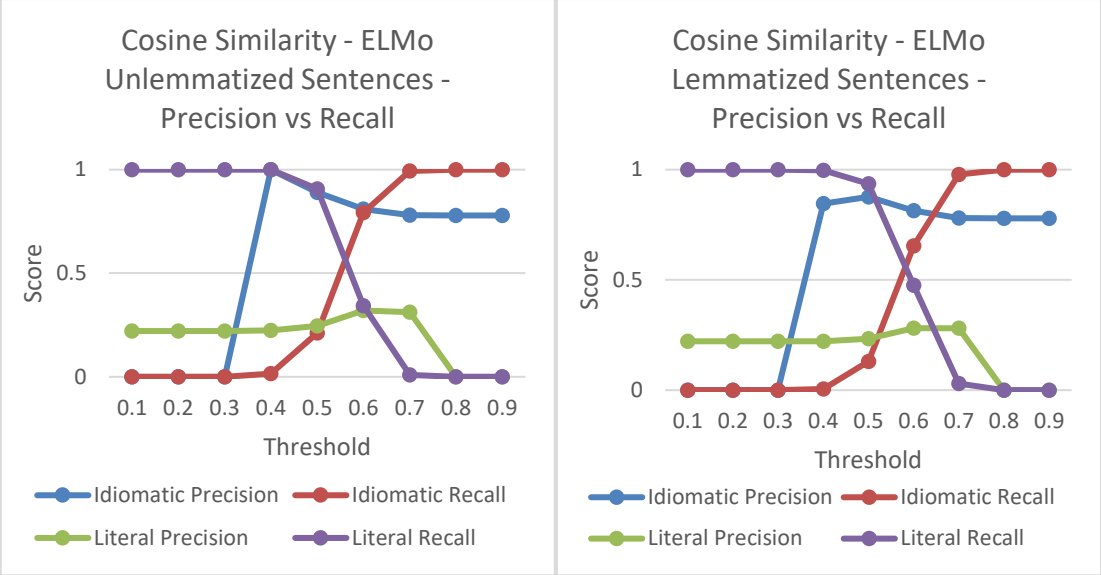


Figure 8 - Precision v Recall Scores for CosSim Model using ELMo Embeddings

The Word2Vec and ELMo Precision and Recall results show that, like the Accuracy and F1-Score metrics, around the 0.6 threshold mark there is a trade-off between idiomatic and literal recall, while the precision metric seems to have hit a tangent. The recall metric also shows that for the Word2Vec sentence embeddings, the Cosine Similarity between the VNC and the sentence is more evenly distributed from values 0 to 10, while for the ELMo embeddings, there is a clear indication that most idiomatic sentences have a cosine similarity **lower than 0.7**, while literal ones have a cosine similarity **higher than 0.4~0.5**. This is important for our analysis since it shows that ELMo embeddings better capture this distinction.

We now show the results of the combined metrics. Remember that the value of the Cosine Similarity metric is inverted on this experiment, so examples **over** the threshold are now categorized as idiomatic.

CosSim and CForm - Word2Vec Unlemmatized Sentences - Accuracy												
		BETA										
		0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
TRESHOLD	0	0.7684	0.7790	0.7790	0.7790	0.7790	0.7790	0.7790	0.7790	0.7790	0.7790	0.7786
	0.1	0.7684	0.7684	0.7852	0.7852	0.7831	0.7799	0.7803	0.7803	0.7795	0.7790	0.7786
	0.2	0.7684	0.7684	0.7684	0.7803	0.7852	0.7876	0.7852	0.7835	0.7827	0.7811	0.7762
	0.3	0.7684	0.7684	0.7684	0.7684	0.7713	0.7815	0.7852	0.7880	0.7884	0.7799	0.7701
	0.4	0.7684	0.7684	0.7684	0.7684	0.7684	0.7725	0.7799	0.7795	0.7843	0.7578	0.7155
	0.5	0.7684	0.7684	0.7684	0.7684	0.7684	0.7680	0.7680	0.7644	0.7232	0.6665	0.5907
	0.6	0.7684	0.7684	0.7684	0.7684	0.7680	0.7648	0.7383	0.6702	0.5768	0.5177	0.4598
	0.7	0.7684	0.7684	0.7684	0.7680	0.7676	0.6963	0.5740	0.4847	0.4179	0.3767	0.3514
	0.8	0.7684	0.7684	0.7680	0.7383	0.5740	0.4468	0.3734	0.3339	0.3106	0.2935	0.2841
	0.9	0.7684	0.7680	0.5740	0.3734	0.3106	0.2801	0.2581	0.2515	0.2413	0.2389	0.2373
	1	0.2210	0.2230	0.2230	0.2230	0.2230	0.2230	0.2230	0.2230	0.2230	0.2230	0.2230

Table 30 - Accuracy Scores for CosSim Model using Unlemmatized Word2Vec Sentence Embeddings

CosSim and CForm - Word2Vec Unlemmatized Sentences – F1-Score Idiomatic Prediction												
		BETA										
		0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
TRESHOLD	0	0.8518	0.8758	0.8758	0.8758	0.8758	0.8758	0.8758	0.8758	0.8758	0.8758	0.8755
	0.1	0.8518	0.8518	0.8695	0.8767	0.8770	0.8756	0.8760	0.8763	0.8759	0.8757	0.8754
	0.2	0.8518	0.8518	0.8518	0.8620	0.8695	0.8757	0.8767	0.8768	0.8768	0.8760	0.8724
	0.3	0.8518	0.8518	0.8518	0.8518	0.8547	0.8642	0.8695	0.8746	0.8768	0.8720	0.8640
	0.4	0.8518	0.8518	0.8518	0.8518	0.8518	0.8548	0.8617	0.8631	0.8661	0.8467	0.8145
	0.5	0.8518	0.8518	0.8518	0.8518	0.8518	0.8515	0.8509	0.8459	0.8102	0.7573	0.6817
	0.6	0.8518	0.8518	0.8518	0.8518	0.8515	0.8478	0.8212	0.7545	0.6512	0.5764	0.5006
	0.7	0.8518	0.8518	0.8518	0.8515	0.8478	0.7809	0.6454	0.5230	0.4157	0.3435	0.3049
	0.8	0.8518	0.8518	0.8515	0.8212	0.6454	0.4655	0.3343	0.2566	0.2080	0.1712	0.1509
	0.9	0.8518	0.8515	0.6454	0.3343	0.2080	0.1411	0.0909	0.0755	0.0510	0.0450	0.0410
	1	0.0000	0.0052	0.0052	0.0052	0.0052	0.0052	0.0052	0.0052	0.0052	0.0052	0.0052

Table 31 - F1-Scores for Idiomatic Prediction with CosSim Model using Unlemmatized Word2Vec Sentence Embeddings

CosSim and CForm - Word2Vec Unlemmatized Sentences – F1-Score Literal Prediction												
		BETA										
		0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
TRESHOLD	0	0.4711	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	0.1	0.4711	0.4711	0.3922	0.1648	0.0828	0.0426	0.0358	0.0182	0.0073	0.0073	0.0109
	0.2	0.4711	0.4711	0.4711	0.4615	0.3922	0.2693	0.1648	0.1106	0.0826	0.0661	0.0926
	0.3	0.4711	0.4711	0.4711	0.4711	0.4632	0.4417	0.3922	0.3158	0.2511	0.2128	0.2579
	0.4	0.4711	0.4711	0.4711	0.4711	0.4711	0.4746	0.4611	0.4323	0.4461	0.4233	0.3899
	0.5	0.4711	0.4711	0.4711	0.4711	0.4711	0.4707	0.4775	0.5000	0.4891	0.4674	0.4269
	0.6	0.4711	0.4711	0.4711	0.4711	0.4707	0.4825	0.5122	0.4978	0.4622	0.4401	0.4119
	0.7	0.4711	0.4711	0.4711	0.4707	0.5095	0.5050	0.4666	0.4397	0.4200	0.4067	0.3921
	0.8	0.4711	0.4711	0.4707	0.5122	0.4666	0.4267	0.4082	0.3966	0.3898	0.3844	0.3813
	0.9	0.4711	0.4707	0.4666	0.4082	0.3898	0.3804	0.3733	0.3712	0.3681	0.3673	0.3668
	1	0.3619	0.3625	0.3625	0.3625	0.3625	0.3625	0.3625	0.3625	0.3625	0.3625	0.3625

Table 32 - F1-Scores for Literal Prediction with CosSim Model using Unlemmatized Word2Vec Sentence Embeddings

CosSim and CForm – Siamese CBOW Unlemmatized Sentences - Accuracy												
		BETA										
		0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
TRESHOLD	0	0.7684	0.7790	0.7790	0.7790	0.7790	0.7790	0.7790	0.7790	0.7790	0.7790	0.7786
	0.1	0.7684	0.7827	0.7799	0.7799	0.7799	0.7799	0.7799	0.7799	0.7799	0.7795	0.7758
	0.2	0.7684	0.7684	0.7827	0.7799	0.7799	0.7799	0.7799	0.7799	0.7795	0.7758	0.7758
	0.3	0.7684	0.7684	0.7684	0.7827	0.7799	0.7799	0.7799	0.7799	0.7758	0.7758	0.7758
	0.4	0.7684	0.7684	0.7684	0.7684	0.7827	0.7799	0.7795	0.7758	0.7758	0.7758	0.7758
	0.5	0.7684	0.7684	0.7684	0.7684	0.7684	0.7786	0.7758	0.7758	0.7758	0.7758	0.7758
	0.6	0.7684	0.7684	0.7684	0.7684	0.7644	0.7644	0.7786	0.7762	0.7758	0.7758	0.7758
	0.7	0.7684	0.7684	0.7684	0.7644	0.7644	0.7644	0.7644	0.7786	0.7758	0.7758	0.7758
	0.8	0.7684	0.7684	0.7644	0.7644	0.7644	0.7644	0.7644	0.7644	0.7786	0.7754	0.7758
	0.9	0.7684	0.7644	0.7644	0.7644	0.7644	0.7644	0.7640	0.7570	0.7485	0.7554	0.7424
	1	0.2210	0.4680	0.4680	0.4680	0.4680	0.4680	0.4680	0.4680	0.4680	0.4668	0.4823

Table 33 - Accuracy Scores for CosSim Model using Unlemmatized Siamese CBOW Sentence Embeddings

CosSim and CForm – Siamese CBOW Unlemmatized Sentences – F1-Score Idiomatic Prediction												
		BETA										
		0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
TRESHOLD	0	0.8518	0.8758	0.8758	0.8758	0.8758	0.8758	0.8758	0.8758	0.8758	0.8758	0.8755
	0.1	0.8518	0.8697	0.8761	0.8761	0.8761	0.8761	0.8761	0.8761	0.8761	0.8759	0.8736
	0.2	0.8518	0.8518	0.8697	0.8761	0.8761	0.8761	0.8761	0.8761	0.8759	0.8736	0.8736
	0.3	0.8518	0.8518	0.8518	0.8697	0.8761	0.8761	0.8761	0.8761	0.8736	0.8736	0.8736
	0.4	0.8518	0.8518	0.8518	0.8518	0.8697	0.8761	0.8759	0.8736	0.8736	0.8736	0.8736
	0.5	0.8518	0.8518	0.8518	0.8518	0.8518	0.8669	0.8736	0.8736	0.8736	0.8736	0.8736
	0.6	0.8518	0.8518	0.8518	0.8518	0.8488	0.8488	0.8669	0.8738	0.8736	0.8736	0.8736
	0.7	0.8518	0.8518	0.8518	0.8488	0.8488	0.8488	0.8488	0.8669	0.8734	0.8736	0.8736
	0.8	0.8518	0.8518	0.8488	0.8488	0.8488	0.8488	0.8488	0.8489	0.8669	0.8730	0.8736
	0.9	0.8518	0.8488	0.8488	0.8488	0.8488	0.8488	0.8484	0.8432	0.8370	0.8504	0.8507
	1	0.0000	0.5367	0.5367	0.5367	0.5367	0.5367	0.5367	0.5367	0.5367	0.5378	0.5871

Table 34 – F1-Scores for Idiomatic Prediction with CosSim Model using Unlemmatized Siamese CBOW Sentence Embeddings

CosSim and CForm – Siamese CBOW Unlemmatized Sentences – F1-Score Literal Prediction												
		BETA										
		0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
TRESHOLD	0	0.4711	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	0.1	0.4711	0.3460	0.0110	0.0110	0.0110	0.0110	0.0110	0.0110	0.0110	0.0110	0.0108
	0.2	0.4711	0.4711	0.3460	0.0110	0.0110	0.0110	0.0110	0.0110	0.0110	0.0108	0.0108
	0.3	0.4711	0.4711	0.4711	0.3460	0.0110	0.0110	0.0110	0.0110	0.0108	0.0108	0.0108
	0.4	0.4711	0.4711	0.4711	0.4711	0.3460	0.0110	0.0110	0.0108	0.0108	0.0108	0.0108
	0.5	0.4711	0.4711	0.4711	0.4711	0.4711	0.3418	0.0108	0.0108	0.0108	0.0108	0.0108
	0.6	0.4711	0.4711	0.4711	0.4711	0.4668	0.4668	0.3418	0.0144	0.0108	0.0108	0.0108
	0.7	0.4711	0.4711	0.4711	0.4668	0.4668	0.4668	0.4658	0.3418	0.0179	0.0108	0.0108
	0.8	0.4711	0.4711	0.4668	0.4668	0.4668	0.4668	0.4668	0.4648	0.3418	0.0282	0.0108
	0.9	0.4711	0.4668	0.4668	0.4668	0.4668	0.4668	0.4673	0.4611	0.4496	0.3304	0.0623
	1	0.3619	0.3753	0.3753	0.3753	0.3753	0.3753	0.3753	0.3753	0.3753	0.3699	0.3060

Table 35 - F1-Scores for Literal Prediction with CosSim Model using Unlemmatized Siamese CBOW Sentence Embeddings

CosSim and CForm – Skip-Thoughts Unlemmatized Sentences - Accuracy												
		BETA										
		0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
TRESHOLD	0	0.7684	0.7799	0.7799	0.7799	0.7799	0.7799	0.7799	0.7799	0.7799	0.7799	0.7758
	0.1	0.7684	0.7701	0.7799	0.7795	0.7799	0.7799	0.7799	0.7799	0.7799	0.7758	0.7758
	0.2	0.7684	0.7684	0.7701	0.7799	0.7799	0.7795	0.7795	0.7795	0.7758	0.7758	0.7750
	0.3	0.7684	0.7684	0.7684	0.7701	0.7758	0.7811	0.7799	0.7795	0.7758	0.7742	0.7729
	0.4	0.7684	0.7684	0.7684	0.7684	0.7701	0.7746	0.7758	0.7758	0.7746	0.7733	0.7725
	0.5	0.7684	0.7684	0.7684	0.7684	0.7684	0.7660	0.7693	0.7689	0.7746	0.7733	0.7737
	0.6	0.7684	0.7684	0.7684	0.7684	0.7644	0.7636	0.7640	0.7676	0.7697	0.7737	0.7742
	0.7	0.7684	0.7684	0.7684	0.7644	0.7631	0.7615	0.7623	0.7640	0.7656	0.7709	0.7538
	0.8	0.7684	0.7684	0.7644	0.7623	0.7623	0.7615	0.7623	0.7362	0.7061	0.6612	0.6099
	0.9	0.7684	0.7644	0.7623	0.7623	0.7044	0.6038	0.5316	0.4798	0.4305	0.3942	0.3710
	1	0.2210	0.2344	0.2344	0.2344	0.2344	0.2344	0.2344	0.2344	0.2344	0.2344	0.2360

Table 36 - Accuracy Scores for CosSim Model using Unlemmatized Skip-Thoughts Sentence Embeddings

CosSim and CForm – Skip-Thoughts Unlemmatized Sentences – F1-Score Idiomatic Prediction												
		BETA										
		0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
TRESHOLD	0	0.8518	0.8761	0.8761	0.8761	0.8761	0.8761	0.8761	0.8761	0.8761	0.8761	0.8736
	0.1	0.8518	0.8530	0.8760	0.8759	0.8761	0.8761	0.8761	0.8761	0.8761	0.8736	0.8736
	0.2	0.8518	0.8518	0.8530	0.8753	0.8760	0.8758	0.8759	0.8759	0.8736	0.8736	0.8730
	0.3	0.8518	0.8518	0.8518	0.8530	0.8696	0.8765	0.8760	0.8758	0.8735	0.8725	0.8717
	0.4	0.8518	0.8518	0.8518	0.8518	0.8530	0.8637	0.8727	0.8733	0.8726	0.8718	0.8713
	0.5	0.8518	0.8518	0.8518	0.8518	0.8518	0.8501	0.8574	0.8673	0.8721	0.8716	0.8719
	0.6	0.8518	0.8518	0.8518	0.8518	0.8488	0.8482	0.8484	0.8550	0.8653	0.8711	0.8718
	0.7	0.8518	0.8518	0.8518	0.8488	0.8478	0.8464	0.8469	0.8482	0.8528	0.8625	0.8558
	0.8	0.8518	0.8518	0.8488	0.8471	0.8469	0.8463	0.8456	0.8241	0.7974	0.7589	0.7163
	0.9	0.8518	0.8488	0.8469	0.8456	0.7959	0.6912	0.5973	0.5228	0.4485	0.3910	0.3694
	1	0.0000	0.0340	0.0340	0.0340	0.0340	0.0340	0.0340	0.0340	0.0340	0.0340	0.0390

Table 37 - F1-Scores for Idiomatic Prediction with CosSim Model using Unlemmatized Skip-Thoughts Sentence Embeddings

CosSim and CForm – Skip-Thoughts Unlemmatized Sentences – F1-Score Literal Prediction												
		BETA										
		0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
TRESHOLD	0	0.4711	0.0110	0.0110	0.0110	0.0110	0.0110	0.0110	0.0110	0.0110	0.0110	0.0108
	0.1	0.4711	0.4719	0.0182	0.0110	0.0110	0.0110	0.0110	0.0110	0.0110	0.0108	0.0108
	0.2	0.4711	0.4711	0.4719	0.0625	0.0182	0.0146	0.0110	0.0110	0.0108	0.0108	0.0108
	0.3	0.4711	0.4711	0.4711	0.4719	0.2006	0.0359	0.0182	0.0146	0.0143	0.0107	0.0142
	0.4	0.4711	0.4711	0.4711	0.4711	0.4719	0.3486	0.0614	0.0283	0.0212	0.0211	0.0245
	0.5	0.4711	0.4711	0.4711	0.4711	0.4711	0.4675	0.3966	0.1043	0.0515	0.0347	0.0348
	0.6	0.4711	0.4711	0.4711	0.4711	0.4668	0.4659	0.4673	0.4148	0.2076	0.0765	0.0514
	0.7	0.4711	0.4711	0.4711	0.4668	0.4665	0.4667	0.4695	0.4703	0.4256	0.3146	0.1564
	0.8	0.4711	0.4711	0.4668	0.4666	0.4695	0.4687	0.4836	0.4727	0.4647	0.4304	0.3757
	0.9	0.4711	0.4668	0.4695	0.4836	0.4642	0.4471	0.4403	0.4283	0.4113	0.3974	0.3725
	1	0.3619	0.3660	0.3660	0.3660	0.3660	0.3660	0.3660	0.3660	0.3660	0.3660	0.3660

Table 38 - F1-Scores for Literal Prediction with CosSim Model using Unlemmatized Skip-Thoughts Sentence Embeddings

CosSim and CForm – ELMo Unlemmatized Sentences - Accuracy												
		BETA										
		0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
TRESHOLD	0	0.7684	0.7790	0.7790	0.7790	0.7790	0.7790	0.7790	0.7790	0.7790	0.7790	0.7790
	0.1	0.7684	0.7684	0.7758	0.7799	0.7790	0.7790	0.7790	0.7790	0.7790	0.7790	0.7790
	0.2	0.7684	0.7684	0.7684	0.7684	0.7758	0.7864	0.7799	0.7786	0.7790	0.7786	0.7790
	0.3	0.7684	0.7684	0.7684	0.7684	0.7684	0.7689	0.7758	0.7880	0.7839	0.7799	0.7766
	0.4	0.7684	0.7684	0.7684	0.7684	0.7684	0.7684	0.7684	0.7697	0.7758	0.7803	0.6938
	0.5	0.7684	0.7684	0.7684	0.7684	0.7684	0.7684	0.7684	0.7689	0.7236	0.5385	0.3653
	0.6	0.7684	0.7684	0.7684	0.7684	0.7684	0.7684	0.7599	0.5883	0.3579	0.2674	0.2328
	0.7	0.7684	0.7684	0.7684	0.7684	0.7684	0.6759	0.3579	0.2519	0.2275	0.2218	0.2210
	0.8	0.7684	0.7684	0.7684	0.7599	0.3579	0.2324	0.2218	0.2210	0.2210	0.2210	0.2210
	0.9	0.7684	0.7684	0.3579	0.2218	0.2210	0.2210	0.2210	0.2210	0.2210	0.2210	0.2210
	1	0.2210	0.2210	0.2210	0.2210	0.2210	0.2210	0.2210	0.2210	0.2210	0.2210	0.2210

Table 39 - Accuracy Scores for CosSim Model using Unlemmatized ELMo Sentence Embeddings

CosSim and CForm – ELMo Unlemmatized Sentences – F1-Score Idiomatic Prediction												
		BETA										
		0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
TRESHOLD	0	0.8518	0.8758	0.8758	0.8758	0.8758	0.8758	0.8758	0.8758	0.8758	0.8758	0.8758
	0.1	0.8518	0.8518	0.8587	0.8754	0.8757	0.8758	0.8758	0.8758	0.8758	0.8758	0.8758
	0.2	0.8518	0.8518	0.8518	0.8518	0.8587	0.8749	0.8754	0.8754	0.8757	0.8755	0.8758
	0.3	0.8518	0.8518	0.8518	0.8518	0.8518	0.8521	0.8587	0.8727	0.8758	0.8754	0.8740
	0.4	0.8518	0.8518	0.8518	0.8518	0.8518	0.8518	0.8518	0.8528	0.8587	0.8645	0.8015
	0.5	0.8518	0.8518	0.8518	0.8518	0.8518	0.8518	0.8518	0.8518	0.8125	0.6091	0.3416
	0.6	0.8518	0.8518	0.8518	0.8518	0.8518	0.8518	0.8446	0.6721	0.3167	0.1170	0.0299
	0.7	0.8518	0.8518	0.8518	0.8518	0.8518	0.7680	0.3167	0.0784	0.0166	0.0021	0.0000
	0.8	0.8518	0.8518	0.8518	0.8446	0.3167	0.0289	0.0021	0.0000	0.0000	0.0000	0.0000
	0.9	0.8518	0.8518	0.3167	0.0021	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table 40 - F1-Scores for Idiomatic Prediction with CosSim Model using Unlemmatized ELMo Sentence Embeddings

CosSim and CForm – ELMo Unlemmatized Sentences – F1-Score Literal Prediction												
		BETA										
		0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
TRESHOLD	0	0.4711	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	0.1	0.4711	0.4711	0.4576	0.0559	0.0037	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	0.2	0.4711	0.4711	0.4711	0.4711	0.4576	0.2702	0.0559	0.0073	0.0037	0.0000	0.0000
	0.3	0.4711	0.4711	0.4711	0.4711	0.4711	0.4716	0.4576	0.3659	0.1693	0.0559	0.0179
	0.4	0.4711	0.4711	0.4711	0.4711	0.4711	0.4711	0.4711	0.4705	0.4576	0.4198	0.3313
	0.5	0.4711	0.4711	0.4711	0.4711	0.4711	0.4711	0.4711	0.4745	0.4744	0.4368	0.3872
	0.6	0.4711	0.4711	0.4711	0.4711	0.4711	0.4711	0.4727	0.4469	0.3945	0.3741	0.3655
	0.7	0.4711	0.4711	0.4711	0.4711	0.4711	0.4625	0.3945	0.3705	0.3639	0.3622	0.3619
	0.8	0.4711	0.4711	0.4711	0.4727	0.3945	0.3654	0.3622	0.3619	0.3619	0.3619	0.3619
	0.9	0.4711	0.4711	0.3945	0.3622	0.3619	0.3619	0.3619	0.3619	0.3619	0.3619	0.3619
	1	0.3619	0.3619	0.3619	0.3619	0.3619	0.3619	0.3619	0.3619	0.3619	0.3619	0.3619

Table 41 - F1-Scores for Literal Prediction with CosSim Model using Unlemmatized ELMo Sentence Embeddings

When comparing both F1-Score tables (Literal and Idiomatic predictions) for each sentence embeddings, we can see a clear trade-off between these two values, which is expected. Most notably, similar to the results on the CosSim model, the CosSim + CForm model shows the best results around the **0.4 threshold** value (this is the inverse value of the previous **0.6**). It has to be noted, that looking at the extremes, at a BETA value of 0, the model (acting as Fazly's CForm model) shows great performance towards the detection of literal use of VNICs, while on the other end the CosSim model shows a better overall performance for idiomatic detection. Because of this, we decided to tip the scale in order to favour CosSim for our experiments on the VNIC-Candidates Dataset, setting the value for the BETA parameter to **0.6**.

Experiments on VNIC-Candidates Dataset

We now show the results on the CosSim and CosSim + CForm models. These results were obtained using the best performing metrics, being a **Threshold of 0.6** for the Cosine Similarity and a **Threshold of 0.4** and a **Beta of 0.6** for the CosSim + CForm.

Like our k-Means model's results, we show the results of the tests carried out using **the Overall Fixedness portion of our VNIC-Candidates Dataset**.

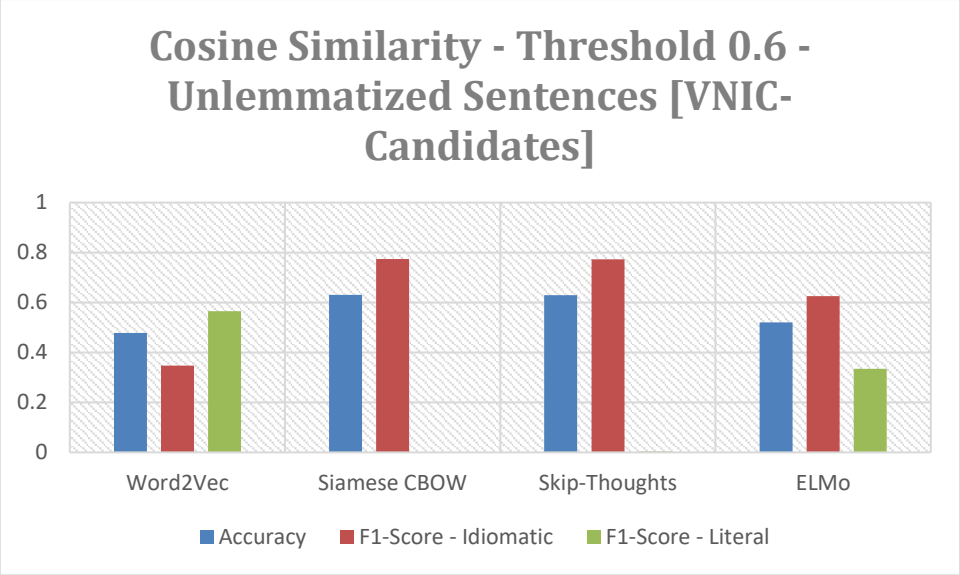


Figure 9 - CosSim Model on Unlemmatized Sentence Embeddings [VNIC-Candidates "OVA" Dataset]

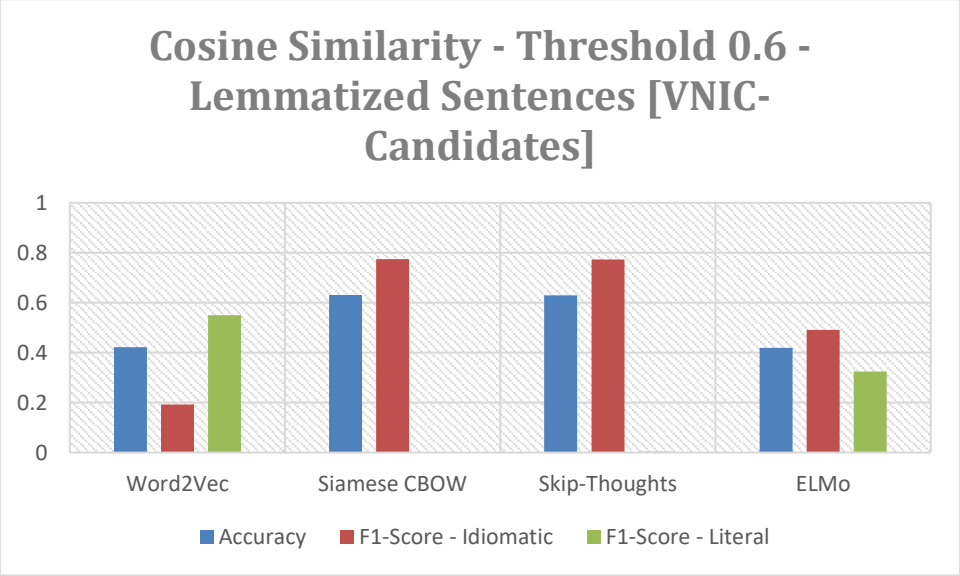


Figure 10 - CosSim Model on Lemmatized Sentence Embeddings [VNIC-Candidates "OVA" Dataset]

Figure 9 and Figure 10 show unsatisfactory results for the Siamese CBOW and Skip-Thoughts embeddings trained CosSim model, both overfitting for idiomatic predictions. For the two remaining embeddings, it can be clearly observed that embeddings from unlemmatized sentences have an overall better performance, with an increased performance on the F1-Score for Idiomatic predictions. This can be better explained with the results shown in Figure 5 and Figure 8; these two figures demonstrate how unlemmatized sentences have a more concentrated population of idiomatic examples on low Cosine Similarity values (see F1-Score – Idiomatic lines in both figures).

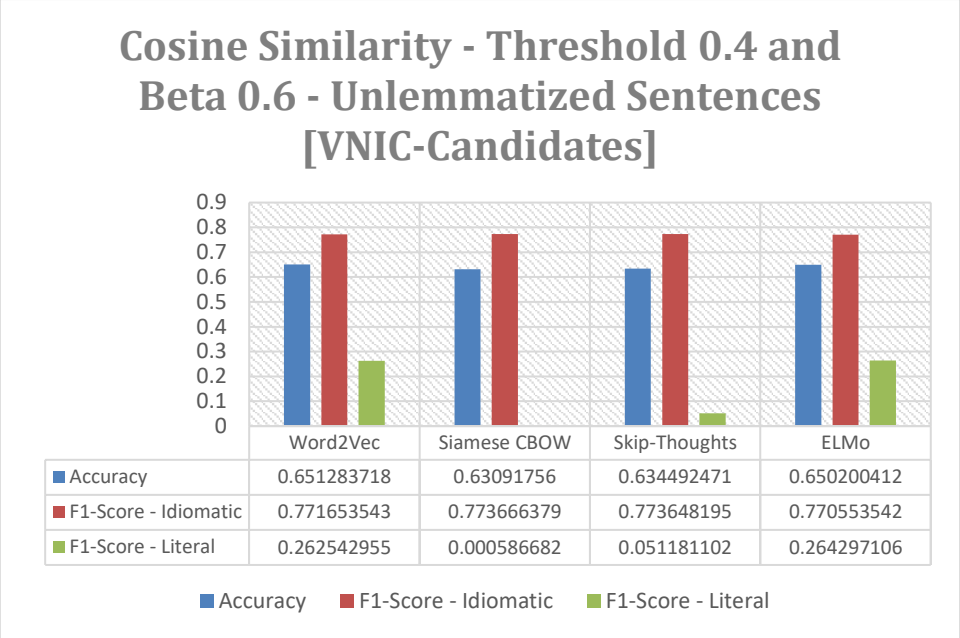


Figure 11 - CosSim and CForm Model on Unlemmatized Sentence Embeddings [VNIC-Candidates "OVA" Dataset]

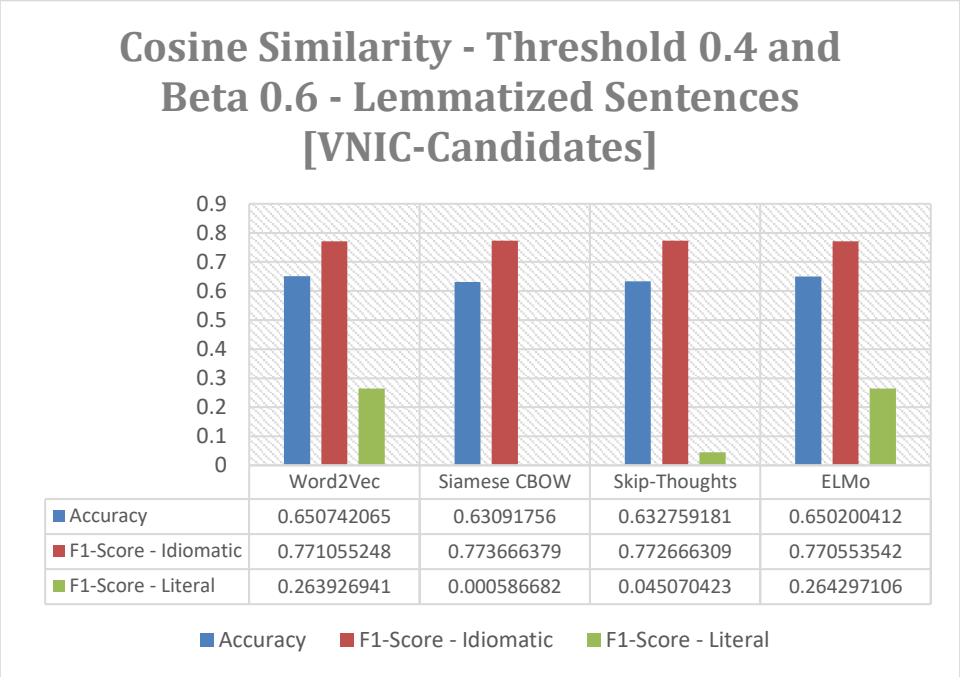


Figure 12 - CosSim and CForm Model on Lemmatized Sentence Embeddings [VNIC-Candidates "OVA" Dataset]

Unlike the CosSim model, the CosSim + CForm model presents more stable results for both unlemmatized and lemmatized tokens with the given parameters. Still, both Siamese CBOW and Skip-Thoughts sentence embeddings present ovefitting, categorizing most examples as idiomatic. We also see a decay on the F1-Score for Literal predictions for the Word2Vec and ELMo embeddings, but a significant increase in the F1-Score for Idiomatic predictions and accuracy score. We will discuss these observations in the following sections.

7 Discussion

7.1 VNIC Candidates Extraction from Corpora

We performed experiments to determine the effectiveness of Fazly's proposed metrics on the extraction of VNICs with a ranking system. As it can be seen on Table 12, both the Syntactical Fixedness and the Overall Fixedness metric present a score higher than the PMI baseline, which suggests that these metrics are strongly suited for extracting potentially new VNICs from untagged and unreviewed Corpora [3]. A $\chi 2$ test on the Top 40 candidates extracted by PMI and Overall Fixedness measures shows that there is a statistically significant difference at $p < 0.1$ to set the Overall Fixedness measure as a better model to find idiomatic VNCs in a Corpora. These results reflect Fazly et al. (2009) previous observations and conclusions regarding the applied metrics, demonstrating that combining the lexical and syntactical fixedness into a single measure, while giving more weight to the better measure (i.e. syntactic fixedness), results in a more effective classifier [3].

7.2 Supervised Classification

To evaluate the performance of the SVM using Word2Vec embeddings trained on the dataset used and the new ELMo embeddings, please refer to Table 4 and Table 15. The former shows King and Cook (2018) accuracy scores, which we use as a baseline, for their proposed feature vectors, while the latter shows our results. Note that for both feature vectors using Siamese CBOW and Skip-Thoughts the accuracy is consistent with that achieved by previous research. Nevertheless, there is an increase in this metric for both our Word2Vec (*80.4% -> 90.0%*) model trained with the BNC XML Corpora and the more ELMo embeddings. This suggest two things: a) Training the embedding model using the same Corpora you are analysing proves to be a factor in increased performance for classifying idiomatic use by using these vectors and; b) ELMo embeddings are capable of encoding semantic information regarding idiomatic use to a higher degree over previously existent sentence embedding models.

Aside from these two conclusions, we need to state that, based on the data returned by our experiments, we could not find a significant increase in performance on the addition of Fazly et al. (2009) CForm and Fixedness metrics to the feature vector. As well, the use of lemmatized tokens showed no improvements on the performance of the SVM classifier. One factor that could explain these results is that lemmatized sentences ignore the semantics of the original sentence used (i.e. He is **kicking the buckets** -> He be **kick the bucket**, which is showing literal use). The semantics of the original sentence influence showed by the patters defined by Fazly et al (2009) in Table 1, which show that, for example, patterns with plural noun use are considered different to those with singular use [3]. This loss of information may deter any improvements from occurring, but surprisingly it does not show any performance drop either, so it is worthy of further investigation.

Taking all of this into account, we therefore propose to use feature vectors, Word2Vec trained on the target Corpora and ELMo embeddings produced on original sentences without any additional data, as baselines for future supervised-classification problems of idiomatic vs literal use of VNICs.

7.3 Semi-Supervised Classification

We now proceed to analyse our results for the proposed measure for the semi-unsupervised classification of idiomatic use of VNICs, which is the k-Means Clustering with sample seed and the Cosine Similarity between the target VNC and the sentence in which its being used.

k-Means Clustering

We compare the performance of the k-Means clustering with seed against the CForm classifier, since it is another model for unsupervised-learning. The k-Means clustering semi-supervised model shows its best performance when a large group of clusters is present, as it can be seen from Table 21 to Table 26. Also, as mentioned in Section 7.2, performance was not enhanced by using lemmatized sentences embeddings, which we proposed in earlier stages. In here, the best performing model occurs when the feature vector contains Word2Vec unlemmatized sentence embeddings.

Comparison Between k-Means (W2V Embeddings) and CForm Models

Model	Metric		
	Accuracy	F1-Score (I)	F1-Score (L)
CForm (Baseline)	0.7685	0.8518	0.4711
k-Means (W2V Embeddings; 10 Clusters)	0.8337	0.8986	0.5374

Table 42 - Comparison between k-Means (10 Clusters; W2V Embedding) and CForm models

The k-Means proposal outperforms the baseline across all considered metrics. To see a better picture of the performance of both classifiers, we show the Confusion Matrixes for both tests.

k-Means Confusion Matrix [W2V]		**Dataset**	
		Idiom	Literal
Prediction	Idiom	1808	305
	Literal	103	237

Table 43 - Confusion Matric for k-Means Model (10 Clusters / Word2Vec Embeddings)

CForm Confusion Matrix (Baseline)		**Dataset**	
		Idiom	Literal
Prediction	Idiom	1632	289
	Literal	279	253

Table 44 - Confusion Matrix for CForm Model

A $\chi 2$ test on the results obtained show that the increase in performance by the k-Means model over the CForm model is significant at $p < 0.01$. These results are promising and show that our proposed semi-unsupervised model outperforms the baseline; However, recall that our proposal requires a seed of previously tagged examples to correctly set the classification of a given cluster, which is undesirable in the task of creating a model that can automatically detect idiomatic usage without expert knowledge on the data.

We attribute the success of the k-Means model to the idea that sentences containing idiomatic text lie closely in the vector space formed by sentence embedding models. These models can extract semantic and syntactic relationships among the individual tokens that conform a sentence [5] [17] [18] [20]. Under Fazly et al (2009) assumption that VNICs have a high syntactical fixedness score (as proven on Section 7.1), it is logical that sentences with idiomatic use will be close-by in vector space [3]. We are certainly surprised about the underperformance of the ELMo embeddings on this task, since it showed success on pair with the simpler Word2Vec embeddings model on the supervised SVM classifier. We ran five additional tests on various random seed values for the classifier seed.

Accuracy and F1-Score - Additional ELMo Test

Random Seed	Accuracy	F1-Score	
		Idiomatic	Literal
10	0.823889	0.895247	0.44757
20	0.772523	0.859233	0.407643
30	0.779046	0.875802	0
40	0.779046	0.875802	0
50	0.815736	0.88981	0.437811

Table 45 - Additional Test on Various Random Seeds for k-Means using ELMo Embeddings

Results demonstrate that the performance of the classifier is susceptible to over-fitting for idiomatic predictions based on the random seed. This may be because of the uneven distribution of classes on the data. Nevertheless, the distribution in the random seed must represent the distribution on the tests, also, the best performing seeds for the ELMo embeddings show poor results compared to those obtained with the Word2Vec embeddings; still, these results challenge the previous CForm model, which motivates further research using ELMo embeddings trained on different Corpora.

Cosine Similarity

Our first proposed metric, the Cosine Similarity measure between the target VNC and the context sentence, achieved unsatisfactory results in comparison to the selected baseline (CForm model). Table 27 shows the overall performance of our baseline, while Table 28 and Table 29 show the scores proposed model with the original and lemmatized sentences respectively. It is also no be noted that, as in the previous experiments, the use of lemmatized sentences over the original versions showed no improvements in performance, and even demonstrated to be significantly less effective in some cases; again, this could be attributed to the loss of semantic information by the use of lemmatized sentences.

Comparison Between CosSim (W2V Embeddings) and CForm Models

Model	Metric		
	Accuracy	F1-Score (I)	F1-Score (L)
CForm (Baseline)	0.7685	0.8518	0.4711
CosSim (Threshold 0.6)	0.7155	0.8145	0.3899

Table 46 - Comparison between CosSim (0.6 Threshold; W2V Embedding) and CForm models

CosSim Confusion Matrix [W2V]		**Dataset**	
		Idiom	Literal
Prediction	Idiom	1532	319
	Literal	379	223

Table 47 - Confusion Matric CosSim Model (0.6 Threshold / Word2Vec Embeddings)

$\chi 2$ test on the results of both models demonstrates that the difference in performance between our CosSim model and the baseline is significant at $p < 0.01$. This demonstrates that the simpler CForm model achieves better results overall. Nevertheless, we still believe the CosSim model is still a viable solution based on the lack of context it requires over the CForm model. The CForm model requires to extract all pattern counts from a Corpora relevant to the target sentences to be analysed, while embeddings have been demonstrated to be transferable to other contexts with a degree of success. This claim can be challenged by training the CForm model and the CosSim model (with Word2Vec embeddings) with the BNC-XML Corpora and applying them on a new VNIC dataset extracting sentences from a new Corpora.

Cosine Similarity + CForm

Lastly, we evaluate the performance of the proposed CosSim + CForm. Taking the best model available, with parameters 0.4 for threshold and 0.6 for Beta, we compare these results with the stablished CForm baseline.

Comparison Between CosSim + CForm (W2V Embeddings) and CForm Models

Model	Metric		
	Accuracy	F1-Score (I)	F1-Score (L)
CForm (Baseline)	0.7685	0.8518	0.4711
CosSim+CForm (Threshold 0.4; BETA 0.6)	0.7799	0.8617	0.4611

Table 48 - Comparison between CosSim+CForm (0.4 Threshold / 0.6 Beta; W2V Embedding) and CForm models

CosSim+CForm Confusion Matrix [W2V]		**Dataset**	
		Idiom	Literal
Prediction	Idiom	1682	311
	Literal	229	231

Table 49 - Confusion Matric CosSim+CForm Model (0.4 Threshold / 0.6 Beta / Word2Vec Embeddings)

As shown in Table 48, the joint CosSim+CForm fails to surpass the performance of the CForm baseline by a significant margin. A $\chi 2$ test result reveals to be not significant at $p < 0.05$, so the lack of any sort of visible improvement is a deterrent to implement this model over the simpler CosSim model. The small improvement shows that our proposal may be on the right track, but with the small improvement over the state-of-the-art it is safe to say that at this moment it is easier, and as effective, to use a the CForm model by itself.

We conclude that the CForm model poses a more adequate use as standalone classifiers instead of a joint model with the proposed CosSim.

8 Conclusion

The findings in this dissertation demonstrate that a) Fixedness measures alone create a good model to extract VNIC Candidates from a large, unreviewed Corpora, and b) Embeddings sensible to context semantics outperform other embeddings models with fixed embeddings. Our results show how the Word2Vec model performance increased after being trained on the Corpora containing the target examples, and how a semantically sensible model, such as ELMo, also produces outstanding results even if trained on a different dataset. Findings also demonstrated how these embeddings can be used to form clusters of semantically similar sentences, which share literal or idiomatic meaning with good results, that show a significant increase over the state-of-the-art classification, but still require a manually tagged seed that may deter their use from a completely unsupervised model such as CForm. Also, the performance of this semi-unsupervised model relies heavily on the quality of the initial example seed, which is a new problem itself.

On the other end, our experiments also show that the performance of the proposed CosSim model, which aims to make use of the found strengths of the mentioned embeddings, lacks in comparison of previous, simpler methods for idiomatic use identification, such as Fazly et al. (2009) CForm. This also translates to the proposed CosSim+CForm model, that aimed to combine the strengths of the previous state-of-the-art and our assumptions of vector distance between idiomatic VNCs and their context; this model made no contribution to the initial CForm model since performance failed to increase in a significant way. We are confident, however, that the quality of this model may improve as more powerful embedding models emerge since results demonstrate how this implementation achieved a certain degree of success.

In summary, the proposed embeddings surpass previous similar research on the topic, translating adequately to other unsupervised models such as the k-Means clustering algorithm. However, these vectors failed to capture the assumed semantic distance between the idiomatic phrase and the target sentence, causing our CosSim model to underperform.

8.1 Future Work

We encourage the use of the proposed embeddings, as well as the use of clustering algorithms, to challenge our claims in different Corpora. We also encourage the implementation of emerging sentence embeddings models, such as CMOW [24], to continue exploring the power of these resources on the idiomatic identification task.

We also believe that ELMo embeddings trained on the same corpora in which the idiomatic examples lie may also increase the performance in results as observed on the Word2Vec embeddings.

Lastly, we encourage the use of the CosSim model with new, emerging word-embeddings models that may capture the assumed semantic distance between VNC and sentence.

References

[1] I. A. Sag, T. Baldwin, F. Bond, A. Copestake and D. Flickinger, “Multiword Expressions: A Pain in the Neck for NLP,” in *Proceedings of the 3rd International Conference on Intelligent Text Processing and Computational Linguistics*, Mexico City, 2002.

[2] M. King and P. Cook, “Leveraging distributed representations and lexico-syntactic fixedness for token-level prediction of the idiomaticity of English verb-noun combinations,” in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, Melbourne, 2018.

[3] A. Fazly, P. Cook and S. Stevenson, “Unsupervised Type and Token Identification of Idiomatic Expressions,” *Computational Linguistics,* vol. 35, no. 1, pp. 61-103, 2009.

[4] T. Baldwin and K. Su Nam, “Multiword Expressions,” in *Handbook of Natural Language Processing*, Boca Raton, CRC Press, 2010, pp. 267-292.

[5] T. Kenter, A. Borisov and M. de Rijke, “Siamese CBOW: Optimizing Word Embeddings for Sentence Representations,” in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, Berlin, 2016.

[6] G. D. Salton, R. J. Ross and J. D. Kelleher, “Idiom Token Classification using Sentential Distributed Semantics,” in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, Berlin, 2016.

[7] C. Sporleder and L. Li, “Unsupervised Recognition of Literal and Non-Literal Use of Idiomatic Expressions,” in *Proceedings of the 12th Conference of the European Chapter of the ACL*, Athens, 2009.

[8] S. Z. Riehemann, “A Constructional Approach to Idioms and Word Formation,” Stanford University, Stanford, 2001.

[9] W. Gharbieh, V. C. Bhavsar and P. Cook, “A Word Embedding Approach to Identifying Verb-Noun Idiomatic Combinations,” in *Proceedings of the 12th Workshop on Multiword Expressions*, Berlin, 2016.

[10] A. Savary, C. Ramish, S. R. Cordeiro, F. Sangati, V. Vincze, B. QasemiZadeh, M. Candito, F. Cap, V. Giouli, I. Stoyanova and A. Doucet, “The PARSEME Shared Task on Automatic Identification of Verbal Multiword Expressions,” in *Proceedings of the 13th Workshop on Multiword Expressions*, Valencia, 2017.

[11] T. M. Cover and J. A. Thomas, Elements of Information Theory, New York: John Wiley & Sons, Inc., 1991.

[12] J. Birke and A. Sarkar, “A Clustering Approach for the Nearly Unsupervised Recognition of Nonliteral Language,” in *11th Conference of the European Chapter of the Association for Computational Linguistics*, Trento, 2006.

[13] E. Alpaydin, Introduction to Machine Learning, 3rd ed., Cambridge, Massachusetts: Massachusetts Institute of Technology, 2014.

[14] M. Ester, H. P. Kriegel, J. Sander and X. Xu, “A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise,” in *KDD'96 Proceedings of the*

Second International Conference on Knowledge Discovery and Data Mining, Portland, 1996.

[15] M. Pershina, Y. He and R. Grishman, "Idiom Paraphrases: Seventh Heaven vs Cloud Nine," in *Proceedings of the First Workshop on Linking Computational Models of Lexical, Sentential and Discourse-level Semantics*, Lisbon, 2015.

[16] C. Liu and R. Hwa, "Representations of Context in Recognizing the Figurative and Literal Usages of Idioms," in *Thirty-First AAAI Conference on Artificial Intelligence*, Pittsburgh, 2017.

[17] T. Mikolov, K. Chen, G. Corrado and J. Dean, "Efficient Estimation of Word Representations in Vector Space," in *Proceedings of Workshop at the International Conference on Learning Representations*, Scottsdale, 2013.

[18] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba and S. Fidler, "Skip-Thought Vectors," *Advances in Neural Information Processing Systems*, vol. 28, pp. 3276-3284, 2015.

[19] P. Cook, A. Fazly and S. Stevenson, *The VNC-Tokens Dataset,* Toronto: University of Toronto, 2008.

[20] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee and L. Zettlemoyer, "Deep contextualized word representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, New Orleans, 2018.

[21] B. Consortium, *The British National Corpus, version 3 (BNC XML Edition),* Oxford: University of Oxford, 2007.

[22] Y. Zhu, R. Kiron, R. S. Zemel, R. Salakhutdinov , R. Urtasun, A. Torralba and S. Fidler, "Aligning Books and Movies: Towards Story-like Visual Explanations," *CoRR,* vol. abs/1506.06724, 2015.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and É. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research,* no. 12, pp. 2825-2830, 2011.

[24] F. Mai, L. Galke and A. Scherp, "CBOW Is Not All You Need: Combining CBOW with the Compositional Matrix Space Model," *Computing Research Repository*, vol. abs/1902.06423, 2019.

[25] R. Rehuek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, Valletta, 2010.