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**: 06 July 2019

**Abstract**. Idiomatic use of Verb-Noun Combinations (VNCs) has posed a challenge, particularly for the tasks of text translation and semantic parsing, for NLP researchers since these may also pose literal meaning under particular contexts. We propose that the combination of previous word embeddings based supervised classifiers with unsupervised fixedness metrics can help to further increase the performance of available models. We base these claims on the fact VNCs are known to exhibit lexico-syntactic fixedness, which is carried inside distributed word vector representations extracted from Word2Vec’s Skip-Gram, Siamese CBOW, and Skip-thoughts. We expand on this by experimenting on the recently developed ELMo embeddings. Also, under the assumption that these encodings are linearly separable in vector space, we propose the use of unsupervised clustering algorithms to present a competent learning model for unlabelled data. We carry these experiments using the manually curated VNC-Dataset.

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

Table of Contents

[1 Introduction 3](#_Toc14369319)

[2 Background 5](#_Toc14369320)

[2.1 Multiword Expressions 5](#_Toc14369321)

[2.2 Types of Multiword Expressions 5](#_Toc14369322)

[2.3 Verb-Noun Idiomatic Combinations 6](#_Toc14369323)

[3 Literature Review 8](#_Toc14369324)

[3.1 Unsupervised Methods 8](#_Toc14369325)

[Canonical Forms 11](#_Toc14369326)

[3.2 Supervised Methods 11](#_Toc14369327)

[3.3 Further Word Embedding Models 14](#_Toc14369328)

[3.4 ELMo 14](#_Toc14369329)

[3.5 CMOW 15](#_Toc14369330)

[3.6 Further Unsupervised Methods 15](#_Toc14369331)

[4 Methods 17](#_Toc14369332)

[4.1 Fixedness Metrics [Done] 17](#_Toc14369333)

[Verb-Noun Combination Pattern Counts [Done] 17](#_Toc14369334)

[Pointwise Mutual Information [Done] 18](#_Toc14369335)

[Lexical Fixedness [Done] 19](#_Toc14369336)

[Syntactic Fixedness [Done] 19](#_Toc14369337)

[Overall Fixedness [Done] 20](#_Toc14369338)

[Canonical Forms [Done] 20](#_Toc14369339)

[4.2 Potential VNICs Extraction [Done] 20](#_Toc14369340)

[Frequency Considerations 20](#_Toc14369341)

[Metrics 21](#_Toc14369342)

[Instance Extraction 21](#_Toc14369343)

[4.3 Word Embeddings 21](#_Toc14369344)

[Word2Vec 21](#_Toc14369345)

[Siamese CBOW 22](#_Toc14369346)

[Skip-Thoughts 22](#_Toc14369347)

[ELMo 22](#_Toc14369348)

[CMOW 22](#_Toc14369349)

[4.4 Supervised Idiomatic Use Detection 22](#_Toc14369350)

[SVM Classification 22](#_Toc14369351)

[4.5 Unsupervised Idiomatic Use Detection 23](#_Toc14369352)

[k-Means Clustering 23](#_Toc14369353)

[Cosine Similarity 23](#_Toc14369354)

[5 Apparatus 24](#_Toc14369355)

[5.1 Dataset 24](#_Toc14369356)

[BNC XML Corpora 24](#_Toc14369357)

[VNC-Tokens Dataset 24](#_Toc14369358)

[5.2 Experimental Procedure 24](#_Toc14369359)

[5.3 Hyperparameters 24](#_Toc14369360)

[5.4 Evaluation Metrics 24](#_Toc14369361)

[6 Results 25](#_Toc14369362)

[7 Discussion 26](#_Toc14369363)

[8 Conclusion 27](#_Toc14369364)

[9 Future Work 28](#_Toc14369365)

[10 Appendices 29](#_Toc14369366)

# Introduction

Idiomatic use of phrases has always been a problem in the field of Natural Language Processing (NLP) since its appearance in text is challenging for important tasks such as machine translation and semantic parsing. 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 they are context dependent.

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), that consist of a single verb with a noun in its direct object position (e.g. Break a leg) [1] [4]. Research on idiomatic phrases focuses on VNICs since they are a common type of semantically-idiomatic MWE across several languages. 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 [2].

To demonstrate the complexity of VNIC detection, take for example *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 [1] [5] [6]. Unsupervised methods have also seemed some degree of success on the past [2] [7], but their reliability falls short under the performance of the supervised methods. Both implementations differ from the way they form 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.

This project proposes the use of joint supervised and unsupervised techniques for detecting non-literal usage of Verb-Noun Combinations. We also use unsupervised metrics to automatically extract strong VNICs candidates from a given Corpora.

The proposal first will present findings on the area by relevant previous research on the topic. It will explain their approach and the success of it, and why these ideas are relevant for the task at hand. The next section describes the goals to be met to take this project into completion, as well as the scope and limitations we observe at time of writing. After the goals section we refer to the dataset to be used, the required tools, and the overall steps needed to complete this project. This section is required since it proves the feasibility of our approach with the considered constraints.

Following this, we will present the techniques used to evaluate the performance of our newly trained models over the previous research and how it compares as to determine if our proposal was successful in creating more accurate models for the idiomatic usage detection task.

Lastly, we present the individual tasks needed to fulfil the project in a Work Breakdown Structure diagram that will illustrate the work to be done in a clear decomposition of the milestones to be met. This diagram will be complemented with a Gantt Chart that sets the time boundaries for each task and presents the order in which the different stages of development and evaluation will be done keeping the project deadline in mind.

# Background

Add: Importance of MWE Research in the field of NLP and complexity of generalisation of tools for recognising idioms based on compositionality.

Add: Target VNC research on focusing on Non-decomposable idioms. # Discuss with supervisor.

Add: Mention Verb-Particle compositions and why we don’t consider them.

## 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 [3] [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.

## 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]. Verb-Particle Constructions (VPCs) consist of a Verb and an obligatory particle in the form of an intransitive preposition (e.g. *play around*, *take off*), but including adjectives (e.g. *cut short*, *band together*) and verbs (e.g. *let go*, *let fly*) [4]. [CONTINUE HERE <https://people.eng.unimelb.edu.au/tbaldwin/pubs/handbook2009.pdf>]

## Verb-Noun Idiomatic Combinations

*EXPLAIN THE MEANING OF VNICs ACCORDING TO* [3] *AND* [4]

Recent research of non-literal usage of MWEs focuses on VNIC analysis since they have been able to extract lexical and semantic consistencies across different idiomatic phrases in the English language. First is the observation that most idiomatic VNICs exhibit **lexico-syntactic fixedness** [1] [2]; 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 [1].

**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 [2] (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 [2].

**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**) [2].

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 stablishes the **canonical form**, which is the standard form of a VNC, also known as its “preferred usage” [2]. Subsequent research works on the assumption that idiomatic VNCs are more likely to appear on canonical form that non-idiomatic phrases [2].

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 [2] [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 [2]. 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** [2]. 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 [2].

# Literature Review

Much research on MWE identification focuses on the task of detecting specific kinds on MWEs, such as English VNCs [1] [2] [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].

This research proposal will 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 [2]. 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.

## Unsupervised Methods

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

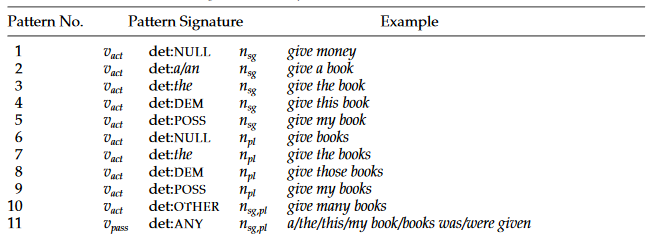


Table 1 - VNC Patterns by [2]. A pattern signature is composed of a verb in active () or passive (); a determiner (det) that can be NULL, indefinite (*a/an*), definite (*the*), demonstrative (DEM), or possessive (POSS); and a noun that can be singular () or plural ()

An approach by [2] 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, [2] 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.

Equation 1 - PMI for Verb+Noun Combinations in [2]

Where , in which , being a parameter for the number of closest verbs to target and the number of closest nouns to target [2]; is the total number of verb-object pairs in the corpus;, , and 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.

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

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 [2]. The higher the degree, the more lexically fixed the pair is, thus . In Equation 2, and are the mean and standard deviation of the following sample: .

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 [2]. 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.

Equation 3 - Syntactic Behaviour of Typical Verb-Noun Pair in [2]

For the target pairs, the syntactic behaviour is defined as the posterior probability distribution over the patterns given the pair, as shown in Equation 4.

Equation 4 - Syntactic Behaviour of Target Verb-Noun Pair in [2]

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 .

Equation 5 - Degree of Syntactic Fixedness for a Target Verb-Noun pair in [2]

[2] hypothesizes 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 .

Equation 6 - Overall Fixedness for Target Verb-Noun pair in [2]

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.

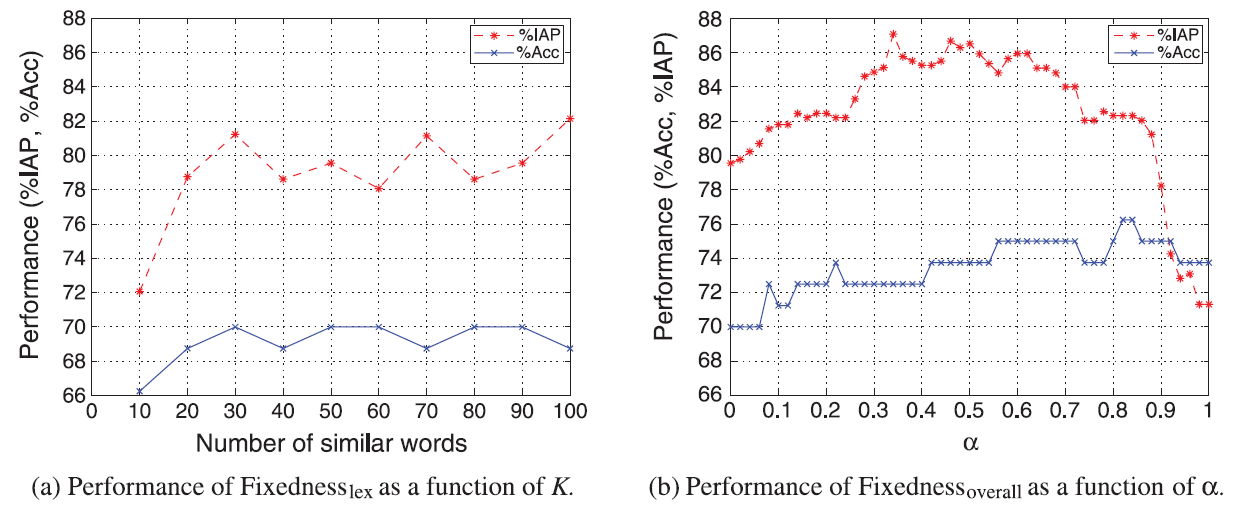


Figure 1 - %IAP and %Acc of Fixedness\_lex and Fixedness\_overall over development data [2]

### 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”) [2] [8]. An unsupervised method by [2] aims to find this set of Canonical Forms of a target pair with the following equation:

Equation 7 - Algorithm to find set of Canonical Forms by [2]

Where is the set of possible verb/noun patterns defined in Table 1, is the static z-score in Equation 8, and is a predefined threshold.

Equation 8 - z-score for Pattern counts by [2]

Where is the frequency of a given verb/noun pair appearing in a particular pattern (), is the sample mean, and the sample standard deviation.

## 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 [1] [6].

Recent research makes use of classifier models such as k-Nearest Neighbours (k-NNs) SVM with linear and polynomial kernels, since these algorithms have been proven to work for binary classification problems [1] [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 [1] [9] [6].

One approach tries to train a Linear SVM Classifier with data obtained by using **Word Embeddings** from Word2Vec [12]. This implementation by [9] proposes the creation of two vectors that contain both the representation of the VNC and its context, called and respectively. is created by averaging the word embedding vectors of the lemmatized component words of the VNC, while is the averaging of two other vectors: and , that represent the context of the verb and noun components respectively [9]. Once and 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 [2]. 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 [2].

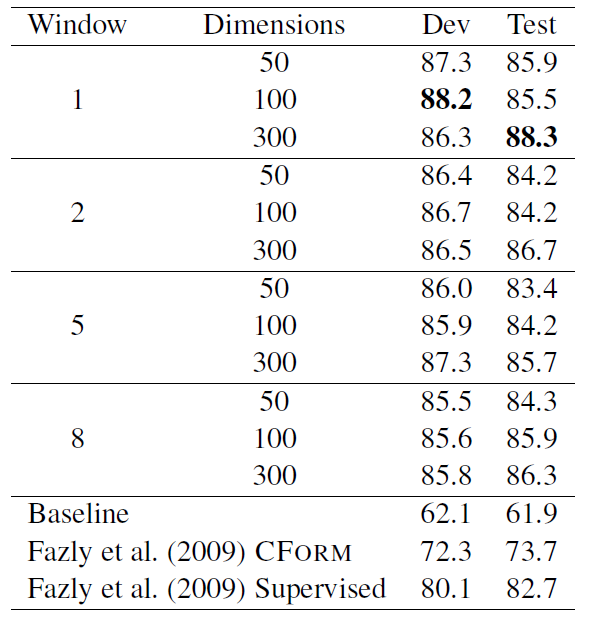


Table 2 - Accuracy Score for supervised word2vec approach by [9]

Other such approach is that of **Skip-Thoughts Vectors** (Sent2Vec) [13], used originally by [6] and then used as a base of comparison by [1]. 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 [13]. This results in an encoder that can product highly generic sentence representations [13]. Sent2Vec was first used for idiom detection by [6] on the assumption that in a real-world application, target phrases won’t have access to a surrounding context; 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] [13], which allows the classifier to learn lexical and syntactic patterns without complex methods. [6] uses the resulting encodings to train three SVM classifiers with the VNC-Tokens Dataset [14]: 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.

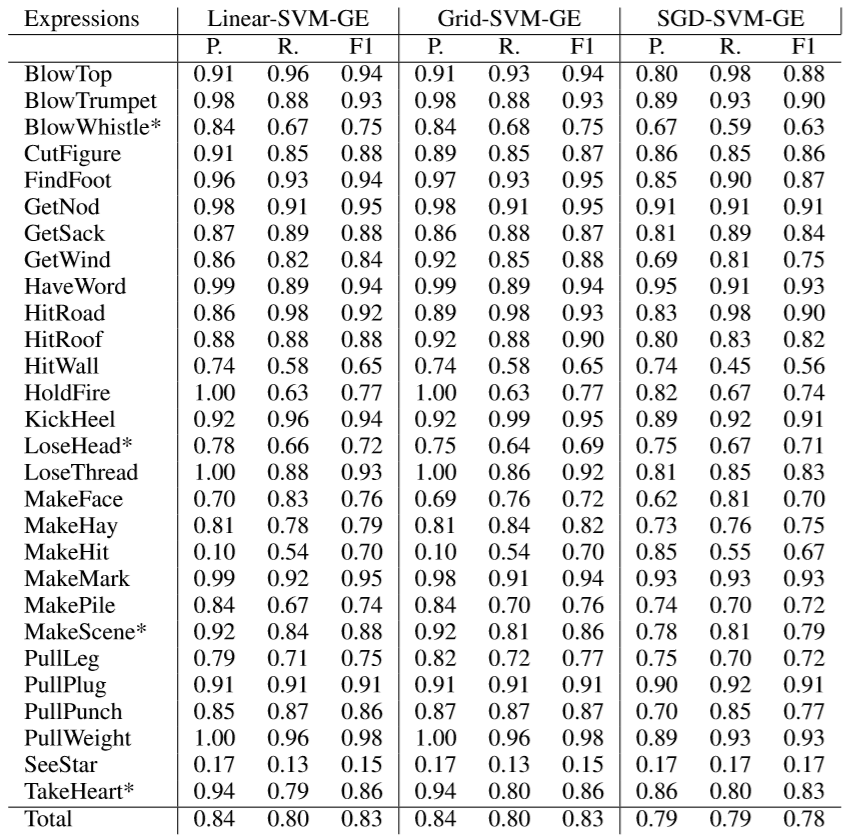


Table 3 - Precision (P.), Recall (R.), and F1-Score (F1) results on Generic Classifiers by [6]

The last studied research for the supervised learning portion of this project is that developed by [1], which also used a Linear SVM kernel but experiments with three different feature encodings for the VNCs and their context. First, they use Word2Vec’s **Skip-Gram** model[12], similarly to the approach taken by [9]; however, instead of encoding the VNC and context in different vectors and then subtracting them, [1]’s implementation averages the normalized word embeddings for each word in the sentences containing a target VNC. 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” [1]; 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 [2] 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.

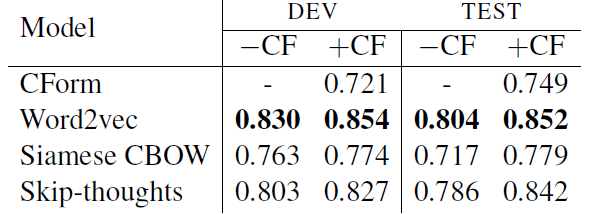


Table 4 - Accuracy Score for supervised word2vec, siamese cbow and skip-thoughts approaches by [1]

## 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) by [15] and **Continual Multiplication of Words** (CMOW) embeddings to evaluate performance.

## ELMo

Equation 9 - ELMo Layer Computation by [15]

**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 in Equation 9, where is the token layer [15]. Each biLM consist of a Forward and a Backward Linear Model , each one computing a context-independent representation of the target token, repeated L times in Equation 9 [15]. The output of these biLMs goes through the ELMo layer, which passes its enhanced representations to the task in hand. are softmax-normalized weights and the scalar parameter allows the task model to scale the entire ELMo vector [15]. According to [15], 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 [2]. 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.

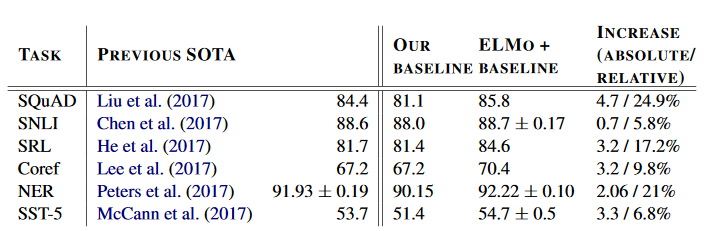


Table 5 - Test set comparison of ELMo enhanced neural models with state-of-the-art model baselines across six benchmark NLP tasks by [15]

## CMOW

[https://arxiv.org/pdf/1902.06423.pdf]

## Further Unsupervised Methods

[2]’s success on discriminating idiomatic use of VNCs with their metrics plus the high performance of SVMs with linear kernels have demonstrated that word vector representations of target sentences can be linearly separated. 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 embbedings based on previous research using this tool by [9].

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) [16]. 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 [16]. This can be better observed in Figure 2, where can be any distance function:

Figure 2 - k-Means Algorithm from [16]

**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 [17]) 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.

PMI, Lexical Fixedness, Syntactical Fixedness, and Overall Fixedness for unsupervised Candidate extraction.

Cosine Similarity for idiomatic use of VNC [18] [19].

# Methods

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

## Fixedness Metrics [Done]

These metrics were implemented as described previously in Section 3.1, based on the work by [2].

### Verb-Noun Combination Pattern Counts [Done]

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 [20] (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.

|  |  |  |
| --- | --- | --- |
| 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 |  |
|  | 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 |  |
|  | VBN, VDN, VHN, VVN, VVN-AJ0, VVN-VVD |  |
|  | NN0, NN1, NP0, NN1-AJ0, NN1-NP0, NN1-VVB, NN1-VVG, NP0-NN1, VVB-NN1, AJ0-NN1 |  |
|  | 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.

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

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 [2]’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.

### Pointwise Mutual Information [Done]

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 :

* : Total number of captured VNCs (in this case, the length of the dictionary described earlier in this section).
* : Frequency of the target VNC (in this case, the position 0 of the stored list).
* : Total frequency of in all extracted patterns (in this case, the sum of all pattern frequencies in which appears).
* : Total frequency of in all extracted patterns (in this case, the sum of all pattern frequencies in which appears).

We use log base 2 for all calculations.

### Lexical Fixedness [Done]

The implementation of the Lexical Fixedness metric follows [2]’s description observed in Equation 2.

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: , in which , being and the number of the most similar verbs and nouns to those in the VNC respectively.

To get the most similar verbs and the most similar nouns, we use one of two options: Lin’s Thesaurus following implementation by [2] 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
* Use Lin’s Thesaurus or WordNet+Word2Vec for similar Verb/Nouns

### Syntactic Fixedness [Done]

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:

* : Accumulated frequency for a given pattern, that is a given index in the stored list, across all keys in our VNC dictionary.
* : 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:

* : Index 1-11 from the list of pattern counts given a key (VNC).
* : 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 [Done]

As described by [2], 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 , being the larger the value, the higher the weight of Syntactical Fixedness at viceversa.

As described, this equation requires the following hyperparameters:

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

* Word Window in Pattern Counts
* Use Lin’s Thesaurus or WordNet+Word2Vec for similar Verb/Nouns

### Canonical Forms [Done]

To find the Canonical Forms of the VNCs in our project, we implement the Equation 7 and Equation 8 by [2].

For Equation 8, we use the stored frequencies from out Verb-Noun Pattern counts:

* : Index 1-11 of the list of pattern counts given a key (VNC).
* : Mean value stored in indexes 1-11 from the list of pattern counts given a key.
* : 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 .

Our described implementation is affected by the following hyperparameters:

* Word Window in Pattern Counts

## Potential VNICs Extraction [Done]

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.

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 poor candidates such as: .

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 [14] 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.

## Word Embeddings

Word Embeddings are used for experiments following research by [1] [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*”.

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

### Word2Vec

Implementation of Word2Vec [12] model was made with the publicly available library **gensim**. Training was performed using the available BNC XML Corpora, with the parameters used by [1]:

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

Since this model does not offer sentence embeddings, we follow [1]’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 (<https://bitbucket.org/TomKenter/siamese-cbow/src/master/>). We used a pre-trained model trained on the Wikipedia (INEX) Corpus, following the approach by [1]. 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 [13] model was implemented using the code made available in Kiros’ public github repository (<https://github.com/ryankiros/skip-thoughts>). We use the available Comb-Skip model, which is a concatenation of the Uni-Skip and Bi-Skip models trained using the BookCorpus [21], similar to the approaches by [1] 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 [15] embeddings were implemented using the TesorFlow Hub available model (<https://tfhub.dev/google/elmo/2>) 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.

### CMOW

[https://github.com/florianmai/word2mat]

## Supervised Idiomatic Use Detection

Supervised classification of idiomatic vs non-idiomatic use is performed using a Support Vector Machine following experiments by [1], [6], and [9].

### SVM Classification

For our SVM Classifier, we used the Scikit-Learn’s SVC packet (<https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>) [22]. 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.

## Unsupervised Idiomatic Use Detection

Unsupervised techniques include k-Means clustering following experiments by [9] and the proposed Cosine Similarity metric between the target VNC and the sentence embeddings following previous use of this score in previous related research by [18] and [19].

### k-Means Clustering

For the k-Means algorithm, we also used Scikit-Learn’s implementation on the KMeans packet (<https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html>) [22]. This implementation was selected since it allows for quick parameter tuning for a different number of initial centroids.

### Cosine Similarity

The Cosine Similarity metric was implemented by calculating the EQUATION FROM LIT REVIEW. The dot product and vector norms where obtained using Numpy.

# Apparatus

This sections 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.

## Dataset

### BNC XML Corpora

The BNC XML Corpora [20] 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 [14] 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 [1], [2], 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.

## Experimental Procedure

1. Train Unlemmatized and Lemmatized Word Embeddings (Only Word2Vec, all other models pre-trained)
2. Extract Fazly’s VNC Pattern Counts from the Corpora.
   1. Window Size
3. Sort VNICs Candidates based on the PMI, Lexical Fixedness, Syntactical Fixedness, and Overall Fixedness.
4. Extract all instances of the top candidates for each metric ranking. Store them in a similar format to that of VNC-Tokens Dataset.
   1. Top Candidates
   2. Window Size
   3. Maximum Sentence Length (for ELMo)
   4. Lexical Fixedness:
      1. K
      2. Use Lin
   5. Overall Fixedness:
      1. K
      2. Use Lin
      3. Alpha
5. Extract all unlemmatized and lemmatized instances of the candidate VNICs and the metrics:
   1. Lexical Fixedness:
      1. K
      2. Use Lin
   2. Overall Fixedness:
      1. K
      2. Use Lin
      3. Alpha
6. Generate Embeddings for the VNC-Tokens and VNC-Candidates datasets (unlemmatized and lemmatized).
7. EXPERIMENTS 1:
8. TRAIN SVMS
9. EXPERIMENTS 2
10. EXPERIMENTS 3

# - VIRTUAL EVIRONMENT SETUP - #

virtualenv -p python36 venv && source venv/bin/activate

pip3 install -r requirements.txt

################################

echo "Training Embeddings"

# Train SVMs - Unlemmatized + CForms + Fixedness

python36 TrainSVMs.py --VECTORS\_FILE "embeddings.csv" --MODELS\_DIR "./SVM\_Models/" --MODEL\_EXT "\_clean" \

--C\_W2V 0.1 --C\_SKIP 1 --C\_SCBOW 1 --C\_ELMo 1

echo "Training Embeddings + CForms"

# Train SVMs - Unlemmatized + CForms + Fixedness

python36 TrainSVMs.py --VECTORS\_FILE "embeddings.csv" --MODELS\_DIR "./SVM\_Models/" --MODEL\_EXT "\_cForms" \

--USE\_CFORM \

--C\_W2V 0.1 --C\_SKIP 1 --C\_SCBOW 1 --C\_ELMo 1

echo "Training Embeddings + CForms + Fixedness"

# Train SVMs - Unlemmatized + CForms + Fixedness

python36 TrainSVMs.py --VECTORS\_FILE "embeddings.csv" --MODELS\_DIR "./SVM\_Models/" --MODEL\_EXT "\_cForms\_Fix" \

--USE\_CFORM --USE\_SYN\_FIX --USE\_LEX\_FIX --USE\_OVA\_FIX \

--C\_W2V 0.1 --C\_SKIP 1 --C\_SCBOW 1 --C\_ELMo 1

echo "Training Embeddings Lemmatized"

# Train SVMs - Lemmatized + CForms + Fixedness

python36 TrainSVMs.py --VECTORS\_FILE "embeddings\_lemm.csv" --MODELS\_DIR "./SVM\_Models/" --MODEL\_EXT "\_lemm" \

--C\_W2V 0.1 --C\_SKIP 1 --C\_SCBOW 1 --C\_ELMo 1

echo "Training Embeddings Lemmatized + CForms"

# Train SVMs - Unlemmatized + CForms + Fixedness

python36 TrainSVMs.py --VECTORS\_FILE "embeddings\_lemm.csv" --MODELS\_DIR "./SVM\_Models/" --MODEL\_EXT "\_lemm\_cForms" \

--USE\_CFORM \

--C\_W2V 0.1 --C\_SKIP 1 --C\_SCBOW 1 --C\_ELMo 1

echo "Training Embeddings Lemmatized + CForms + Fixedness"

# Train SVMs - Lemmatized + CForms + Fixedness

python36 TrainSVMs.py --VECTORS\_FILE "embeddings\_lemm.csv" --MODELS\_DIR "./SVM\_Models/" --MODEL\_EXT "\_lemm\_cForms\_Fix" \

--USE\_CFORM --USE\_SYN\_FIX --USE\_LEX\_FIX --USE\_OVA\_FIX \

--C\_W2V 0.1 --C\_SKIP 1 --C\_SCBOW 1 --C\_ELMo 1

deactivate

#==========================================================#

#----------------------------------------------------------#

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_#

echo "RunExperiments.sh"

# ============================================================ #

# ============================================================ #

# Experiments 1

cd Experiments\_1

# - VIRTUAL EVIRONMENT SETUP - #

virtualenv -p python36 venv && source venv/bin/activate

pip3 install -r requirements.txt

################################

echo "Experiments 1"

# King and Cook Experiments - SVM - Embeddings + CForm

echo "Experient 1-1 : King and Cook Experiments - SVM - Embeddings + CForm"

python36 Experiment.py

deactivate

cd ..

# ============================================================ #

# ============================================================ #

# Experiments 2

cd Experiments\_2

# - VIRTUAL EVIRONMENT SETUP - #

virtualenv -p python36 venv && source venv/bin/activate

pip3 install -r requirements.txt

################################

echo "Experiments 2"

# SVM - CForm + Fazly's Fixedness Metrics

echo "Experient 2-1 : SVM - CForm + Fazly's Fixedness Metrics"

python36 Experiment\_2\_1.py

# King and Cook Experiments - SVM - Embeddings + CForm + Fazly's Fixedness Metrics

echo "Experient 2-2 : King and Cook Experiments - SVM - Embeddings"

python36 Experiment\_2\_2.py --VECTORS\_FILE "embeddings.csv" --RESULTS\_DIR "./results/Experiment\_2\_2/Clean/"

echo "Experient 2-2 : King and Cook Experiments - SVM - Embeddings + CForm"

python36 Experiment\_2\_2.py --VECTORS\_FILE "embeddings.csv" --RESULTS\_DIR "./results/Experiment\_2\_2/CForm/" \

--USE\_CFORM

echo "Experient 2-2 : King and Cook Experiments - SVM - Embeddings + CForm + Fazly's Fixedness Metrics"

python36 Experiment\_2\_2.py -VECTORS\_FILE "embeddings.csv" --RESULTS\_DIR "./results/Experiment\_2\_2/CForm\_Fix/" \

--USE\_CFORM --USE\_SYN\_FIX --USE\_LEX\_FIX --USE\_OVA\_FIX

echo "Experient 2-2 : King and Cook Experiments - SVM - Embeddings Lemmatized"

python36 Experiment\_2\_2.py --VECTORS\_FILE "embeddings\_lemm.csv" --RESULTS\_DIR "./results/Experiment\_2\_2/Lemm/"

echo "Experient 2-2 : King and Cook Experiments - SVM - Embeddings Lemmatized + CForm"

python36 Experiment\_2\_2.py -VECTORS\_FILE "embeddings\_lemm.csv" --RESULTS\_DIR "./results/Experiment\_2\_2/CForm\_Lemm/" \

--USE\_CFORM

echo "Experient 2-2 : King and Cook Experiments - SVM - Embeddings Lemmatized + CForm + Fazly's Fixedness Metrics"

python36 Experiment\_2\_2.py -VECTORS\_FILE "embeddings\_lemm.csv" --RESULTS\_DIR "./results/Experiment\_2\_2/CForm\_Fix\_Lemm/" \

--USE\_CFORM --USE\_SYN\_FIX --USE\_LEX\_FIX --USE\_OVA\_FIX

# Clustering Proposal - k-Means - Embeddings + CForm + Fazly's Fixedness Metrics

echo "Experient 2-3 : Clustering Proposal - k-Means - Embeddings + CForm + Fazly's Fixedness Metrics"

python36 Experiment\_2\_3.py --VECTORS\_FILE "embeddings.csv" --RESULTS\_DIR "./results/Experiment\_2\_3/Clean/" \

--USE\_CFORM --USE\_SYN\_FIX --USE\_LEX\_FIX --USE\_OVA\_FIX \

--SAVE\_PLOT

echo "Experient 2-3 : Clustering Proposal - k-Means - Lemmatized Embeddings + CForm + Fazly's Fixedness Metrics"

python36 Experiment\_2\_3.py --VECTORS\_FILE "embeddings\_lemm.csv" --RESULTS\_DIR "./results/Experiment\_2\_3/Lemm/" \

--USE\_CFORM --USE\_SYN\_FIX --USE\_LEX\_FIX --USE\_OVA\_FIX \

--SAVE\_PLOT

# Unsupervised - Cosine Similarity

echo "Experient 2-4 : Unsupervised - Cosine Similarity - Embeddings"

python36 Experiment\_2\_4.py --VECTORS\_FILE "embeddings.csv" --VECTORS\_FILE\_VNC "embeddings\_VNC.csv" --RESULTS\_DIR "./results/Experiment\_2\_4/Clean/" \

--COS\_DIST\_T 0.6

echo "Experient 2-4 : Unsupervised - Cosine Similarity - Embeddings"

python36 Experiment\_2\_4.py --VECTORS\_FILE "embeddings\_lemm.csv" --VECTORS\_FILE\_VNC "embeddings\_VNC\_lemm.csv" --RESULTS\_DIR "./results/Experiment\_2\_4/Lemm/" \

--COS\_DIST\_T 0.6

# Unsupervised - New Metrics -> Cosine Similarity and Overall Fixedness

echo "Experient 2-5 : Unsupervised - New Metrics - Embeddings -> Cosine Similarity and Overall Fixedness"

python36 Experiment\_2\_5.py --VECTORS\_FILE "embeddings.csv" --VECTORS\_FILE\_VNC "embeddings\_VNC.csv" --RESULTS\_DIR "./results/Experiment\_2\_5/Clean/" \

--UNM\_MET\_T 0.7 --BETA 0.6

echo "Experient 2-5 : Unsupervised - New Metrics - Lemmatized Embeddings -> Cosine Similarity and Overall Fixedness"

python36 Experiment\_2\_5.py --VECTORS\_FILE "embeddings\_lemm.csv" --VECTORS\_FILE\_VNC "embeddings\_VNC\_lemm.csv" --RESULTS\_DIR "./results/Experiment\_2\_5/Lemm/" \

--UNM\_MET\_T 0.7 --BETA 0.6

echo "Experient 2-5\_b : Unsupervised - New Metrics - Embeddings -> Cosine Similarity and Overall Fixedness"

python36 Experiment\_2\_5\_b.py --VECTORS\_FILE "embeddings.csv" --VECTORS\_FILE\_VNC "embeddings\_VNC.csv" --RESULTS\_DIR "./results/Experiment\_2\_5\_b/Clean/"

echo "Experient 2-5\_b : Unsupervised - New Metrics - Lemmatized Embeddings -> Cosine Similarity and Overall Fixedness"

python36 Experiment\_2\_5\_b.py --VECTORS\_FILE "embeddings\_lemm.csv" --VECTORS\_FILE\_VNC "embeddings\_VNC\_lemm.csv" --RESULTS\_DIR "./results/Experiment\_2\_5\_b/Lemm/"

deactivate

cd ..

# ============================================================ #

# ============================================================ #

# Experiments 3

cd Experiments\_3

# - VIRTUAL EVIRONMENT SETUP - #

virtualenv -p python36 venv && source venv/bin/activate

pip3 install -r requirements.txt

################################

echo "Experiments 3"

# Unsupervised - Cosine Similarity [Candidates]

echo "Experient 3-1 : Unsupervised - Cosine Similarity [Candidates] - PMI Candidates"

python36 Experiment\_3\_1.py --VECTORS\_FILE "embeddings\_cand\_PMI.csv" --VECTORS\_FILE\_VNC "embeddings\_cand\_PMI\_VNC.csv" --RESULTS\_DIR "./results/Experiment\_3\_1/PMI\_Clean/" \

--SVM\_W2V "../SVM\_Models/W2V\_cForms.model" \

--SVM\_SCBOW "../SVM\_Models/SCBOW\_cForms.model" \

--SVM\_SKIP "../SVM\_Models/SKIP\_cForms.model" \

--SVM\_ELMO "../SVM\_Models/ELMO\_cForms.model" \

--USE\_CFORM \

--CFORM\_DIR "../targets/CForms\_cand\_PMI.csv" \

--COS\_DIST\_T 0.6

echo "Experient 3-1 : Unsupervised - Cosine Similarity [Candidates] - PMI Candidates - Lemmatized"

python36 Experiment\_3\_1.py --VECTORS\_FILE "embeddings\_cand\_PMI\_lemm.csv" --VECTORS\_FILE\_VNC "embeddings\_cand\_PMI\_VNC\_lemm.csv" --RESULTS\_DIR "./results/Experiment\_3\_1/PMI\_Lemm/"\

--SVM\_W2V "../SVM\_Models/W2V\_lemm\_cForms\_.model" \

--SVM\_SCBOW "../SVM\_Models/SCBOW\_lemm\_cForms\_.model" \

--SVM\_SKIP "../SVM\_Models/SKIP\_lemm\_cForms\_.model" \

--SVM\_ELMO "../SVM\_Models/ELMO\_lemm\_cForms\_.model" \

--USE\_CFORM \

--CFORM\_DIR "../targets/CForms\_cand\_PMI.csv" \

--COS\_DIST\_T 0.6

echo "Experient 3-1 : Unsupervised - Cosine Similarity [Candidates] - Lexical Fixedness Candidates"

python36 Experiment\_3\_1.py --VECTORS\_FILE "embeddings\_cand\_LEX.csv" --VECTORS\_FILE\_VNC "embeddings\_cand\_LEX\_VNC.csv" --RESULTS\_DIR "./results/Experiment\_3\_1/LEX\_Clean/" \

--SVM\_W2V "../SVM\_Models/W2V\_cForms.model" \

--SVM\_SCBOW "../SVM\_Models/SCBOW\_cForms.model" \

--SVM\_SKIP "../SVM\_Models/SKIP\_cForms.model" \

--SVM\_ELMO "../SVM\_Models/ELMO\_cForms.model" \

--USE\_CFORM \

--CFORM\_DIR "../targets/CForms\_cand\_LEX.csv" \

--COS\_DIST\_T 0.6

echo "Experient 3-1 : Unsupervised - Cosine Similarity [Candidates] - Lexical Fixedness Candidates - Lemmatized"

python36 Experiment\_3\_1.py --VECTORS\_FILE "embeddings\_cand\_LEX\_lemm.csv" --VECTORS\_FILE\_VNC "embeddings\_cand\_LEX\_VNC\_lemm.csv" --RESULTS\_DIR "./results/Experiment\_3\_1/LEX\_Lemm/"\

--SVM\_W2V "../SVM\_Models/W2V\_lemm\_cForms\_.model" \

--SVM\_SCBOW "../SVM\_Models/SCBOW\_lemm\_cForms\_.model" \

--SVM\_SKIP "../SVM\_Models/SKIP\_lemm\_cForms\_.model" \

--SVM\_ELMO "../SVM\_Models/ELMO\_lemm\_cForms\_.model" \

--USE\_CFORM \

--CFORM\_DIR "../targets/CForms\_cand\_LEX.csv" \

--COS\_DIST\_T 0.6

echo "Experient 3-1 : Unsupervised - Cosine Similarity [Candidates] - Syntactical Fixedness Candidates"

python36 Experiment\_3\_1.py --VECTORS\_FILE "embeddings\_cand\_SYN.csv" --VECTORS\_FILE\_VNC "embeddings\_cand\_SYN\_VNC.csv" --RESULTS\_DIR "./results/Experiment\_3\_1/SYN\_Clean/" \

--SVM\_W2V "../SVM\_Models/W2V\_cForms.model" \

--SVM\_SCBOW "../SVM\_Models/SCBOW\_cForms.model" \

--SVM\_SKIP "../SVM\_Models/SKIP\_cForms.model" \

--SVM\_ELMO "../SVM\_Models/ELMO\_cForms.model" \

--USE\_CFORM \

--CFORM\_DIR "../targets/CForms\_cand\_SYN.csv" \

--COS\_DIST\_T 0.6

echo "Experient 3-1 : Unsupervised - Cosine Similarity [Candidates] - Syntactical Fixedness Candidates - Lemmatized"

python36 Experiment\_3\_1.py --VECTORS\_FILE "embeddings\_cand\_SYN\_lemm.csv" --VECTORS\_FILE\_VNC "embeddings\_cand\_SYN\_VNC\_lemm.csv" --RESULTS\_DIR "./results/Experiment\_3\_1/SYN\_Lemm/"\

--SVM\_W2V "../SVM\_Models/W2V\_lemm\_cForms\_.model" \

--SVM\_SCBOW "../SVM\_Models/SCBOW\_lemm\_cForms\_.model" \

--SVM\_SKIP "../SVM\_Models/SKIP\_lemm\_cForms\_.model" \

--SVM\_ELMO "../SVM\_Models/ELMO\_lemm\_cForms\_.model" \

--USE\_CFORM \

--CFORM\_DIR "../targets/CForms\_cand\_SYN.csv" \

--COS\_DIST\_T 0.6

echo "Experient 3-1 : Unsupervised - Cosine Similarity [Candidates] - Overall Fixedness Candidates"

python36 Experiment\_3\_1.py --VECTORS\_FILE "embeddings\_cand\_OVA.csv" --VECTORS\_FILE\_VNC "embeddings\_cand\_OVA\_VNC.csv" --RESULTS\_DIR "./results/Experiment\_3\_1/OVA\_Clean/" \

--SVM\_W2V "../SVM\_Models/W2V\_cForms.model" \

--SVM\_SCBOW "../SVM\_Models/SCBOW\_cForms.model" \

--SVM\_SKIP "../SVM\_Models/SKIP\_cForms.model" \

--SVM\_ELMO "../SVM\_Models/ELMO\_cForms.model" \

--USE\_CFORM \

--CFORM\_DIR "../targets/CForms\_cand\_OVA.csv" \

--COS\_DIST\_T 0.6

echo "Experient 3-1 : Unsupervised - Cosine Similarity [Candidates] - Overall Fixedness Candidates - Lemmatized"

python36 Experiment\_3\_1.py --VECTORS\_FILE "embeddings\_cand\_OVA\_lemm.csv" --VECTORS\_FILE\_VNC "embeddings\_cand\_OVA\_VNC\_lemm.csv" --RESULTS\_DIR "./results/Experiment\_3\_1/OVA\_Lemm/"\

--SVM\_W2V "../SVM\_Models/W2V\_lemm\_cForms\_.model" \

--SVM\_SCBOW "../SVM\_Models/SCBOW\_lemm\_cForms\_.model" \

--SVM\_SKIP "../SVM\_Models/SKIP\_lemm\_cForms\_.model" \

--SVM\_ELMO "../SVM\_Models/ELMO\_lemm\_cForms\_.model" \

--USE\_CFORM \

--CFORM\_DIR "../targets/CForms\_cand\_OVA.csv" \

--COS\_DIST\_T 0.6

# Unsupervised - New Metrics -> Cosine Similarity and Overall Fixedness [Candidates]

echo "Experient 3-2 : Unsupervised - New Metrics -> Cosine Similarity and Overall Fixedness [Candidates] - PMI Candidates"

python36 Experiment\_3\_2.py --VECTORS\_FILE "embeddings\_cand\_PMI.csv" --VECTORS\_FILE\_VNC "embeddings\_cand\_PMI\_VNC.csv" --RESULTS\_DIR "./results/Experiment\_3\_2/PMI\_Clean/" \

--SVM\_W2V "../SVM\_Models/W2V\_cForms.model" \

--SVM\_SCBOW "../SVM\_Models/SCBOW\_cForms.model" \

--SVM\_SKIP "../SVM\_Models/SKIP\_cForms.model" \

--SVM\_ELMO "../SVM\_Models/ELMO\_cForms.model" \

--USE\_CFORM \

--CFORM\_DIR "../targets/CForms\_cand\_PMI.csv" \

--UNM\_MET\_T 0.4 --BETA 0.6

echo "Experient 3-2 : Unsupervised - New Metrics -> Cosine Similarity and Overall Fixedness [Candidates] - PMI Candidates - Lemmatized"

python36 Experiment\_3\_2.py --VECTORS\_FILE "embeddings\_cand\_PMI\_lemm.csv" --VECTORS\_FILE\_VNC "embeddings\_cand\_PMI\_VNC\_lemm.csv" --RESULTS\_DIR "./results/Experiment\_3\_2/PMI\_Lemm/"\

--SVM\_W2V "../SVM\_Models/W2V\_lemm\_cForms\_.model" \

--SVM\_SCBOW "../SVM\_Models/SCBOW\_lemm\_cForms\_.model" \

--SVM\_SKIP "../SVM\_Models/SKIP\_lemm\_cForms\_.model" \

--SVM\_ELMO "../SVM\_Models/ELMO\_lemm\_cForms\_.model" \

--USE\_CFORM \

--CFORM\_DIR "../targets/CForms\_cand\_PMI.csv" \

--UNM\_MET\_T 0.4 --BETA 0.6

echo "Experient 3-2 : Unsupervised - New Metrics -> Cosine Similarity and Overall Fixedness [Candidates] - Lexical Fixedness Candidates"

python36 Experiment\_3\_2.py --VECTORS\_FILE "embeddings\_cand\_LEX.csv" --VECTORS\_FILE\_VNC "embeddings\_cand\_LEX\_VNC.csv" --RESULTS\_DIR "./results/Experiment\_3\_2/LEX\_Clean/" \

--SVM\_W2V "../SVM\_Models/W2V\_cForms.model" \

--SVM\_SCBOW "../SVM\_Models/SCBOW\_cForms.model" \

--SVM\_SKIP "../SVM\_Models/SKIP\_cForms.model" \

--SVM\_ELMO "../SVM\_Models/ELMO\_cForms.model" \

--USE\_CFORM \

--CFORM\_DIR "../targets/CForms\_cand\_LEX.csv" \

--UNM\_MET\_T 0.4 --BETA 0.6

echo "Experient 3-2 : Unsupervised - New Metrics -> Cosine Similarity and Overall Fixedness [Candidates] - Lexical Fixedness Candidates - Lemmatized"

python36 Experiment\_3\_2.py --VECTORS\_FILE "embeddings\_cand\_LEX\_lemm.csv" --VECTORS\_FILE\_VNC "embeddings\_cand\_LEX\_VNC\_lemm.csv" --RESULTS\_DIR "./results/Experiment\_3\_2/LEX\_Lemm/"\

--SVM\_W2V "../SVM\_Models/W2V\_lemm\_cForms\_.model" \

--SVM\_SCBOW "../SVM\_Models/SCBOW\_lemm\_cForms\_.model" \

--SVM\_SKIP "../SVM\_Models/SKIP\_lemm\_cForms\_.model" \

--SVM\_ELMO "../SVM\_Models/ELMO\_lemm\_cForms\_.model" \

--USE\_CFORM \

--CFORM\_DIR "../targets/CForms\_cand\_LEX.csv" \

--UNM\_MET\_T 0.4 --BETA 0.6

echo "Experient 3-2 : Unsupervised - New Metrics -> Cosine Similarity and Overall Fixedness [Candidates] - Syntactical Fixedness Candidates"

python36 Experiment\_3\_2.py --VECTORS\_FILE "embeddings\_cand\_LEX.csv" --VECTORS\_FILE\_VNC "embeddings\_cand\_LEX\_VNC.csv" --RESULTS\_DIR "./results/Experiment\_3\_2/LEX\_Clean/" \

--SVM\_W2V "../SVM\_Models/W2V\_cForms.model" \

--SVM\_SCBOW "../SVM\_Models/SCBOW\_cForms.model" \

--SVM\_SKIP "../SVM\_Models/SKIP\_cForms.model" \

--SVM\_ELMO "../SVM\_Models/ELMO\_cForms.model" \

--USE\_CFORM \

--CFORM\_DIR "../targets/CForms\_cand\_SYN.csv" \

--UNM\_MET\_T 0.4 --BETA 0.6

echo "Experient 3-2 : Unsupervised - New Metrics -> Cosine Similarity and Overall Fixedness [Candidates] - Syntactical Fixedness Candidates - Lemmatized"

python36 Experiment\_3\_2.py --VECTORS\_FILE "embeddings\_cand\_LEX\_lemm.csv" --VECTORS\_FILE\_VNC "embeddings\_cand\_LEX\_VNC\_lemm.csv" --RESULTS\_DIR "./results/Experiment\_3\_2/LEX\_Lemm/"\

--SVM\_W2V "../SVM\_Models/W2V\_lemm\_cForms\_.model" \

--SVM\_SCBOW "../SVM\_Models/SCBOW\_lemm\_cForms\_.model" \

--SVM\_SKIP "../SVM\_Models/SKIP\_lemm\_cForms\_.model" \

--SVM\_ELMO "../SVM\_Models/ELMO\_lemm\_cForms\_.model" \

--USE\_CFORM \

--CFORM\_DIR "../targets/CForms\_cand\_SYN.csv" \

--UNM\_MET\_T 0.4 --BETA 0.6

echo "Experient 3-2 : Unsupervised - New Metrics -> Cosine Similarity and Overall Fixedness [Candidates] - Overall Fixedness Candidates"

python36 Experiment\_3\_2.py --VECTORS\_FILE "embeddings\_cand\_LEX.csv" --VECTORS\_FILE\_VNC "embeddings\_cand\_LEX\_VNC.csv" --RESULTS\_DIR "./results/Experiment\_3\_2/LEX\_Clean/" \

--SVM\_W2V "../SVM\_Models/W2V\_cForms.model" \

--SVM\_SCBOW "../SVM\_Models/SCBOW\_cForms.model" \

--SVM\_SKIP "../SVM\_Models/SKIP\_cForms.model" \

--SVM\_ELMO "../SVM\_Models/ELMO\_cForms.model" \

--USE\_CFORM \

--CFORM\_DIR "../targets/CForms\_cand\_OVA.csv" \

--UNM\_MET\_T 0.4 --BETA 0.6

echo "Experient 3-2 : Unsupervised - New Metrics -> Cosine Similarity and Overall Fixedness [Candidates] - Overall Fixedness Candidates - Lemmatized"

python36 Experiment\_3\_2.py --VECTORS\_FILE "embeddings\_cand\_LEX\_lemm.csv" --VECTORS\_FILE\_VNC "embeddings\_cand\_LEX\_VNC\_lemm.csv" --RESULTS\_DIR "./results/Experiment\_3\_2/LEX\_Lemm/"\

--SVM\_W2V "../SVM\_Models/W2V\_lemm\_cForms\_.model" \

--SVM\_SCBOW "../SVM\_Models/SCBOW\_lemm\_cForms\_.model" \

--SVM\_SKIP "../SVM\_Models/SKIP\_lemm\_cForms\_.model" \

--SVM\_ELMO "../SVM\_Models/ELMO\_lemm\_cForms\_.model" \

--USE\_CFORM \

--CFORM\_DIR "../targets/CForms\_cand\_OVA.csv" \

--UNM\_MET\_T 0.4 --BETA 0.6

deactivate

cd ..

# ============================================================ #

# ============================================================ #

## Hyperparameters

## Evaluation Metrics

# Results

# Discussion

# Conclusion

# Future Work

# Appendices

References

|  |  |
| --- | --- |
| [1] | 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. |
| [2] | 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. |
| [3] | 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. |
| [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] | 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. |
| [13] | 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. |
| [14] | P. Cook, A. Fazly and S. Stevenson, *The VNC-Tokens Dataset,* Toronto: University of Toronto, 2008. |
| [15] | 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. |
| [16] | E. Alpaydin, Introduction to Machine Learning, 3rd ed., Cambridge, Massachusetts: Massachusetts Institute of Technology, 2014. |
| [17] | 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. |
| [18] | 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. |
| [19] | 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. |
| [20] | B. Consortium, *The British National Corpus, version 3 (BNC XML Edition),* Oxford: University of Oxford, 2007. |
| [21] | 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. |
| [22] | 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. |
| [23] | 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. |