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: Aline Villavicencio

Date: 22 March 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*

Table of Contents

1 Introduction

Idiomatic use of phrases has always been a problem in the field of Natural Language Processing (NLP) since its appearance in text posse 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.

In recent years, the task of idiomatic detection has focus its interest on the identification of Multiword Expressions (MWEs), which are combinations of multiple words that present some degree of idiosyncrasy in nature, which means that their use has little to no variation in usage due to their inherent fixedness [1] [2]. In particular, there has been interest of MWEs in the form of Verb-Noun Combinations (VNCs), that consist of a single verb with a noun in its direct object position (e.g. <u>Break</u> a <u>leg</u>) [1]. The focus on these types of idiomatic usage is because they are a common type of semantically-idiomatic MWE across several languages. Idiomatic usage is not restricted to VNCs 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 idiomatic VNC 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 <u>break</u> a <u>leg</u>.
2. John fell from the stage and <u>broke</u> a <u>leg</u>.

This has motivated previous research to frame the problem as a supervised binary classification problem [1] [3] [4]. Unsupervised methods have also seemed some degree of success on the past [2] [5], 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 idiomatic usage of VNCs to train new supervised and unsupervised models that may present increased performance for this classification task.

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.

2 Literature Review

Much research on MWE identification focuses on the task of detecting specific kinds on MWEs, such as English VNCs [1] [2] [6], while other authors focus on multilingual detection of MWEs [7]. 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 [4].
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.

2.1 Background

Past research focuses on VNC 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 VNCs 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] demonstrates that idiomatic phrases are not as fixed as literature assumed in the past, since “the corpus data in this chapter show 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”. 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 base form of the idiomatic VNC (i.e. *see stars*), as a startup point for their detection. 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] [4]. 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].

2.2 Unsupervised Methods

Unsupervised models’ approach to detect idiomatic phrases used the assumptions mentioned in Section 2.1.

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).

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

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

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 [2]; 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 [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 $Fixedness_{lex}(v,n) \in [-\infty, +\infty]$. 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 [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.

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

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

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 [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. 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 [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 $Fixedness_{overall}(v,n) \in [0,1]$.

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

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

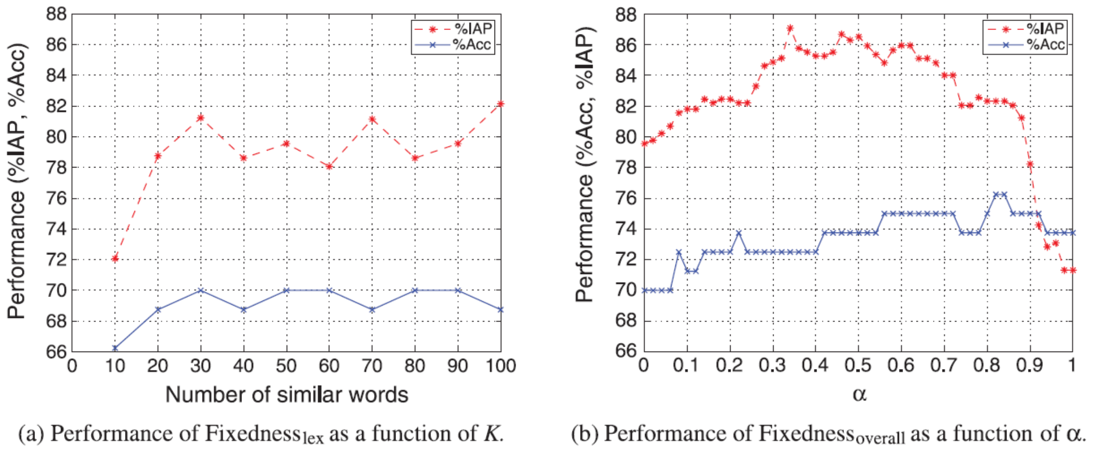


Figure 1 - %IAP and %Acc of Fixedness_lex and Fixedness_overall over development data [2]

2.3 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 [1] [4].

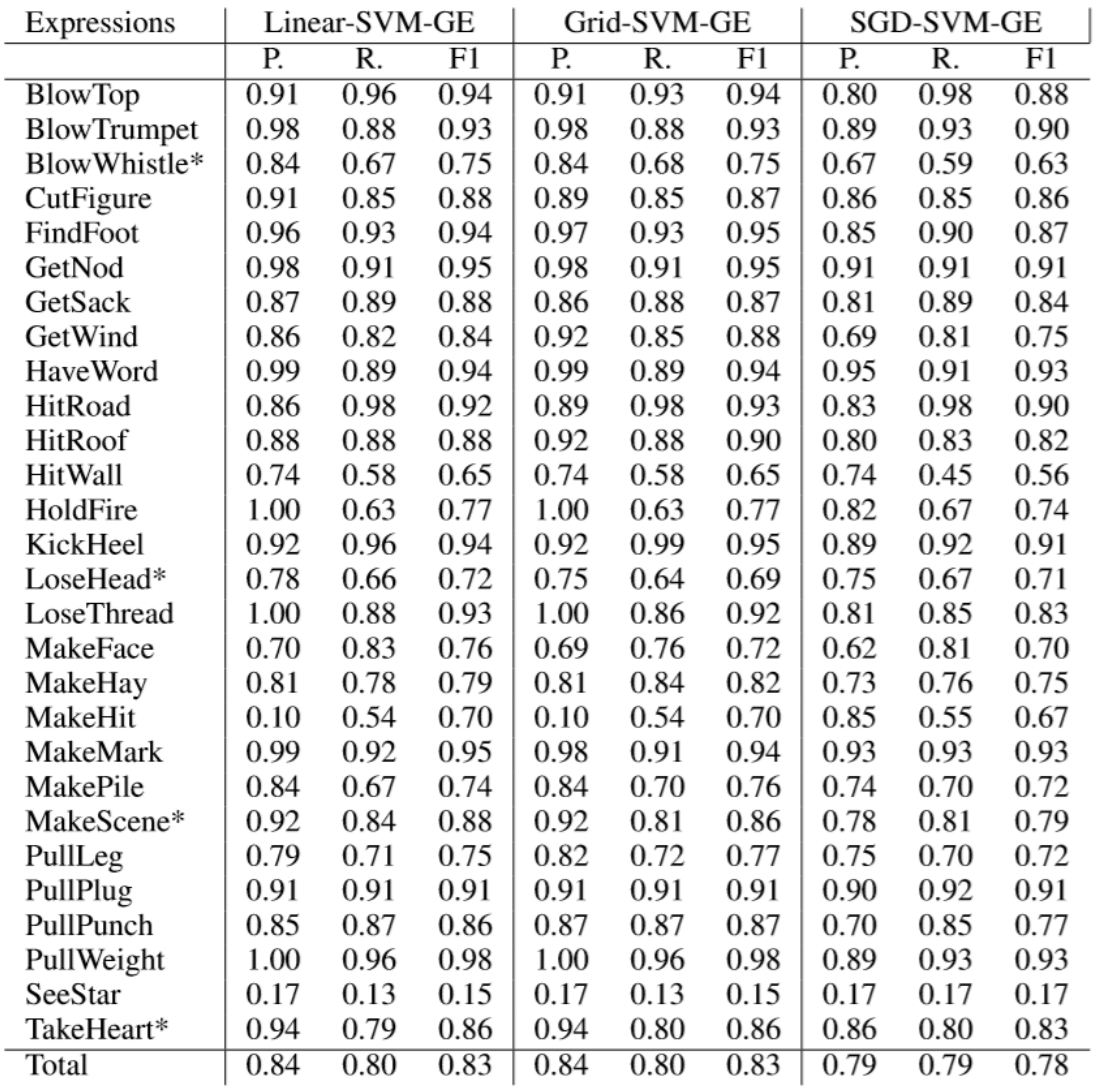
Recent research makes use of classifier models such as k-Nearest Neighbours (k-NNs) SVM with linear and polynomial kernels, since these algorithms have been proven to work for binary classification problems [1] [4]. However, the pre-processing of known VNCs and feature creation is the task in which most research focuses on. In an attempt to exploit the knowledge on lexical and syntactic patterns presented in Section **Error! Reference source not found.**, researchers have made use of unsupervised feature encoders to train the classifiers [1] [6] [4].

One approach tries to train a Linear SVM Classifier with data obtained by using **Word Embeddings** from Word2Vec [9]. This implementation by [6] proposes the creation of two vectors that contain both the representation of the VNC and its context, called $\vec{e}$ and $\vec{c}$ respectively. $\vec{e}$ is created by averaging the word embedding vectors of the lemmatized component words of the VNC, while $\vec{c}$ is the averaging of two other vectors: $\overrightarrow{c_{verb}}$ and $\overrightarrow{c_{noun}}$, that represent the context of the verb and noun components respectively [6]. 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 [2]. The resultant feature vector is then used to train an SVM with linear kernel. The results of this approach are shown on Table 1, comparing them to the models presented in [2].

Window	Dimensions	Dev	Test
	50	87.3	85.9
1	100	**88.2**	85.5
	300	86.3	**88.3**
	50	86.4	84.2
2	100	86.7	84.2
	300	86.5	86.7
	50	86.0	83.4
5	100	85.9	84.2
	300	87.3	85.7
	50	85.5	84.3
8	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 1 - Accuracy Score for supervised word2vec approach by [6]

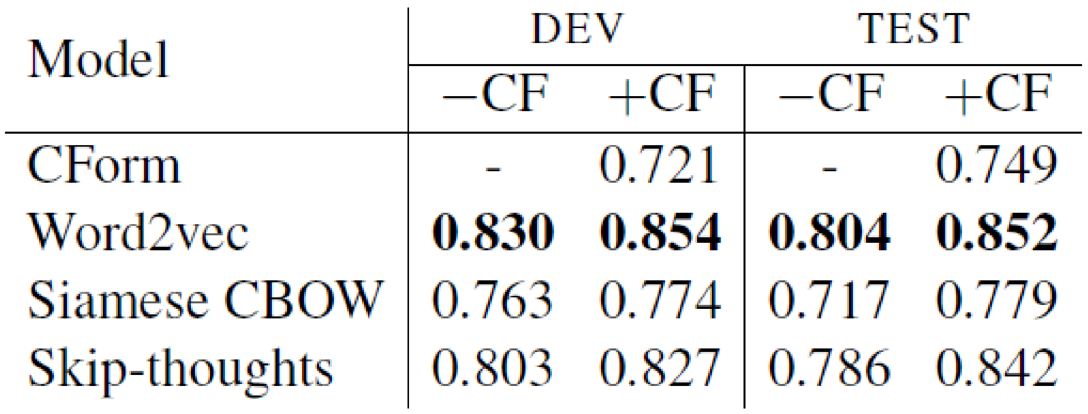
Other such approach is that of **Skip-Thoughts Vectors** (Sent2Vec) [10], used originally by [4] and then used as a base of comparison by [1]. This model uses the continuity of text from books to train an encoder-decoder model that aims to reconstruct the surrounding sentences of an encoded passage, so sentences with similar semantic and syntactic properties are mapped to similar vector representations [10]. This results in an encoder that can product highly generic sentence representations [10]. Sent2Vec was first used for idiom detection by [4] 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 [4]. Utility found in the Sent2Vec model is that it is possible to infer properties of the surrounding context only from the input sentence [4] [10], which allows the classifier to learn lexical and syntactic patterns without complex methods. [4] uses the resulting encodings to train three SVM classifiers with the VNC-Tokens Dataset [11]: 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 2) 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 2 - Precision (P.), Recall (R.), and F1-Score (F1) results on Generic Classifiers by [4]

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 [9], similarly to the approach taken by [6]; 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 [3] 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 [4] 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 [6]. The accuracy score results for the different word embeddings methods used with and without the added CForm are shown on Table 3.



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

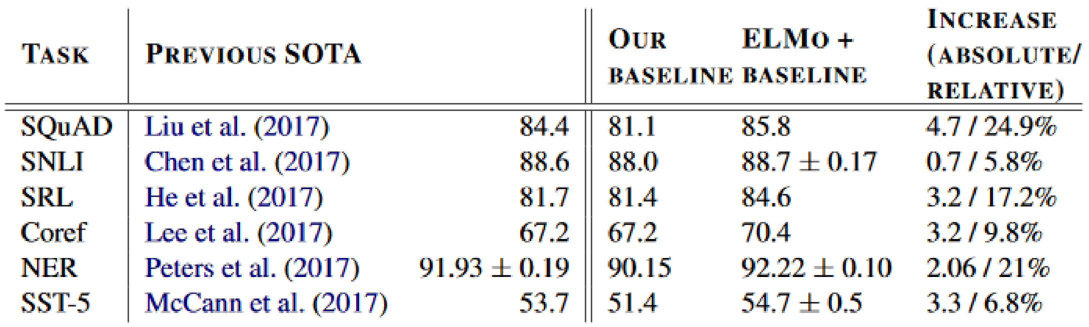
2.4 Further Word Embedding Models

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) representations to evaluate performance developed by [12].

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

Equation 7 - ELMo Layer Computation by [12]

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 7, where h_{k0}^{LM} is the token layer [12]. 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 7 [12]. 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 [12]. According to [12], 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 4. 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 4 - Test set comparison of ELMo enhanced neural models with state-of-the-art model baselines across six benchmark NLP tasks by [12]

2.5 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 **k-Means** algorithm**s** to find these clusters.

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:

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

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.

3 Research Questions and Hypotheses

The hypothesis for this project is that the combination of previously used word embedding vector representations with fixedness metrics will increase the performance of *idiom token* classification by creating linearly separable word representations in vector space. The linear discrimination of these vectors will be evaluated with the usage of a Support Vector Machine with Linear Kernel and the k-Means clustering algorithm.

4 Goals

The goal of this project is to develop a supervised and unsupervised model for idiom detection in text constructed over prior research and combining knowledge for further improvement. The supervised model will have the task of detecting idiomatic use of a VNC in a target sentence, while the unsupervised model will detect idiomatic usage of an unknown VNC in target text and potentially annotate the VNC detected.

4.1 Objectives

The objective of developing the models start with first creating the supervised model, and then expanding the knowledge to train the unsupervised model.

To create the supervised model, we'll replicate the experiments from [1]:

- Implement **Word2Vec**'s Skip-Gram model for sentence encoding
- Implement **Siamese CBOW** model for sentence encoding
- Implement **Skip-thoughts** model for sentence encoding
- Add **CFORM** feature to encoded feature vectors
- Implement **ELMo** model for <u>enhanced</u> sentence encoding
- Implement and train **SVM** with available sentence encodings
- **Evaluate** model performance with Accuracy and F1-Score Metrics

After recreating [1] experiments, a supervised and unsupervised learning models will be implemented using combined knowledge from [1] and [2]:

- Implement **Lexical Fixedness** and **Syntactic Fixedness** formulas on the available corpus
- Implement **Overall Fixedness** formula
- Implement method that extracts the verb-noun pair with highest Overall Fixedness in target sentence. Use this method for unsupervised model.
- Append **Overall Fixedness** metric to the created sentence encodings with the **Word2Vec**, **Siamese CBOW**, **Skip-thoughts**, and **ELMo** models.
- Train **Linear SVM** supervised models and **k-Means** unsupervised model with the new feature vectors.
- Evaluate new models' performance over the previous implementations with Accuracy and F1-Score Metrics.

4.2 Scope and Limitations

The **scope** of the project is the development of a supervised and unsupervised classifier that can detect idiomatic usage of VNCs in text. It is limited to only make use of Verb Noun Combinations type of MWE, such as *see stars*, and won't go into more complex forms of MWEs such as *Time flies when you're having fun*, that still have idiomatic meaning but are not considered in the background research. Also, the VNCs considered during training and testing will only be those that appear in the available datasets, since the objective of this project is not that of providing manual annotations for idiomatic VNCs.

Limitations of the project are the equipment for model training. Hardware is limited by that provided by the University of Essex, so tests may have time and memory constraints regarding the number of tests that can be performed and the number of training examples to use in those states limited by available machine state. Google Colab tool is contemplated to use for the project development, particularly on the model training portion of the project. However, we are still unsure about computation limitations set by Google to stop abuse of their system, so this is not a certain solution to the hardware limitation problem.

5 Project Description

This section explains how the project will be developed, as well as any third-party tools used for its completion.

5.1 VNC Tokens Dataset

The dataset used to finish this project will be the VNC Tokens Dataset developed by Cook et al. 2008 [11]. This dataset was used by previous researches such as [1], [6], and [2] this it will be ideal to replicate their findings and use as a fair baseline for the performance of our classifiers.

The dataset consists of 53 VNC expression, used within roughly 3000 sentences in which they are tagged as expressing "Literal", "Idiomatic", and "Unknown" usage. VNC 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. The corpus corresponds to an entry in the BNC XML dataset, which is required as well [15].

5.2 Third-Party Tools

Gensim

Is a free, open source, tool for unsupervised semantic modelling form plain text [16]. This allows for a simple, pretrained, implementation of Word2Vec by [9].

Siamese CBOW

Public repository containing the code of Siamese CBOW to generate word embeddings based on the paper by the same author [3]. Code can be found here: https://bitbucket.org/TomKenter/siamese-cbow.

Skip-thoughts

Sent2Vec encoder and training code used on the paper by [10]. The source code can be found on the following repository: https://github.com/ryankiros/skip-thoughts.

Scikit-Learn

Simple and efficient tools for data mining and data analysis, build over NumPy, SciPy, and matplotlib [17]. These tools are open source. We will use the SVM and k-Means implementation found in this library for Python.

Tensorflow

Famous open-source platform for machine learning. It has an implementation of the ELMo layer based directly on the original paper by Peters et al. [12].

Google Colab

A Jupyter Notebook type project stored in Google Drive. This technology allows to share and execute on a virtual machine dedicated to a user account. The virtual machines provided allow the user of GPUs and TPUs for faster model training. More information about Google Colab can be found here: https://colab.research.google.com/

5.3 Project Development Methodology

The development of this project will make use of the Agile development process for the implementation of the word embedding algorithms and the supervised and unsupervised classifier algorithms.

Word Embedding Implementation

The first part of the project is to replicate the experiments performed by [1]. This first iteration will implement the embedding models used by [1] to set a baseline for comparison, plus the new embeddings by [12] using the tools described in Section 5.2. Once these word embedding models are created, we'll compute the feature vectors for the sentences corresponding to the idiomatic VNCs inside the VNC-Dataset. These word embeddings will be used to train an SVM implemented with the Scikit-Learn library.

The trained SVM will evaluate the performance of the different vector representations of the tokens in a supervised environment to hold as a baseline for the second part of the project. The evaluation will be done with the Accuracy and F1-Score metrics, as well as a graphical description using a Confusion Matrix graph.

Proposed Feature Vectors and Unsupervised Method Implementation

The second part of the project will first be to implement the fixedness calculation formulas proposed by [2] with word counts from the available BNC XML dataset, as it is the one over which the VNC-Dataset is based. With these wordcounts, the lexical, syntactical and overall fixedness returned by the formulas will be added to the word vectors used in the first part. Note that, since the unsupervised method's purpose is to acquire information without any label intervention, that is the target VNC, we will use the verb-noun combination that returns the highest overall fixedness as our "target" VNC. With these new enhanced word vector representations, we will train a new Linear SVM. The k-Means algorithm will be also trained by initializing the two clusters with a single random example from a literal and an idiomatic use of a VNC.

As in the first part, the new Linear SVM and the k-Means classifiers will be evaluated using the Accuracy and F1-Score metrics. We will evaluate new results for the new feature vectors and determine if the added features cause any increase in performance over the implementation by [1].

6 Evaluation

This section will discuss the evaluation methods for the project. The focus is to evaluate the performance of the models trained on the new feature vectors over the baseline defined by [1].

6.1 Implementation Evaluation

To ensure that our implementations of the described methods are correct and won't reflect incorrect metrics for evaluation over the baseline, we will perform the following actions:

Word-Embedding Evaluation

First, we expect that our word embeddings are correctly extracted from the BNC XML sentences that contain VNCs. To evaluate this, we replicate the experiments by [1] using the same word embeddings plus the recent ELMo vectors. If there's a significant variation (±10%) on the scores generated by our trained SVM, we will review the implementation. This will ensure that our computed word embeddings are correctly generated from the publicly available implementations defined in Section 5.2.

Fixedness Formulas Evaluation

To ensure that the implementation of the Lexical, Syntactic, and Overall Fixedness metrics proposed by [2] are correctly implemented, we will compute the values on a smaller subset of the BNC XML Corpora. These values will be compared to manually curated calculations to ensure that there are no implementation errors on these formulas.

6.2 Performance Evaluation

Following [1] and [4], we use Stratified 10-Fold Cross-Validation for model evaluation. This consists on dividing the available corpus on 10 different sections with same size and same representation of idiomatic and literal use of the target VNCs [13]. We will train the models with 9 sections and evaluate performance with the Accuracy and F1-Score metrics [13]. This is repeated 10 times so that we evaluate the models on each portion of the dataset once [13]. We average the metric scores over all ten runs and use that value to compute the performance of the model [13].

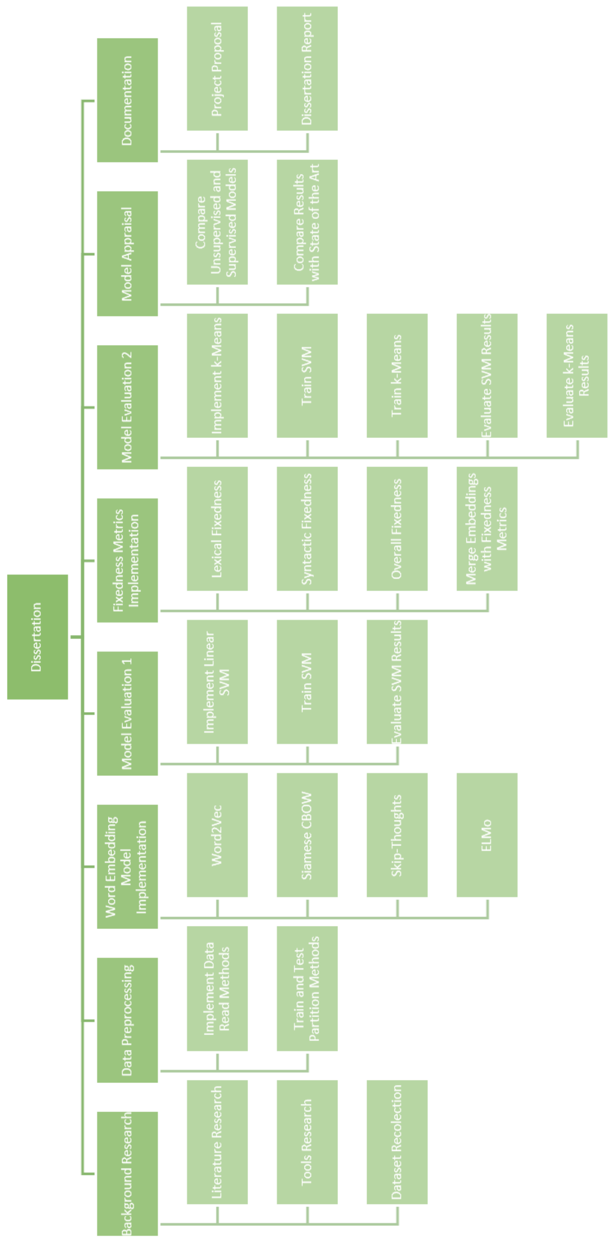
For the unsupervised model evaluation, since the implementation of **k-Means** is inherently random due to the original definition of the algorithm in the process of initial centroid selection, we will perform 10-Fold Cross-Validation using the same random seed for each cluster across all 10 runs. We will repeat this process 10 times for 10 different seeds, averaging the performance of the classifier over the 10, 10-Fold Cross-Validation runs.

For all iterations of the 10-Fold Cross-Validation algorithm, we compute the Confusion Matrix recovered from the predictions by each model to understand the classifications they give and in which scenarios they perform better or not than the baseline classifiers. We also store the misclassified examples for in-depth analysis of were the new models fail in contrast to the baseline implementations.

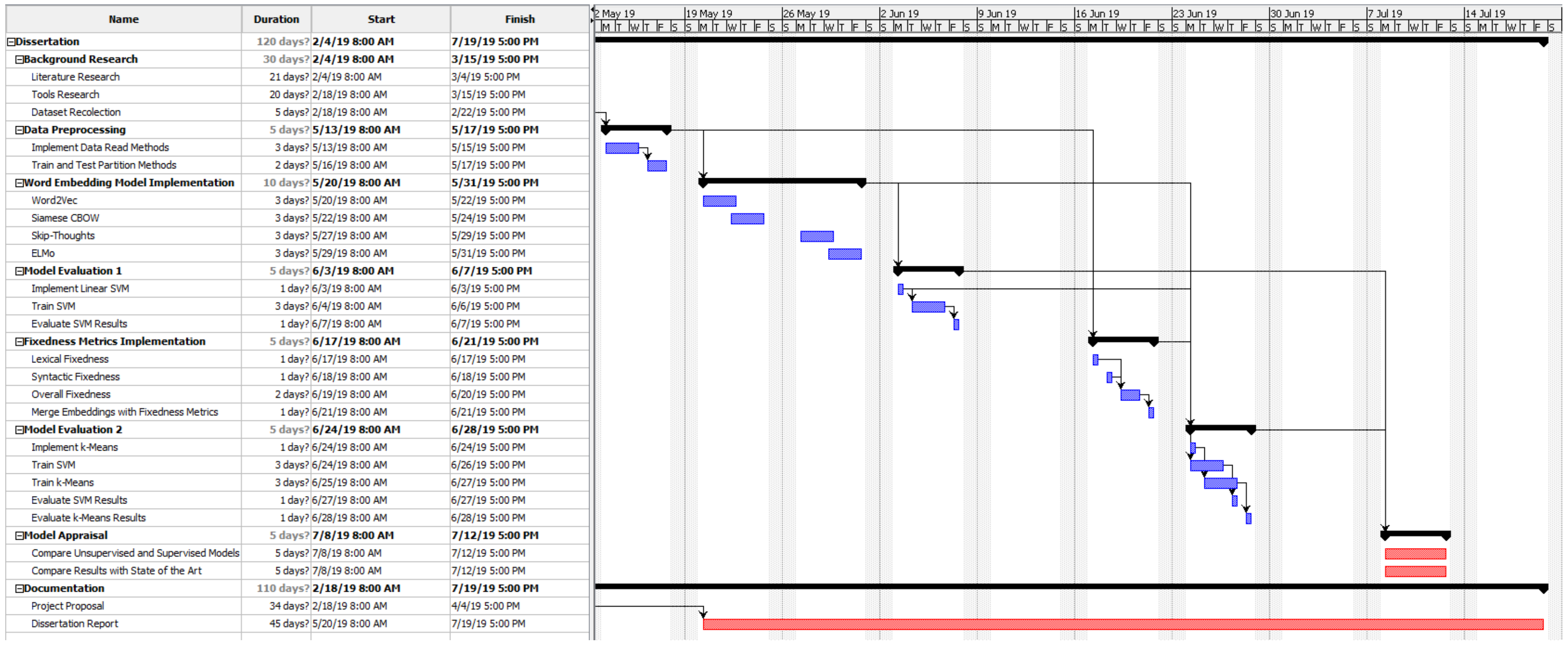
7 Project Plan

The work is sectioned using a Work Breakdown Structure (WBS) diagram for task listing and a Gantt Chart to present the time-bound limitations for development.

7.1 Work Breakdown Structure



7.2 Gantt Chart



Name	Duration	Start	Finish
⊟**Dissertation**	120 days?	**2/4/19 8:00 AM**	**7/19/19 5:00 PM**
⊟**Background Research**	30 days?	**2/4/19 8:00 AM**	**3/15/19 5:00 PM**
Literature Research	21 days?	2/4/19 8:00 AM	3/4/19 5:00 PM
Tools Research	20 days?	2/18/19 8:00 AM	3/15/19 5:00 PM
Dataset Recolection	5 days?	2/18/19 8:00 AM	2/22/19 5:00 PM
⊟**Data Preprocessing**	5 days?	**5/13/19 8:00 AM**	**5/17/19 5:00 PM**
Implement Data Read Methods	3 days?	5/13/19 8:00 AM	5/15/19 5:00 PM
Train and Test Partition Methods	2 days?	5/16/19 8:00 AM	5/17/19 5:00 PM
⊟**Word Embedding Model Implementation**	10 days?	**5/20/19 8:00 AM**	**5/31/19 5:00 PM**
Word2Vec	3 days?	5/20/19 8:00 AM	5/22/19 5:00 PM
Siamese CBOW	3 days?	5/22/19 8:00 AM	5/24/19 5:00 PM
Skip-Thoughts	3 days?	5/27/19 8:00 AM	5/29/19 5:00 PM
ELMo	3 days?	5/29/19 8:00 AM	5/31/19 5:00 PM
⊟**Model Evaluation 1**	5 days?	**6/3/19 8:00 AM**	**6/7/19 5:00 PM**
Implement Linear SVM	1 day?	6/3/19 8:00 AM	6/3/19 5:00 PM
Train SVM	3 days?	6/4/19 8:00 AM	6/6/19 5:00 PM
Evaluate SVM Results	1 day?	6/7/19 8:00 AM	6/7/19 5:00 PM
⊟**Fixedness Metrics Implementation**	5 days?	**6/17/19 8:00 AM**	**6/21/19 5:00 PM**
Lexical Fixedness	1 day?	6/17/19 8:00 AM	6/17/19 5:00 PM
Syntactic Fixedness	1 day?	6/18/19 8:00 AM	6/18/19 5:00 PM
Overall Fixedness	2 days?	6/19/19 8:00 AM	6/20/19 5:00 PM
Merge Embeddings with Fixedness Metrics	1 day?	6/21/19 8:00 AM	6/21/19 5:00 PM
⊟**Model Evaluation 2**	5 days?	**6/24/19 8:00 AM**	**6/28/19 5:00 PM**
Implement k-Means	1 day?	6/24/19 8:00 AM	6/24/19 5:00 PM
Train SVM	3 days?	6/24/19 8:00 AM	6/26/19 5:00 PM
Train k-Means	3 days?	6/25/19 8:00 AM	6/27/19 5:00 PM
Evaluate SVM Results	1 day?	6/27/19 8:00 AM	6/27/19 5:00 PM
Evaluate k-Means Results	1 day?	6/28/19 8:00 AM	6/28/19 5:00 PM
⊟**Model Appraisal**	5 days?	**7/8/19 8:00 AM**	**7/12/19 5:00 PM**
Compare Unsupervised and Supervised Models	5 days?	7/8/19 8:00 AM	7/12/19 5:00 PM
Compare Results with State of the Art	5 days?	7/8/19 8:00 AM	7/12/19 5:00 PM
⊟**Documentation**	110 days?	**2/18/19 8:00 AM**	**7/19/19 5:00 PM**
Project Proposal	34 days?	2/18/19 8:00 AM	4/4/19 5:00 PM
Dissertation Report	45 days?	5/20/19 8:00 AM	7/19/19 5:00 PM

8 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] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

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

[9] 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.

[10] 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.

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

[12] 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.

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

[16] 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.

[17] 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.