Idiom Token Classification using Sentential Distributed Semantics

Giancarlo D. Salton and **Robert J. Ross** and **John D. Kelleher**
Applied Intelligence Research Centre
School of Computing
Dublin Institute of Technology
Ireland
giancarlo.salton@mydit.ie {robert.ross,john.d.kelleher}@dit.ie

Abstract

Idiom token classification is the task of deciding for a set of potentially idiomatic phrases whether each occurrence of a phrase is a literal or idiomatic usage of the phrase. In this work we explore the use of Skip-Thought Vectors to create distributed representations that encode features that are predictive with respect to idiom token classification. We show that classifiers using these representations have competitive performance compared with the state of the art in idiom token classification. Importantly, however, our models use only the sentence containing the target phrase as input and are thus less dependent on a potentially inaccurate or incomplete model of discourse context. We further demonstrate the feasibility of using these representations to train a competitive general idiom token classifier.

1 Introduction

Idioms are a class of multiword expressions (MWEs) whose meaning cannot be derived from their individual constituents (Sporleder et al., 2010). Idioms often present idiosyncratic behaviour such as violating selection restrictions or changing the default semantic roles of syntactic categories (Sporleder and Li, 2009). Consequently, they present many challenges for Natural Language Processing (NLP) systems. For example, in Statistical Machine Translation (SMT) it has been shown that translations of sentences containing idioms receive lower scores than translations of sentences that do not contain idioms (Salton et al., 2014).

Idioms are pervasive across almost all languages and text genres and as a result broad coverage NLP systems must explicitly handle idioms (Villavicencio et al., 2005). A complicating factor, however, is that many idiomatic expressions can be used both literally or figuratively. In general, idiomatic usages are more frequent, but for some expressions the literal meaning may be more common (Li and Sporleder, 2010a). As a result, there are two fundamental tasks in NLP idiom processing: *idiom type* classification is the task of identifying expressions that have possible idiomatic interpretations and *idiom token* classification is the task of distinguishing between idiomatic and literal usages of potentially idiomatic phrases (Fazly et al., 2009). In this paper we focus on this second task, *idiom token* classification.

Previous work on idiom token classification, such as (Sporleder and Li, 2009) and (Peng et al., 2014), often frame the problem in terms of modelling the global lexical context. For example, these models try to capture the fact that the idiomatic expression *break the ice* is likely to have a literal meaning in a context containing words such as *cold*, *frozen* or *water* and an idiomatic meaning in a context containing words such as *meet* or *discuss* (Li and Sporleder, 2010a). Frequently these global lexical models create a different idiom token classifier for each phrase. However, a number of papers on idiom type and token classification have pointed to a range of other features that could be useful for idiom token classification; including local syntactic and lexical patterns (Fazly et al., 2009) and cue words (Li and Sporleder, 2010a). However, in most cases these non-global features are specific to a particular phrase. So a key challenge is to identify from a range of features which features are the correct features to use for idiom token classification for a specific expression.

Meanwhile, in recent years there has been an explosion in the use of neural networks for learning distributed representations for language (e.g., Socher et al. (2013), Kalchbrenner et al. (2014) and Kim (2014)). These representations are automatically trained from data and can simultaneously encode multiple linguistics features. For example, word embeddings can encode gender distinctions and plural-singular distinctions (Mikolov et al., 2013b) and the representations generated in sequence to sequence mappings have been shown to be sensitive to word order (Sutskever et al., 2014). The recent development of Skip-Thought Vectors (or Sent2Vec) (Kiros et al., 2015) has provided an approach to learn distributed representations of sentences in an unsupervised manner.

In this paper we explore whether the representations generated by Sent2Vec encodes features that are useful for idiom token classification. This question is particularly interesting because the Sent2Vec based models only use the sentence containing the phrase as input whereas the baselines systems use full the paragraph surrounding the sentence. We further investigate the construction of a "general" classifier that can predict if a sentence contains literal or idiomatic language (independent of the expression) using just the distributed representation of the sentence. This approach contrasts with previous work that has primarily adopted a "per expression" classifier approach and has been based on more elaborate context features, such as discourse and lexical cohesion between and sentence and the larger context. We show that our method needs less contextual information than the state-of-the-art method and achieves competitive results, making it an important contribution to a range of applications that do not have access to a full discourse context. We proceed by introducing that previous work in more detail.

2 Previous Work

One of the earliest works on idiom token classification was on Japanese idioms (Hashimoto and Kawahara, 2008). This work used a set of features, commonly used in Word Sense Disambiguation (WSD) research, that were defined over the text surrounding a phrase, as well as a number of idiom specific features, which were in turn used to train an SVM classifier based on a corpus of sentences tagged as either containing an idiomatic usage or a literal usage of a phrase. Their results indicated that the WSD features worked well on idiom token classification but that their idioms specific features did not help on the task.

Focusing on idiom token classification in English, Fazly et al. (2009) developed the concept of a canonical form (defined in terms of local syntactic and lexical patterns) and argued that for each idiom there is a distinct canonical form (or small set of forms) that mark idiomatic usages of a phrase. Meanwhile Sporleder and Li (2009) proposed a model based on how strongly an expression is linked to the overall cohesive structure of the discourse. Strong links result in a literal classification, otherwise an idiomatic classification is returned. In related work, Li and Sporleder (2010a) experimented with a range of features for idiom token classification models, including: global lexical context, discourse cohesion, syntactic structures based on dependency parsing, and local lexical features such as cue words, occurring just before or after a phrase. An example of a local lexical feature is when the word *between* occurs directly after *break the ice*; here this could mark an idiomatic usage of the phrase: *it helped to break the ice between Joe and Olivia*. The results of this work indicated that features based on global lexical context and discourse cohesion were the best features to use for idiom token classification. The inclusion of syntactic structures in the feature set provided a boost to the performance of the model trained on global lexical context and discourse cohesion. Interestingly, unlike the majority of previous work on idiom token classification Li and Sporleder (2010a) also investigated building general models that could work across multiple expressions. Again they found that global lexical context and discourse cohesion were the best features in their experiments.

Continuing work on this topic, Li and Sporleder (2010b) present research based on the assumption that literal and figurative language are generated by two different Gaussians. The model representation is based on semantic relatedness features similar to those used earlier in (Sporleder and Li, 2009). A Gaussian Mixture Model was trained using an Expectation Maximization method with the classification of instances performed by choosing the category which maximises the probability of fitting either of the Gaussian components. Li and Sporleder (2010b)'s results confirmed the findings from previous work that figurative language exhibits less cohesion with the surrounding context then literal language.

More recently, Feldman and Peng (2013) describes an approach to idiom token identification that frames the problem as one of outlier detection. The intuition behind this work is that because idiomatic usages of phrases have weak cohesion with the surrounding context they are semantically distant from local topics. As a result, phrases that are semantic outliers with respect to the context are likely to be idioms. Feldman and Peng (2013) explore two different approaches to outlier detection based on principle component analysis (PCA) and linear discriminant analysis (LDA) respectively. Building on this work, Peng et al. (2014) assume that phrases within a given text segment (e.g., a paragraph) that are semantically similar to the main topic of discussion in the segment are likely to be literal usages. They use Latent Dirichlet Allocation (LDA) (Blei et al., 2003) to extract a topic representation, defined as a topic term document matrix, of each text segment within a corpus. They then trained a number of models that classify a phrase in a given text segment as a literal or idiomatic usage by using the topic term document matrix to project the phrase into a topic space representation and label outliers within the topic space as idiomatic. To the best of our knowledge, Peng et al. (2014) is currently the best performing approach to idiom token classification and we use their models as our baseline[1].

3 Skip-Thought Vectors

While idiom token classification based on long range contexts, such as is explored in a number of the models outlined in the previous section, generally achieve good performance, an NLP system may not always have access to the surrounding context, or may indeed find it challenging to construct a reliable interpretation of that context. Moreover, the construction of classifiers for each individual idiom case is resource intensive, and we argue fails to easily scale to under-resourced languages. In light of this, in our work we are exploring the potential of distributed compositional semantic models to produce reliable estimates of idiom token classification.

Skip-Thought Vectors (Sent2Vec) (Kiros et al., 2015) are a recent prominent example of such distributed models. Skip-Thought Vectors are an application of the Encoder/Decoder framework (Sutskever et al., 2014), a popular architecture for NMT (Bahdanau et al., 2015) based on recurrent neural networks (RNN). The encoder takes an input sentence and maps it into a distributed representation (a vector of real numbers). The decoder is a language model that is conditioned on the distributed representation and, in Sent2Vec, is used to "predict" the sentences surrounding the input sentence. Consequently, the Sent2Vec encoder learns (among other things) to encode information about the context of an input sentence without the need of explicit access to it. Figure 1 presents the architecture of Sent2Vec.

More formally, assume a given tuple (s_{i-1}, s_i, s_{i+1}) where s_i is the input sentence, s_{i-1} is the previous sentence to s_i and s_{i+1} is the next sentence to s_i. Let w_i^t denote the t-th word for s_i and $\mathbf{x}_i^t$ denote its word embedding. We follow Kiros et al. (2015) and describe the model in three parts: encoder, decoder and objective function.

Encoder. Given the sentence s_i of length N, let $w_i^1, \ldots, w_i^N$ denote the words in s_i. At each timestep t, the encoder (in this case an RNN with Gated Recurrent Units - GRUs (Cho et al., 2014)) produces a hidden state $\mathbf{h}_i^t$ that represents the sequence $w_i^1, \ldots, w_i^t$. Therefore, $\mathbf{h}_i^N$ represents the full sentence. Each $\mathbf{h}_i^N$ is produced by iterating the following equations (without the subscript i):

$$\mathbf{r}^t = \sigma(\mathbf{W}_r^e \mathbf{x}^t + \mathbf{U}_r^e \mathbf{h}^{t-1}) \quad (1)$$
$$\mathbf{z}^t = \sigma(\mathbf{W}_z^e \mathbf{x}^t + \mathbf{U}_z^e \mathbf{h}^{t-1}) \quad (2)$$
$$\tilde{\mathbf{h}}^t = tanh(\mathbf{W}^e \mathbf{x}^t + \mathbf{U}^{\mathbf{e}}(\mathbf{r}^t \odot \mathbf{h}^{t-1})) \quad (3)$$
$$\mathbf{h}^t = (1 - \mathbf{z}^t) \odot \mathbf{h}^{t-1} + \mathbf{z}^t \odot \tilde{\mathbf{h}}^t \quad (4)$$

where $\mathbf{r}^t$ is the reset gate, $\mathbf{z}^t$ is the update gate, $\tilde{\mathbf{h}}^t$ is the proposed update state at time t and $\odot$ denotes a component-wise product.

Decoder. The decoder is essentially a neural language model conditioned on the input sentence representation $\mathbf{h}_i^N$. However, two RNNs are used (one for the sentence s_{i-1} and the other for the sentence s_{i+1}) with different parameters except the embedding matrix ($\mathbf{E}$), and a new set of matrices ($\mathbf{C}_r$, $\mathbf{C}_z$ and $\mathbf{C}$) are introduced to condition the GRU on $\mathbf{h}_i^N$. Let $\mathbf{h}_{i+1}^t$ denote the hidden state of the decoder of the sentence s_{i+1} at time t. De-

[1]However, it is not possible for us to reproduce their results directly as they "apply the (modified) Google stop list before extracting the topics" (Peng et al., 2014, p. 2023) and, to date, we do not have access to the modified list. So in our experiments we compare our results with the results they report on the same data.

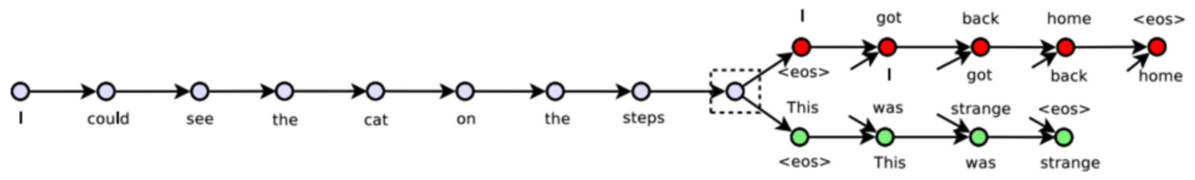

Figure 1: Picture representing the Encoder/Decoder architecture used in the Sent2Vec as shown in Kiros et al. (2015). The gray circles represent the Encoder unfolded in time, the red and the green circles represent the Decoder for the previous and the next sentences respectively also unfolded in time. In this example, the input sentence presented to the Encoder is *I could see the cat on the steps*. The previous sentence is *I got back home* and the next sentence is *This was strange*. Unattached arrows are connected to the encoder output (which is the last gray circle).

coding s_{i+1} requires iterating the following equations:

$$\mathbf{r}^t = \sigma(\mathbf{W}_r^d \mathbf{x}^t + \mathbf{U}_r^d \mathbf{h}^{t-1} + \mathbf{C}_r \mathbf{h}_i^N) \quad (5)$$

$$\mathbf{z}^t = \sigma(\mathbf{W}_z^d \mathbf{x}^t + \mathbf{U}_z^d \mathbf{h}^{t-1} + \mathbf{C}_z \mathbf{h}_i^N) \quad (6)$$

$$\tilde{\mathbf{h}}^t = tanh(\mathbf{W}^d \mathbf{x}^t + \mathbf{U^d}(\mathbf{r}^t \odot \mathbf{h}^{t-1}) + \mathbf{C}\mathbf{h}_i^N) \quad (7)$$

$$\mathbf{h}_{i+1}^t = (1 - \mathbf{z}^t) \odot \mathbf{h}^{t-1} + \mathbf{z}^t \odot \tilde{\mathbf{h}}^t \quad (8)$$

where $\mathbf{r}^t$ is the reset gate, $\mathbf{z}^t$ is the update gate, $\tilde{\mathbf{h}}^t$ is the proposed update state at time t and $\odot$ denotes a component-wise product. An analogous computation is required to decode s_{i-1}.

Given $\mathbf{h}_{i+1}^t$, the probability of the word w_{i+1}^t conditioned on the previous $w_{i+1}^{<t}$ words and the encoded representation produced by the encoder (h_i^N) is:

$$P(w_{i+1}^t | w_{i+1}^{<t}, h_i^N) \propto exp(\mathbf{E}_{w_{i+1}^t} \mathbf{h}_{i+1}^t) \quad (9)$$

where $\mathbf{E}_{w_{i+1}^t}$ denotes the embedding for the word w_{i+1}^t. An analogous computation is performed to find the probability of s_{i-1}.

Objective. Given the tuple (s_{i-1}, s_i, s_{i+1}), the objective is to optimize the sum of the log-probabilities of the next (s_{i+1}) and previous (s_{i-1}) sentences given the distributed representation ($\mathbf{h}_i^N$) of s_i:

$$\sum \log P(w_{i+1}^t | w_{i+1}^{<t}, h_i^N) + P(w_{i-1}^t | w_{i-1}^{<t}, h_i^N) \quad (10)$$

where the total objective is summed over all training tuples (s_{i-1}, s_i, s_{i+1}).

The utility of Sent2Vec is that it is possible to infer properties of the surrounding context only from the input sentence. Therefore, we can assume that the Sent2Vec distributed representation is also carrying information regarding its context (without the need to explicitly access it). Following that intuition, we can train a supervised classifier only using the labelled sentences containing examples of idiomatic or literal language use without modelling long windows of context or using methods to extract topic representations.

4 Experiments

In the following we describe a study that evaluates the predictiveness of the distributed representations generated by Sent2Vec for idiom token classifier. We first evaluate these representations using a "per expression" study design (i.e., one classifier per expression) and compare our results to those of Peng et al. (2014) who applied multi-paragraphs contexts to generate best results. We also experiment with a "general" classifier trained and tested on a set of mixed expressions.

4.1 Dataset

In order to make our results comparable with (Peng et al., 2014) we used the same VNC-Tokens dataset (Cook et al., 2008) that they used in their experiments. The dataset used is a collection of sentences containing 53 different Verb Noun Constructions[2] (VNCs) extracted from the British National Corpus (BNC) (Burnard, 2007). In total, the VNC-Token dataset has 2984 sentences where each sample sentence is labelled with one of three labels: *I* (idiomatic); *L* (literal); or *Q* (unknown).

[2]This verb-noun constructions can be used either idiomatically or literally.

Of the 56 VNCs in the dataset 28 of these expressions have a reasonably balanced representation (with similar numbers of idiomatic and literal occurrences in the corpus) and the other 28 expressions have a skewed representation (with one class much more common then the other). Following the approach taken by (Peng et al., 2014), in this study we use the "balanced" part of the dataset and considered only those sentences labelled as "I" and "L" (1205 sentences - 749 labelled as "I" and 456 labelled as "L").

Peng et al. (2014) reported the precision, recall and f1-score of their models on 4 of the expressions from the balanced section of dataset: **BlowWhistle**; **MakeScene**; **LoseHead**; and **TakeHeart**. So, our first experiment is designed to compare our models with these baseline systems on a "per-expression" basis. For this experiment we built a training and test set for each of these expressions by randomly sampling expressions following the same distributions presented in Peng et al. (2014). In Table 1 we present those distribution and the split into training and test sets. The numbers in parentheses denote the number of samples labelled as "I".

Expression	Samples	Train Size	Test Size
BlowWhistle	78 (27)	40 (20)	38 (7)
LoseHead	40 (21)	30 (15)	10 (6)
MakeScene	50 (30)	30 (15)	20 (15)
TakeHeart	81 (61)	30 (15)	51 (46)

Table 1: The sizes of the samples for each expression and the split into training and test set. The numbers in parentheses indicates the number of idiomatic labels within the set. We follow the same split as described in Peng et al. (2014).

While we wish to base our comparison on the work of Peng et al. (2014) as it is the current state of the art, this is not without its own challenges. In particular we see the choice of these 4 expression as a somewhat random decision as other expressions could also be selected for the evaluation with similar ratios to those described in Table 1. Moreover, the choosen expressions are all semi-compositional and do not consider fully non-compositional expressions (although we believe the task of classifying non-compositional expressions would be easier for any method aimed at idiom token classification as these expressions are high-fixed) .A better evaluation would consider all the 28 expressions of the balanced part of the VNC-tokens dataset. In addition, we also see this choice of training and test splits as somewhat arbitrary. For two of the expressions the test set contain samples in a way that one of the classes outnumber the other by a great amount: for *BlowWhistle*, the literal class contains roughly 4 times more samples than the idiomatic class; and for *TakeHeart* the idiomatic class contains roughly 9 times more samples than the literal class. Our concerns with these very skewed test set ratios is that it is very easy when applying a per expression approach (i.e., a separate model for each expression) for a model to achieve good performance (in terms of precision, recall, ad f1) if the positive class is the majority class in the test set. However, despite these concerns, in our first experiment in order to facilitate comparison with the prior art we follow the expression selections and training/test splits described in Peng et al. (2014).

Studies on the characteristics of distributed semantic representations of words have shown that similar words tend to be represented by points that are close to each other in the semantic feature space (e.g. Mikolov et al. (2013a)). Inspired by these results we designed a second experiment to test whether the Sent2Vec representations would cluster idiomatic sentences in one part of the feature space and literal sentences in another part of the space. For this experiment we used the entire "balanced" part of the VNC-tokens dataset to train and test our "general" (multi-expression) models. In this experiment we wanted the data to reflect, as much as possible, the real distribution of the idiomatic and literal usages of each expression. So, in constructing our training and test set we tried to maintain for each expression the same ratio of idiomatic and literal examples across the training and test set. To create the training and test sets, we split the dataset into roughly 75% for training (917 samples) and 25% for testing (288 samples). We randomly sample the expressions ensuring that the ratio of idiomatic to literal expressions of each expression were maintained across both sets. In Table 2 we show the expressions used and their split into training and testing. The numbers in parentheses are the number of samples labelled as "I".

4.2 Sent2Vec Models

To encode the sentences into their distributed representations we used the code and models made available[3] by Kiros et al. (2015). Using their

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

Expression	Samples	Train Size	Test Size
BlowTop	28 (23)	21 (18)	7 (5)
BlowTrumpet	29 (19)	21 (14)	8 (5)
BlowWhistle	78 (27)	59 (20)	19 (7)
CutFigure	43 (36)	33 (28)	10 (8)
FindFoot	53 (48)	39 (36)	14 (12)
GetNod	26 (23)	19 (17)	7 (6)
GetSack	50 (43)	40 (34)	10 (9)
GetWind	28 (13)	20 (9)	8 (4)
HaveWord	91 (80)	69 (61)	22 (19)
HitRoad	32 (25)	24 (19)	8 (6)
HitRoof	18 (11)	14 (9)	4 (2)
HitWall	63 (7)	50 (6)	13 (1)
HoldFire	23 (7)	19 (5)	4 (2)
KickHeel	39 (31)	30 (23)	9 (8)
LoseHead	40 (21)	29 (15)	11 (6)
LoseThread	20 (18)	16 (15)	4 (3)
MakeFace	41 (27)	31 (21)	10 (6)
MakeHay	17 (9)	12 (6)	5 (3)
MakeHit	14 (5)	9 (3)	5 (2)
MakeMark	85 (72)	66 (56)	19 (16)
MakePile	25 (8)	18 (6)	7 (2)
MakeScene	50 (30)	37 (22)	13 (8)
PullLeg	51 (11)	40 (8)	11 (3)
PullPlug	64 (44)	49 (33)	15 (11)
PullPunch	22 (18)	18 (15)	4 (3)
PullWeight	33 (27)	24 (20)	9 (7)
SeeStar	61 (5)	49 (3)	12 (2)
TakeHeart	81 (61)	61 (45)	20 (16)

Table 2: The sizes of the samples for each expression and the split into training and test set. The numbers in parentheses indicates the number of idiomatic labels within the set.

models it is possible to encode the sentences into three different formats: *uni-skip* (which uses a regular RNN to encode the sentence into a 2400-dimensional vector); *bi-skip* (that uses a bidirectional RNN to encode the sentence also into a 2400-dimensional vector); and the *comb-skip* (a concatenation of *uni-skip* and *bi-skip* which has 4800 dimensions). Their models were trained using the BookCorpus dataset (Zhu et al., 2015) and has been tested in several different NLP tasks as semantic relatedness, paraphrase detection and image-sentence ranking. Although we experimented with all the three models, in this paper we only report the results of classifiers trained and tested using the *comb-skip* features.

4.3 Classifiers

4.3.1 "Per-expression" models

The idea behind Sent2Vec is similar to those of word embeddings experiments: sentences containing similar meanings should be represented by points close to each other in the feature space. Following this intuition we experiment first with a similarity based classifier, the K-Nearest Neighbours (k-NN). For the k-NNs we experimented with $k = \{2, 3, 5, 10\}$.

We also experimented with a more advanced algorithm, namely the Support Vector Machine (SVM) (Vapnik, 1995). We trained the SVM under three different configurations:

Linear-SVM-PE[4]. This model used a "linear" kernel with $C = 1.0$ on all the classification setups.

Grid-SVM-PE. For this model we performed a grid search for the best parameters for each expression. The parameters are: *BlowWhiste* $= \{$ kernel: 'rbf', $C = 100\}$; *LoseHead* $= \{$ kernel: 'rbf', $C = 1$ $\}$; *MakeSene* $= \{$ kernel: 'rbf', $C = 100$ $\}$; *TakeHeart* $= \{$ kernel: 'rbf', $C = 1000$ $\}$.

SGD-SVM-PE. This model is a SVM with linear kernel but trained using stochastic gradient descent (Bottou, 2010). We set the SGD's learning rates (α) using a grid search: *BlowWhiste* $= \{\alpha = 0.001$ $\}$; *LoseHead* $= \{\alpha = 0.01$ $\}$; *MakeSene* $= \{\alpha = 0.0001$ $\}$; *TakeHeart* $= \{\alpha = 0.0001$ $\}$; *FullDataset* $= \{\alpha = 0.0001$ $\}$. We trained these classifiers for 15 epochs.

4.3.2 "General" models

We consider the task of creating a "general" classifier that takes an example of any potential idiom and classifying it into idiomatic or literal usage more difficult than the "per-expression" classification task. Hence we executed this part of the study with the SVM models only. We trained the same three types of SVM models used in the "per-expression" approach but with the following parameters:

Linear-SVM-GE[5]. This model used a linear kernel with $C = 1.0$ for all the classification sets.

Grid-SVM-GE. For this model we also performed a grid search and set the kernel to "polynomial kernel" of $degree = 2$ with $C = 1000$.

SGD-SVM-GE. We also experimented with a SVM with linear kernel trained using stochastic gradient descent. We set the SGD's learning rate $\alpha = 0.0001$ after performing a grid search. We trained this classifier for 15 epochs.

5 Results and Discussion

We first present the results for the per expression comparison with Peng et al. (2014) and then in

[4]PE stands for "per-expression"

[5]GE stands for "general".

Models	BlowWhistle			LoseHead			MakeScene			TakeHeart		
	P.	R.	F1	P.	R.	F1	P.	R.	F1	P.	R.	F1
Peng et. al (2014)												
FDA-Topics	0.62	0.60	0.61	0.76	0.97	**0.85**	0.79	0.95	0.86	0.93	0.99	**0.96**
FDA-Topics+A	0.47	0.44	0.45	0.74	0.93	0.82	0.82	0.69	0.75	0.92	0.98	0.95
FDA-Text	0.65	0.43	0.52	0.72	0.73	0.72	0.79	0.95	0.86	0.46	0.40	0.43
FDA-Text+A	0.45	0.49	0.47	0.67	0.88	0.76	0.80	0.99	**0.88**	0.47	0.29	0.36
SVMs-Topics	0.07	0.40	0.12	0.60	0.83	0.70	0.46	0.57	0.51	0.90	1.00	0.95
SVMs-Topics+A	0.21	0.54	0.30	0.66	0.77	0.71	0.42	0.29	0.34	0.91	1.00	0.95
SVMs-Text	0.17	0.90	0.29	0.30	0.50	0.38	0.10	0.01	0.02	0.65	0.21	0.32
SVMs-Text+A	0.24	0.87	0.38	0.66	0.85	0.74	0.07	0.01	0.02	0.74	0.13	0.22
Distributed Representations												
KNN-2	0.61	0.41	0.49	0.30	0.64	0.41	0.55	0.89	0.68	0.46	0.96	0.62
KNN-3	0.84	0.32	0.46	0.58	0.65	0.61	0.88	0.88	**0.88**	0.72	0.94	0.81
KNN-5	0.79	0.28	0.41	0.57	0.65	0.61	0.87	0.83	0.85	0.73	0.94	0.82
KNN-10	0.83	0.30	0.44	0.28	0.68	0.40	0.85	0.83	0.84	0.78	0.94	0.85
Linear SVM	0.77	0.50	0.60	0.72	0.84	0.77	0.81	0.91	0.86	0.73	0.96	0.83
Grid SVM	0.80	0.51	**0.62**	0.83	0.89	**0.85**	0.80	0.91	0.85	0.72	0.96	0.82
SGD SVM	0.70	0.40	0.51	0.73	0.79	0.76	0.85	0.91	**0.88**	0.61	0.95	0.74

Table 3: Results in terms of precision (P.), recall (R.) and f1-score (F1) on the four chosen expressions. The results of (Peng et al., 2014) are those of the multi-paragraphs method. The bold values indicates the best results for that expression in terms of f1-score.

Section 5.2 we present the results for the "general' classifier approach.

5.1 Per-Expression Classification

The averaged results over 10 runs in terms of precision, recall and f1-score are presented in Table 3. When calculating these metrics, we considered the positive class to be the "I" (idiomatic) label. We used McNemar's test (McNemar, 1947) to check the statistical significance of our models' results and found all our results to be significant at $p < 0.05$.

We can see in Table 3 that some of our models outperform the baselines on 1 expression (*BlowWhistle*) and achieved the same f1-scores on 2 expressions (*LoseHead* and *MakeScene*). For theses 3 expressions, our best models generally had higher precision than the baselines, finding more idioms on the test sets. In addition, for *MakeScene*, 2 of our models achieved the same f1-scores (*KNN-3* and *SGD-SVM-PE*), although they have different precision and recall.

The only expression on which a baseline model outperformed all our models was *TakeHeart* where it achieved higher precision, recall and f1-scores. Nevertheless, this expression had the most imbalanced test set, with roughly 9 times more idioms than literal samples. Therefore, if the baseline label all the test set samples as idiomatic (including the literal examples), it would still have the best results. It is thus worth emphasizing that the choices of distributions for training and test sets in Peng et al's work seems arbitrary and does not reflect the real distribution of the data in a balanced corpus. Also, Peng et al. (2014) did not provide the confusion matrices for their models so we cannot analyse their model behaviour across the classes.

That aside, while our best models share the same f1-score with the baseline on 2 of the expressions, we believe that our method is more powerful if we take into account that we do not explicitly access the context surrounding our input sentences. We can also consider that our method is cheaper than the baseline in the sense that we do not need to process words other than the words in the input sentence.

In addition, we note that the SVMs generally outperform the KNNs, although no single model perform best across all expressions. Regardless of the fact that the KNN-3 achieved the same f1-score as SGD-SVM on *MakeScene*, the SVM consistently scored higher than the KNNs on all expressions. This is an interesting finding if we consider that our feature vector is 4800-dimensional and the SVMs are projecting these features into a space that has much more than 4800 dimensions and not incurring into the "curse of dimensionality". Furthermore, other work using Sent2vec have shown the capabilities of the Sent2Vec representations to capture features that are suited to various NLP tasks where semantics is involved (e.g., paraphrase detection and semantic relatedness (Kiros et al., 2015)). These results together with our findings suggests that the factors in-

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 4: Precision (P.), recall (R.) and f1-scores (F1) calculated on the expressions of the balanced part of the VNC-Tokens dataset. The expressions marked with * indicate the expressions also evaluated with the "per-expression" classifiers.

volved in distinguishing between the semantics of idiomatic and literal language are deeply entrenched in language generation and only a high-dimensional representation can enable a classifier to make that distinction. This observation also implies that the contribution of each feature (generated by the distributed representation) is very small, given the fact that we need that many dimensions and the space needed to unpack the components of literal and idiomatic language has many more dimensions than the input space. Therefore, the current manually engineered features (i.e., the features used in previous idiom token classification) are only capturing a small portion of these dimensions and assigning more weight to these dimensions while other dimensions (not captured) are not considered (i.e., as they are not considered, the features represented by these dimensions have their weight equal to 0)

Another point for consideration is the fact that the combination of our model with the work of Peng et al. (2014) may result in a stronger model on this "per-expression" setting. Nevertheless, as previously highlighted, it was not possible for us to directly re-implement their work.

5.2 General Classification

Moving on to the general classification case, we present the average results (in terms of precision, recall and f1-score) over 10 runs to our "general" classifiers on the balanced part of the VNC-Tokens dataset. Once again, the positive class is assumed to be the "I" (idiomatic) label and we split the outcomes per expression. It should be noted that the "per-expression" evaluation was performed using a balanced set to train the classifiers while in this experiment we maintained the ratio of idiomatic to literal usages for each expression across the training and test sets. Our motivation for maintaining this ratio was to simulate the real distribution of the classes in the corpus.

We present results for the four individual MWEs used in the per-sentence based evaluation as well as a set of averages made over all 28 expression in the "balanced" portion of the dataset. Referring to the results we first of all note the overall performance of the "general" classifiers is fairly high with 2 classifiers (Linear-SVM-GE and Grid-SVM-GE) sharing the same precision, recall and f1-scores. While averages here are the same across the two classifiers, it is worth noting that deviations occured across individual MWE types, though these deviations balanced out across the data set. Although not displayed in this table due to space limitations, it should be noted that all the 3 classifier had a extremely low performance on *SeeStar* (f1 = 0.15, 0.15 and 0.17 respectively).

If we compare the performance of the 4 expressions analysed in the "per-expression" experiment we can observe that all the "general" classifiers had a better performance over *BlowWhistle* and the Linear-SVM-GE also performed better on *MakeScene*. Nevertheless we should emphasize that the "general" classifier's evaluation is closer to what we would expect in a real data distribution than the evaluation presented on the "per-expression" section. This does not invalidate the evaluation of the latter but when we have access to a real data distribution it should also be taken into account when performing a ML evaluation.

In general, the results look promising. It is interesting to see how the classifiers trained on a set of mixed expressions ("general" classifiers) had a performance close to the "per-expression" classifiers, even though the latter were trained and tested on "artificial" training and test sets that do not reflect the real data distributions. We believe that these results indicate that the distributed representations generated by Sent2Vec are indeed clustering together sentences within the same class (idiomatic or literal) in feature space.

6 Conclusions and Future Work

In this paper we have investigated the use of distributed compositional semantics in literal and idiomatic language classification, more specifically using Skip-Thought Vectors (Sent2Vec). We followed the intuition that the distributed representations generated by Sent2Vec also include information regarding the context where the potential idiomatic expression is inserted and therefore is sufficient for distinguishing between idiomatic and literal language use.

We tested this approach with different Machine Learning (ML) algorithms (K-Nearest Neighbours and Support Vector Machines) and compared our work against a topic model representation that include the full paragraph or the surrounding paragraphs where the potential idiom is inserted. We have shown that using the Sent2Vec representations our classifiers achieve better results in 3 out of 4 expressions tested. We have also shown that our models generally present better precision and/or recall than the baselines.

We also investigated the capability of Sent2Vec clustering representations of sentences within the same class in feature space. We followed the intuition presented by previous experiments with distributed representations that words with similar meaning are clustered together in feature space and experimented with a "general" classifier that is trained on a dataset of mixed expressions. We have shown that the "general" classifier is feasible but the traditional "per-expression" does achieve better results in some cases.

In future work we plan to investigate the use of Sent2Vec to encode larger samples of text - not only the sentence containing idioms. We also plan to further analyse the errors made by our "general" model and investigate the "general" approach on the skewed part of the VNC-tokens dataset. We also plan to investigate an end-to-end approach based on deep learning-based representations to classify literal and idiomatic language use.

In addition, we also plan to compare our work to the method of Sporleder et al. (2010) as well apply our work on the IDX Corpus (Sporleder et al., 2010) and to other languages. The focus of these future experiments will be to test how our approach which is relatively less dependent on NLP resources compares with these other methods for idiom token classification.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments and feedback. Giancarlo D. Salton would like to thank CAPES ("Coordenação de Aperfeiçoamento de Pessoal de Nível Superior") for his Science Without Borders scholarship, proc n. 9050-13-2.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March.

Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of the 19th International Conference on Computational Statistics (COMPSTAT'2010)*, pages 177–187.

Lou Burnard. 2007. Reference guide for the british national corpus (xml edition). Technical report, http://www.natcorp.ox.ac.uk/.

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

Paul Cook, Afsaneh Fazly, and Suzanne Stevenson. 2008. The VNC-Tokens Dataset. In *Proceedings of the LREC Workshop: Towards a Shared Task for Multiword Expressions (MWE 2008)*, Marrakech, Morocco.

Afsanesh Fazly, Paul Cook, and Suzanne Stevenson. 2009. Unsupervised type and token identification of idiomatic expressions. In *Computational Linguistics*, volume 35, pages 61–103.

Anna Feldman and Jing Peng. 2013. Automatic detection of idiomatic clauses. In *Proceedings of the 14th International Conference on Computational Linguistics and Intelligent Text Processing - Volume Part I*, CICLing'13, pages 435–446.

Chikara Hashimoto and Daisuke Kawahara. 2008. Construction of an idiom corpus and its application to idiom identification based on wsd incorporating idiom-specific features. In *Proceedings of the conference on empirical methods in natural language processing*, pages 992–1001. Association for Computational Linguistics.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. June.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in Neural Information Processing Systems 28*, pages 3276–3284.

Linlin Li and Caroline Sporleder. 2010a. Linguistic cues for distinguishing literal and non-literal usages. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 683–691.

Linlin Li and Caroline Sporleder. 2010b. Using gaussian mixture models to detect figurative language in context. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 297–300, Stroudsburg, PA, USA. Association for Computational Linguistics.

Quinn McNemar. 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157.

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

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *The 2013 Conference of the North Americal Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 746–751.

Jing Peng, Anna Feldman, and Ekaterina Vylomova. 2014. Classifying idiomatic and literal expressions using topic models and intensity of emotions. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2019–2027, October.

Giancarlo D. Salton, Robert J. Ross, and John D. Kelleher. 2014. An Empirical Study of the Impact of Idioms on Phrase Based Statistical Machine Translation of English to Brazilian-Portuguese. In *Third Workshop on Hybrid Approaches to Translation (HyTra) at 14th Conference of the European Chapter of the Association for Computational Linguistics.*

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642.

Caroline Sporleder and Linlin Li. 2009. Unsupervised recognition of literal and non-literal use of idiomatic expressions. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 754–762.

Caroline Sporleder, Linlin Li, Philip Gorinski, and Xaver Koch. 2010. Idioms in context: The idix corpus. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC-2010)*, pages 639–646.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112.

Vladimir N. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA.

Aline Villavicencio, Francis Bond, Anna Korhonen, and Diana McCarthy. 2005. Editorial: Introduction to the special issue on multiword expressions: Having a crack at a hard nut. *Comput. Speech Lang.*, 19(4):365–377.

Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *arXiv preprint arXiv:1506.06724*.