The Impact of Word2Vec Embeddings in Improving Compositionality Classification

Anonymous ACL submission

Abstract

We investigate whether the task of predicting the meaning of a closed compound word from the meaning of its two constituents on a dataset containing simple features can be improved through computing the word2vec embedding similarities of the constituents with the compound word. We construct an experiment wherein Logistic regression, SVM and AdaBoost models are employed to determine the efficacy of the similarities in such a setting. We find that the addition of similarities obtained through word2vec embeddings as features considerably improve performance on the task at hand.

1 Introduction

In a general exploration of word etymologies, we noted that some closed compound words have a very clear source, such as the word ‘lifejacket’, and can be classified as endocentric compounds. On the other hand, exocentric compound nouns are seemingly an arbitrary combination with no real connection to their meaning- for instance, the word ‘butterfly’. Others still sit somewhat in the middle, in a gray area where their classification into either camp can be convincingly argued: for example, the word lawsuit is clearly related to legal discourse, but the literal meaning of its subwords do not directly infer ts conventional meaning. The evaluation of a closed compound can be difficult for us as people to classify, which sparked the question about whether an NLP model would be able to, and to what extent the constituents’ meaning inform this classification.

The motivation behind this paper is to devise an experiment that investigates our hypothesis that word2vec word embeddings play a role in enhancing a model’s ability to distinguish and categorize holistic closed English compound words from other, seemingly arbitrary closed compounds, when added to a preexisting closed compound word dataset. We hereafter refer to this as the compositionality of a compound word. These features are specifically chosen due to the efficiency of word2vec embeddings in determining word relatedness through cosine similarities, which we believe to be crucial in inferring compositionality.

2 Related Works

There has been an abundance of academic research around the compositionality of open compound words (which include a space between their constituents) and hyphenated compound words. Below we discuss a few of the leading academics in this field and their methodologies, results and conclusions. However, work around closed compound words seems to be fairly sparse. Nevertheless, given the close relation between our hypothesis and those explored below, these papers were very informative for our purposes and methodology.

2.1 LADEC: The Large Database of English Compounds (Gagné et al., 2019)

Explained in more detail in section 3.1, this is the dataset we use to gather training data for our experiment. With upward of 8,000 English words, it is the biggest existing dataset specifically designed to be used in compound word research in the English language. Not only does it provide the compound words themselves, but the authors also include scores and ratings for a number of different parameters. While we only use a handful of these, which are outlined in detail in section 3.1, some of their main parameters included a measure of semantic transparency, which, similarly to our process, assigns a score that reflects the relationship between a compound and its constituents. However, their approach differs in that they actually gathered participants to perform a rating task in order to collect meaning retention and meaning predictability judgement data. In our experiment, we produce three cosine similarity vectors for each word and compare them to draw a score. This process is explained further in section 3.2.

2.2 Algorithm for Automatic Interpretation of Noun Sequences (Vanderwende, 1994)

Vanderwende's 1994 paper made major strides in compound word interpretation due to the design and presentation of an algorithm specifically aimed at working without context. The approach instead uses dictionary definitions of member nouns in a compound to interpret and output a definition for the overall sequence. Some of Vanderwende's techniques included labelling nouns with better specifiers, such as 'locative' or 'time' to better inform the model's interpretation. Although this paper was extremely informative to contextualize early approaches to compound noun interpretation, the experiment dealt exclusively with open compound nouns, which has the big consequence of dealing then mostly with endocentric compounds. While there is already a heavy majority of endocentric closed compounds, the imbalance is even greater in open compounds as they have generally been formed from the common description of a noun (eg 'swimming pool, 'ice cream'). In the case of closed compounds, their formation is often more nuanced as they themselves have to classify overall as a noun. Thus, this is a big difference in the experiment we conducted as compared to Vanderwende's work.

2.3 Automatic Noun Compound Interpretation using Deep Neural Networks and Word Embeddings (Dima and Hinrichs, 2015)

Aside from also using open compound words, Dima and Hinrich's 2015 work has another major difference as related to our current experiment: while we use the LADEC dataset (see Subsection 3.1), the authors here use the Tratz dataset. Tratz compound words come categorized, with 12 possible labels for each word: some examples include Objective (eg leaf blower), Temporal Group (eg birth date) and Personal, (Ronald Reagan). These 12 categories are split into as many as 6 more detailed sub-categories Temporal Group contains both the TIMEOF-1 tag (eg night work) and TIMEOF-2 tag (which is where 'birth date' is classified).

The result of using a Tratz dataset is that Dima Hinrichs' model, a Deep Neural Network in this case, has a lot more context and information to work with than in our experiment, which expects the model to gather context while in training.

3 Dataset

Our final working datasets are a combination of features from the LADEC dataset, Word2Vec embedding similarity scores and some feature engineering in order to produce ten slightly different datasets on which models will be evaluated.

3.1 LADEC

For our purposes, we choose to narrow down from the many parameters put forth by LADEC to a handful of parameters we deem most important on a basic level - further in the experiment, we add or remove certain parameters as a test measure to see which provide better performance. We choose the following parameters: nparses, bgJonesMewhort, bgSUBTLEX and bgFacebook. nparses represesnts how many different parsing a compound word has. We discard rows where its components are incorrectly parsed (e.g., if parasailing has 2 parsing, para and sailing as well as paras and ailing, we discard paras and ailing). bgJonesMewhort, bgSUBTLEX and bgFacebook are the bigram frequency of letters at the morpheme boundary (e.g., the r and f in waterfall), we retrieve bgJonesMewhort's data from the table reported in Jones and Mewhort (Jones and Mewhort, 2004). bgSUBTLEX's data is calculated using Mathematica, such that the bigram frequencies would accurately reflect exposure to the various letter combinations. As for bgFacebook, the data is collected from English posts on Facebook between November 2014 and January 2015 (Gagné et al., 2019).

3.2 Word2Vec

The Word2Vec embeddings are vector representation of words obtained by training neural networks, and inherently capture semantic and syntactic relationships between words by modelling words and their contexts. There are two training tasks through which word embeddings are generated: the Skip-gram model that learns the embeddings of context words given a target embedding (Mikolov et al., 2013a), and the Continuous Bag-of-Words model that learns target word embeddings given the embeddings of the context words (Mikolov et al., 2013b). They are especially good at encoding word relatedness using cosine distances between pairs of words.

We use word2vec embeddings that are pre-trained on one billion words from Google News, which account for a vocabulary of 692K words (Mikolov et al., 2013a). These word embeddings are then used to produce three cosine similarity scores for each constituents and compound word pair found in the aforementioned LADEC dataset. The three scores are sim_c1stim and sim_c2stim, the similarity between the first and second constituents' embeddings and the compound words' respectively, as well as $sim\_c1c2stim$ the similarity between the result of the vector addition of the first and second constituents and the embedding of the compound word itself. This allows the incorporation of similarity features into our working datasets, and will ultimately be crucial in assessing the impact of the similarities in the overall classification task.

3.3 Feature Engineering

The features that have been introduced so far may need to be augmented in order to increase the complexity of our models and reduce their bias. To alleviate this problem, we use two methods of engineering new features from our current set.

The first method is to compute the first order interactions between the features, and then augment the feature set with these. This not only increases the complexity of the model but also allows for complex dependencies between features to be captured. Therefore, for an instance with feature vector $\mathbf{x} = [x_1, x_2, ...x_m]$ where m represents the number of features, we compute interaction terms $\mathbf{x_{int}} = [x_{11}, x_{12}, ..., x_{21}, .., x_{\text{mm}}]$ where $x_{\text{ij}} = x_{\text{i}}x_{\text{j}}$ $\forall i, j \in 1, .., m$. Augmenting our dataset with these new features means each example has a new feature vector which is the concatenation of $\mathbf{x}$ and $\mathbf{x_{int}}$.

The second method computes a log transformation of each of the feature column $\mathbf{x}_1, ..., \mathbf{x}_m$ of the examples in the dataset according to the following formula.

$$\log(\mathbf{x}_i - x_{i,\text{min}} + 1) \; i \in 1, .., m$$

where $x_{i,\text{min}}$ is the minimum value of the i^{th} column of the dataset. Computing these transformation and then augmenting the original features with these additional transformed features. This again increases the complexity of the model via the addition of more parameters to learn, but also helps with potentially skewed data, making the distribution of the features closer to normal(Feng et al., 2014).

3.4 Final Working Datasets

Using the features described in 3.1, we further clean our dataset by using z score normalization on the bigram letter frequencies. Thus, handling the outlier issue. Moreover, we discard any examples that have empty entries in at least one of the mentioned features as well as any instances of constituent or compound words that are not part of the pre-trained word2vec embeddings. Finally, we generate the following ten datasets:

- **V**: Vanilla, the features of this dataset are nparses, bgJonesMewhort, bgSUBTLEX and bgFacebook.
- **NV**: Non-vanilla, the features of this dataset are nparses, bgJonesMewhort, bgSUBTLEX, bgFacebook, sim_c1stim, sim_c2stim, and $sim\_c1c2stim$.
- **VI**: Vanilla and interactions, has extra features from applying interactions on the vanilla dataset.
- **VL**: Vanilla and log transformation, has extra features from applying log transformation on the vanilla dataset.
- **VIL**: Vanilla, interactions and log transformation, has extra features from applying log transformation and interactions on the vanilla dataset.
- **NVI**: Non-Vanilla and interactions, has extra features from applying interactions on the non-vanilla dataset.
- **NVL (S)**: Non-Vanilla and Log Transformation (similarities), has extra features from applying log transformation on the non-vanilla dataset.
- **NVL (SF)**: Non-Vanilla and log transformation (similarities and frequencies), has extra features from applying log transformation on the non-vanilla dataset.
- **NVIL (S)**: Non-Vanilla, interactions and log transformation (Similarities), has extra features from applying log transformation and interactions on the non-vanilla dataset.

- **NVIL (SF)**: Non-Vanilla, interactions and log transformation (Similarities and frequencies), has extra features from applying log transformation and interactions on the non-vanilla dataset.

The finalized datasets contain 6159 compound words out of which 4723 and 1436 are positively and negatively labelled respectively.

4 Experimental Approach

Our approach in the verification of our hypothesis is to implement three simple models, namely Logistic Regression, SVM, as well as an ensemble AdaBoost model. We subject them to classifying the compositionality of the constituents to the meaning of the compound word in the presence of different features. This allows for the possible identification of the features that contribute most to the performance in this task.

4.1 Logistic Regression

The logistic regression model is a probabilistic, discriminative model that directly models the conditional probability $P(y|x)$, that is the probability of seeing a class y given an input sequence of features x. This is accomplished by approximating a log-odds ratio as a linear function, and subsequently using this ratio as the decision boundary between two classes (Hastie et al., 2001).

4.2 SVM

Support Vector Machines are another method of creating a decision boundary in the feature space in order to predict the class given a sequence of features. These models strive to determine the decision boundary that maximizes the distance between two classes. This boundary is represented by data points in the training data, called the support vectors. These are the points that lie directly on the boundary, and as such specify the boundary itself, and are sufficient to make predictions(Hastie et al., 2001). Their use of support vectors also allow kernels to be used, which can project the features into higher dimensional spaces without explicitly having to compute these new features, thus allowing for very complex decision boundaries to be learned without too big of an increase to the computational cost (Shalev-Shwartz and Ben-David, 2014).

4.3 AdaBoost

The AdaBoost algorithm, explored by Freund and Shapire (1997), is a method that ensembles many weak learners employing the use of boosting. These weak learners are trained in a complementary manner, such that each are efficient at predicting only a portion of the data, and their shortcomings are then compensated for by other learners. This is ensured by boosting the probability mass of the misclassified examples as compared to those classified properly, thus inducing a higher loss in the following models if they also misclassify those points. The probability mass of each example is increased or decreased based on the error rate of the weak learner trained on the current dataset, and this process is repeated for many iterations. In our implementation, we use decision stumps as the weak learning model. These weak learners' predictions are then combined, weighed by their error weights in order to produce the final prediction which allows for the final model to have overcome the low variances of its individual components in order to provide good predictions(Shalev-Shwartz and Ben-David, 2014).

4.4 Model Selection

The selection process for each of the aforementioned models is done through a grid-search approach over a subset of each models' respective hyperparameters. The optimal hyperparameters for a given model are determined through a 5-fold cross validation on each of the 10 datasets mentioned in 3.4. This process is performed for all three models being studied. The hyperparameters being tuned are the following:

- **Logistic Regression**: Penalty p sets the type of regularization applied to the model, regularization coefficient C which is the inverse of regularization strength, tolerance t the tolerance for the stopping criteria, class weights cw that adjust weights associated with classes.

- **AdaBoost**: Number of estimators num_est the number of estimators ensembled, learning rate lr shrinks the contribution of each individual estimator.

- **SVM**: Penalty coefficient c which is inversely proportional to the penalty being applied, kernel k the type of kernel used by the model.

Model	V	VI	VL	VIL	NV	NVI	NVL (S)	NVL (SF)	NVIL (S)	NVIL (SF)
Logistic Regression	p: None C: 1000 t: 10000 cw: None	p: None C: 1000 t: 100 cw: Balanced	p: None C: 1000 t: 10000 cw: None	p: None C: 1000 t: 100 cw: Balanced	p: None C: 1000 t: 10000 cw: None	p: None C: 1000 t: 100 cw: Balanced	p: None C: 1000 t: 10000 cw: None	p: None C: 1000 t: 10000 cw: None	p: l2 C: 1 t: 100 cw: Balanced	p: l2 C: 1 t: 100 cw: Balanced
SVM	c: 2 k: rbf	c: 3 k: rbf	c: 2 k: rbf	c: 1 k: rbf	c: 2 k: rbf	c: 3 k: rbf	c: 3 k: rbf	c: 3 k: rbf	c: 3 k: rbf	c: 3 k: rbf
AdaBoost	num_est: 200 lr: 1.0	num_est: 200 lr: 1.0	num_est: 200 lr: 1.0	num_est: 200 lr: 1.0	num_est: 200 lr: 1.0	num_est: 200 lr: 1.0	num_est: 200 lr: 1.0	num_est: 200 lr: 1.0	num_est: 200 lr: 1.0	num_est: 200 lr: 1.0

Table 1: Model's optimal hyperparameters

Model	V	VI	VL	VIL	NV	NVI	NVL (S)	NVL (SF)	NVIL (S)	NVIL (SF)
Logistic Regression	0.423	0.499	0.423	0.500	0.443	0.562	0.443	0.443	0.562	0.563
SVM	0.423	0.423	0.423	0.423	0.443	0.472	0.443	0.443	0.466	0.467
AdaBoost	0.448	0.441	0.441	0.441	0.504	0.497	0.504	0.504	0.497	0.497
Random	0.447	0.462	0.442	0.461	0.446	0.461	0.468	0.459	0.464	0.454

Table 2: Model's performance with respect to F1 Macro-averaging

The final evaluation of the performances of each different type of model is then performed on the same 90% training, 10% validation split on the different datasets. This ensures that the relative differences in performance of the models being reported on a given dataset may not be attributed to differing training and evaluation conditions. A random predictor is also implemented as a baseline and reports measures on the same splits and datasets.

The evaluation measure chosen to both select models during the cross-validation process as well as to report the results on the various datasets is the F1 macro measure. This decision is informed by the imbalance in the distribution of the classes. Due to the F1 macro attributing equal weight to both classes regardless of proportions (Lipton et al., 2014), it provides the best notion of performance in the current case.

The hyperparameters that are found to be optimal through this pipeline for the various datasets are presented in Table 1 above.

5 Results

5.1 Impact of similarity scores

We analyze the performance of our models on the datasets described in 3.4 from Table 2. The general trend between the V datasets and the NV datasets, the latter having the similarity scores as features, is that under the same category of feature engineering, all three models perform better on the task when the dataset contains similarity scores. This is seen by comparing the scores V to NV, VI to NVI, VL to NVL(S) and NVL(SF) and so on.

5.2 Impact of engineered features

Looking more closely at the impact of different engineered features within the V and NV datasets, we notice that in the case of logistic regression models, their performance is highest when both interactions and log transformations are added or when only interactions are added. Indeed, VIL and VI provide the best results among the V datasets, as do NVIL and NVI in the NV datasets. The same is perceivable for the SVM models, but only in the NV datasets, it maintains poor performance across the V datasets. AdaBoost is fairly indifferent to the engineered features, showing very minimal decreases in performance when subjected to a dataset with augmentation I or IL.

5.3 Performance against baseline

In general, models trained on V datasets perform poorly compared to the random baseline, with the exception of Logistic Regression on the VI and VIL datasets. As far as the NV datasets are concerned, AdaBoost consistently beats the random baseline performances, with its reported measurements varying very little from one to the other. Logistic regression and SVM on the other hand, handily surpass the baseline in the presence of interactions, but do poorly on NVL(S) and NVL(SF) where only log transformations are added to the features.

6 Discussion and Conclusion

The results seem to point towards the notion that similarities as features have a positive impact of the performance of the models, independent of the feature engineering methods employed. Furthermore, the almost identical scores on both the dataset that only applies both feature augmentations to the similarity scores NVIL(S) and the dataset that that applies it to the frequencies as well NVIL(SF) reinforce the belief that the increase in performance seem to independent of the frequencies features.

In terms of the impact of the engineered features, as far as the AdaBoost model is concerned, its indifference to interactions and engineered feature augmentations in general seems to provide insight into the lack of complexity it can handle. Logistic regression and SVMs seem to be able to use the extra complexity and non-linear dependencies introduced by first order interactions, as evident from the considerable increase in performance they acquire. The log transformation indifference that these models seem to have may possibly be due to being dependant on the original features and thus not providing additional complexity.

Our process did suffer from a few limitations, notably a fairly small dataset with a big imbalance in class distribution. Furthermore, our datasets, even with the augmentations, were still lacking in the number of features, and having more complex feature vectors may have provided better performances from the models as a whole.

We conclude that enhancing a simple pre-existing closed compound word dataset with word2vec similarities does improve the performance of models in assessing the compositional meaning of a compound word from its two constituents. We further discern that most of the advantage of these similarities are brought forth by the first order interactions between them. This suggests the possibility of the similarities', and thus the compound words constituents', interdependence playing an essential role in informing the meaning of the compound word as a whole, which would be an interesting avenue to further investigate.

7 Statement of Contribution

All authors contributed to devising the action plan, gathering related works, performing the experiments, and writing the final manuscript.

References

Corina Dima and Erhard Hinrichs. 2015. Automatic noun compound interpretation using deep neural networks and word embeddings. In *Proceedings of the 11th International Conference on Computational Semantics*, pages 173–183, London, UK. Association for Computational Linguistics.

Changyong Feng, Wang Hongyue, Naiji Lu, Tian Chen, Hua He, Ying Lu, and Xin Tu. 2014. Log-transformation and its implications for data analysis. *Shanghai archives of psychiatry*, 26:105–9.

Yoav Freund and Robert E Schapire. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119 – 139.

Christina L. Gagné, Thomas L. Spalding, and Daniel Schmidtke. 2019. Ladec: The large database of english compounds. *Behavior Research Methods*, 51:2152–2179.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2001. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA.

Michael N. Jones and D. J. K. Mewhort. 2004. Case-sensitive letter and bigram frequency counts from large-scale english corpora. *Behavior Research Methods, Instruments, & Computers*, 36(3):388–396.

Zachary C. Lipton, Charles Elkan, and Balakrishnan Naryanaswamy. 2014. Optimal thresholding of classifiers to maximize f1 measure. In *Machine Learning and Knowledge Discovery in Databases*, pages 225–239, Berlin, Heidelberg. Springer Berlin Heidelberg.

Thomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Distributed representations of words and phrases and their compositionality. In *Neural and Information Processing System (NIPS)*.

Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013b. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Shai Shalev-Shwartz and Shai Ben-David. 2014. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, New York, NY, USA.

Lucy Vanderwende. 1994. Algorithm for automatic interpretation of noun sequences. In *COLING 1994 Volume 2: The 15th International Conference on Computational Linguistics*.