# Detecting semantic change in sarcasm

**Samarendra Chandan Bindu Dash**
University of Toronto
dashsam1@cs.toronto.edu

## Abstract

In Distributional Semantics, we train vector representation for words (called word-embeddings) by maximizing the probability of occurrence with co-occurring words. This allows the embeddings to learn interesting syntactic as well as semantic properties. In the study of semantic shifts of words, word-embeddings are frequently used to track the shift in the meaning of a word. In Irony/Sarcasm, people usually use positive semantic words with negative situation words/phrases or vice versa. This makes sentiment words occur in a different neighborhood than usual. In this paper, we analyze the possibility of detecting this shift between embeddings trained on sarcastic and non-sarcastic corpora.

## 1 Introduction

Sarcasm detection is generally a very difficult task because people usually conceal their true intention by using an overly positive or negative expression. It is easier to detect sarcasm in a spoken context because we have verbal cues to identify the true intention. (Peters, 2018; Bharti et al.) Lack of verbal cues in the written text doesn't make the intention immediately apparent.

One of the prominent methods of sarcasm detection is to identify the positive/negative contrast between the situation and the sentiment to identify the concealed emotion. (Bharti et al.; Riloff et al., 2013) classify phrases in the sentences as positive sentiment phrases and negative situation phrases. Then a sentence is classified as sarcasm when a positive sentiment is in close proximity to negative situation phrases. (Van Hee et al., 2018b) follows a slightly different approach, where they use an existing lexico-semantic database to identify contrastive emotion in a sentence to detect irony. The common theme here is that in irony/sarcasm, an emotion expressing a word doesn't appear in the expected neighborhood.

In computational linguistics, we generate a distributional representation of a word by using its context words (Mikolov et al.). Using this we get a vector representation of the semantic information encoded in the word. Hence, in theory, if we train word vectors in sentences that convey sarcasm, its vector representation should encode different semantic information. In this paper, we test this hypothesis and find out whether the semantic change between the word used in a general context vs the word used in a sarcastic context is statistically significant.

## 2 Related Works

The following papers and ideas helped design the pipeline for this research project.

### 2.1 Polarity Contrast In Sarcasm

To the best of the author's knowledge, training word embeddings to identify semantic shifts between sarcastic and non-sarcastic contexts haven't been attempted before. Still this concept of "Positive Sentiment Words in Negative Situation phrase" is played around a lot for sarcasm detection as well as sarcasm generation tasks. Given that over 72% of sarcasm is denoted by polarity contrast (Van Hee et al., 2018a), using this contrast is one of the prominent techniques in the "NLP in sarcasm" community.

#### 2.1.1 Sarcasm Detection

(Bharti et al.) uses a parsing-based method to detect sarcasm in tweets. The model does a bottom-up read of the parse tree of the sentence, marking different phrases and words as positive, negative, and neutral sentiments. Then the ratio of the total positive score and the total negative score is used to identify sarcasm. (Riloff et al., 2013) does pattern matching and pos tagging to identify a list of positive sentiment phrases and negative situation phrases. A tweet is identified as sarcastic if

a positive phrase is in close proximity to a negative situation. One interesting thing to note here is, both of these methods have accuracy in the range of 70% which matches closely to the distribution of polarity contrast bases sarcastic tweets in the SemEval2018 dataset (Van Hee et al., 2018a).

#### 2.1.2 Sarcasm Generation

Similarly, in sarcasm generation tasks, one of the prominent methods is to replace a negative sentiment phrase with a positive sentiment phrase to generate a sarcastic sentence. The papers (Mishra et al., 2019; Zhu et al., 2019) have implemented this idea. But it is not as straightforward as replacing a negative word with it's semantically opposite word. (Mishra et al., 2019) first neutralizes the statement by removing sentiment expressing words, then adds phrases with strong positive sentiment followed by negative situation. (Zhu et al., 2019) points out that this idea of swapping negative sentiment words with positive sentiment words is not viable for converting every non-sarcastic statement into a sarcastic statement.

### 2.2 Semantic Shift Identification

For identifying semantic shift, inspiration is taken from (Rosenfeld and Erk, 2018; Kim et al., 2014). In (Rosenfeld and Erk, 2018), the authors train a static word-embedding across all the time periods and then run it through a linear transformation (non-linear w.r.t time index) to generate temporal embeddings. This was found to be a good model for retrofitting pretrained word-embeddings on different corpora with different context window sizes. (Kim et al., 2014) proposes an idea where the word-embedding trained on the previous time period is passed as initialization for the word-embedding training for the next time period. This prevents the need for aligning word-embeddings trained across different time periods. As our dataset is less and training additional linear transformation for aligning word-embeddings usually introduces more noise to the system, the idea proposed in this paper was helpful for this project.

## 3 Data and Other Materials

The python library snsscrape [1] was used to scrape tweets. One million (1M) tweets were scraped that contained the hashtags *#irony* and *#sarcasm* to form the sarcasm dataset. Then one million tweets were scraped that didn't contain the hashtags *#irony* and *#sarcasm* to form the non-sarcasm dataset. Replies or Retweets were excluded.

For preprocessing the steps suggested at (QuaXP) are followed and some additional preprocessing steps are added to suit the objective. URLs and usernames are removed from the tweets. The '#' symbol is removed from hashtags and the rest of the tag is kept. Numbers are removed from the tweets and 0 is replaced with the word 'zero' (as it indicates strong sentiment). Camelcased words are split into their individual units. In addition to these, *#irony* and *#sarcasm* are removed from the tweets to avoid data leaks to the model (as these two hashtags are the main differentiator between the non-sarcastic and sarcastic corpora).

For calculating word-pair similarity MEN dataset (Bruni et al., 2013) was used. For the sentiment score of words, the synset_scores dict provided by the python library senti_classifier was used [3]. Their use is explained in detail in section 4.

## 4 Methodology

In this section, the overall methodology is explained. The exact values of the hyperparameters are discussed in section 5.

The major problem with training word-vector in sarcasm corpus is that the corpus size is usually very small. Due to this, GloVe (Pennington et al., 2014) objective function was finalized for this experiment as it performs well even with less frequent words. But even GloVe requires too much training data if we are training it from scratch to give good-quality embeddings. The glove-twitter vectors available in the gensim package[2] were trained on 1B tweets. It is really difficult to get a sarcasm corpus that size. So, some additional techniques were adopted to get good results in this experiment.

### 4.1 Collecting cooccurrence count

For collecting the cooccurrence count, one-sided context windows were used. For (Pennington et al., 2014), ideally, for each word pair in the context window, the count should be incremented by one. But this cooccurrence matrix calculation can be extended by incrementing the count by some function of the distance between the word pairs. For this

[1] https://github.com/JustAnotherArchivist/snscrape

[2] https://radimrehurek.com/gensim/apiref.html

paper, inverse $\left(\frac{1}{d}\right)$, inverse of log $\left(\frac{1}{1+log_{10}(d)}\right)$ and inverse of square root $\left(\frac{1}{\sqrt{d}}\right)$ were explored.

The collected cooccurrence matrix is not symmetric. An entry $A_{i,j}$ denotes the count when the word $W_i$ precedes $W_j$ in the context window.

To identify the semantic shift, a subset of words was selected based on their frequency in both the corpus and their sentiment score[3]. For this subset of words (say $S_w$), the cooccurrence count in the sarcastic corpus and the non-sarcastic corpus was recorded under different rows and columns in the cooccurrence matrix. For the rest of the words, no distinction was done between the corpora while collecting the counts.

### 4.2 Learning Linear Transformation on pre-trained vectors

In this step, the glove-twitter-50 vectors were taken and a linear transformation was learned on these vectors to retrofit the corpus[4]. The linear transformation is inspired from (Rosenfeld and Erk, 2018). In the embedding layer, the pretrained glove-twitter-50 was passed. As there is no requirement for the time component, a 1D randomly initialized tensor was used to replace it.

This linear transformation was trained on the cooccurrence data collected from the combined corpus of non-sarcastic and sarcastic tweets. The output was then stored to be used in the next step.

To determine the quality of the embeddings, the MEN dataset (Bruni et al., 2013) was used. In the MEN dataset, for 3000 word pairs, the human-scored similarity is given. We determine the quality of embeddings by the Pearson correlation of the similarity score with the dot product of the embeddings for the corresponding words. The epoch at which the model scores the highest correlation value is stored to be used in the next step.

### 4.3 Training Word-Embeddings and calculating semantic shift

In this step, the output from the linear transformation obtained from the previous step was used to initialize the word-embeddings. All the word-embeddings are not trained. Only the words that were the member of the subset $S_w$ (collected in section 4.1) were trained.

During training the word vectors were frozen, except for the words in the selected subset. But the corresponding bias terms for the words (both in the subset and out of the subset) were not frozen. This is because the bias vectors encode the normalizing factor for the word frequency and adding them ensures that the difference in results is not due to the change in frequency between the corpora appendix A.

One important part to note here is, as we are using the same initialization for training word-vectors in both the corpus and except for a select subset of words rest of all the vectors are frozen, we don't need to align the vectors to detect the shift (Kim et al., 2014).

Cosine distance was used to detect the semantic shift between the word-vectors.

## 5 Experimental Setup

### 5.1 Hyperparameters for the experiment

For this experiment, a context window of size 20 was selected. This is because, in a sarcastic sentence, the situation phrase can be very far from the sentiment word. For the same reason inverse of the log function was selected for the cooccurrence count(appendix A). [5]

For shortlisting words, any word that occurs less than 10 times in either of the corpora is removed. After the filtering, we were left with little over 13K words.

For training GloVe embeddings, the $x_{max}$ value was selected as 100 and the $\alpha$ value was selected as 3/4. For each cooccurrence count the importance given to learning that count is determined by the function $(\min(X_{i,j}/x_{max}, 1))^{\alpha}$. The values for these are kept the same as the original paper (Pennington et al., 2014).

### 5.2 List of experiments

The skeleton of the experiment is as follows,

> Given two corpora say $C_1$ $C_2$ and a subset of words say $S_w$, identifying whether a significant semantic change occurred for the words in the subset.

For our hypothesis, $C_1$ is the non-sarcastic corpus, $C_2$ is the sarcastic corpus and $S_w$ is a list of sentiment words. But to ensure that the method is able to reliably identify the semantic shift, different

[3] https://github.com/kevincobain2000/sentiment_classifier (Cagan et al., 2014)

[4] To understand the reasoning behind this step check appendix B

[5] More explanation on this can be found in appendix C

combinations of $C_1, C_2$, and $S_w$ need to be tested to ensure that the method doesn't detect a semantic shift when there isn't any.

| Ex. No. | Corpus $C_1$ | Corpus $C_2$ | Word Subset ($S_w$) | Semantic shift expected? |
|---|---|---|---|---|
| 1 | Non Sarcastic | Sarcastic | Sentiment Words | Yes |
| 2 | Non Sarcastic | Sarcastic | StopWords | No |
| 3 | Non Sarcastic | Non Sarcastic | Sentiment Words | No |
| 4 | Sarcastic | Sarcastic | Sentiment Words | No |

Table 1: Experiments

As mentioned in section 3, 1M tweets were scrapped each for both non-sarcastic and sarcastic corpus. But for this experiment, only corpora of size 300,000 were used. Hence for situations where two non-sarcastic or two sarcastic corpora were required, 600,000 tweets were randomly sampled from the 1M tweets and were split into two parts to get $C_1$ and $C_2$.

Due to the training of word-embeddings being a stochastic process, the cosine distance between two unique corpus will never be exactly 0. So, the test is, the cosine distance observed when semantic shifts are expected should be much higher than the cosine distance observed when semantic shift is not expected.

## 6 Results

First, experiment 1 was carried out. 16 semantic words were shortlisted based on the previously mentioned criterion. Those were,

good, love, bad, like, sad, sorry, mean, enjoy, congratulations, happiness, worst, dog, excellent, soft, praise, anger

| words | cosine distance |
|---|---|
| good | 0.1831074953 |
| love | 0.275005877 |
| bad | 0.3469482064 |
| like | 0.2209181786 |
| sad | **0.5025382042** |
| sorry | 0.2574074864 |
| mean | 0.1569036841 |
| enjoy | 0.3361491561 |
| congratulations | 0.3316889405 |
| happiness | **0.5472163856** |
| worst | 0.3571274281 |
| dog | 0.1515259147 |
| excellent | 0.2340017557 |
| soft | 0.3603383899 |
| praise | 0.1637061238 |
| anger | 0.3334697485 |

Table 2: Cosine distance of sentiments words in sarcastic and non-sarcastic corpus

For comparison experiment 2 was carried out. The following stop-words were selected,

the, i, to, a, and, you, is, of, for, it, in, that, so, this, my, me

| words | cosine distance |
|---|---|
| the | 0.1572327614 |
| i | 0.200289309 |
| to | 0.1675726175 |
| a | 0.162278235 |
| and | 0.1913046241 |
| you | 0.1971036792 |
| is | 0.1713368297 |
| of | 0.15190202 |
| for | 0.1810332537 |
| it | 0.1622725725 |
| in | 0.1674274802 |
| that | 0.1566486359 |
| so | 0.2660787702 |
| this | 0.1927073002 |
| my | 0.272929728 |
| me | 0.3322527409 |

Table 3: Cosine distance of stopwords in sarcastic and non-sarcastic corpus

| words | cosine similarity |
|---|---|
| blessings | 0.7297375202178955 |
| positivity | 0.6988093256950378 |
| joy | 0.665351390838623 |
| positive | 0.6602541208267212 |
| wishes | 0.6509174108505249 |
| creativity | 0.6494938731193542 |
| gratitude | 0.6480365991592407 |
| success | 0.6375394463539124 |
| blessing | 0.6363803148269653 |
| thankful | 0.6360041499137878 |

Table 4: Top 10 Semantic neighbors of happiness for the embedding trained in non-sarcastic corpus

| words | cosine similarity |
|---|---|
| misery | 0.8019948601722717 |
| sorrow | 0.7789912223815918 |
| sadness | 0.7452418804168701 |
| imagination | 0.7295545935630798 |
| endless | 0.728373110294342 |
| clarity | 0.7210588455200195 |
| oxygen | 0.7202736735343933 |
| passion | 0.7200639843940735 |
| guilt | 0.7200138568878174 |
| confusion | 0.7199685573577881 |

Table 5: Top 10 Semantic neighbors of happiness for the embedding trained in sarcastic corpus

| words | cosine similarity |
|---|---|
| feel | 0.8744090795516968 |
| so | 0.8569358587265015 |
| mad | 0.8508458137512207 |
| im | 0.8507384657859802 |
| like | 0.8433355689048767 |
| reason | 0.841853141784668 |
| bad | 0.8377725481987 |
| really | 0.8344084620475769 |
| crazy | 0.83233243227005 |
| damn | 0.8322743773460388 |

Table 6: Top 10 Semantic neighbors of sad for the embedding trained in non-sarcastic corpus

| words | cosine similarity |
|---|---|
| irony | 0.7705574035644531 |
| ignorance | 0.7524101734161377 |
| liberal | 0.7508460879325867 |
| shameful | 0.7338619232177734 |
| disgrace | 0.7261499762535095 |
| denial | 0.7249147295951843 |
| ignorant | 0.7209689021110535 |
| cruel | 0.719818651676178 |
| racist | 0.7192907333374023 |
| unacceptable | 0.7146875858306885 |

Table 7: Top 10 Semantic neighbors of sad for the embedding trained in sarcastic corpus

The result are presented in table 2 and table 3 respectively.

The cosine distances for the stopwords were in the range of $(0.15, 0.34)$, which is expected because the stopwords don't carry any semantic shift. For the sentiment words, although the cosine distance is on average a little bit higher, it still stays in the same range.

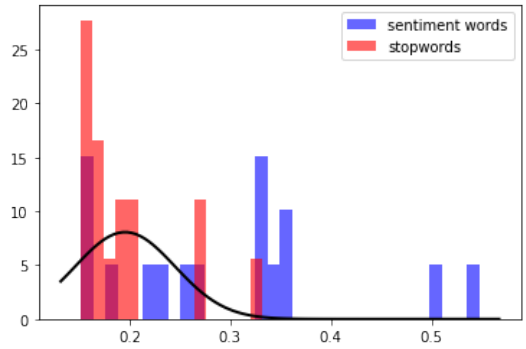


Figure 1: Plot of distribution of cosine distance

In figure 1 the histogram of the cosine distance is plotted. The mean and standard deviation of the cosine distance for the stopwords are $0.196$ and $0.049$ respectively. We can see that most of the sentiment words are under the range where the values can arise out of randomness. The two exceptions (the two points on the far right of the graph) were 'Happiness' and 'Sad'.

Happiness saw the highest semantic shift between the sarcastic and the non-sarcastic corpus.

The word-vector of happiness in the non-sarcastic corpus is closer to positive sentiment words like 'blessings', 'positivity' etc. as shown in table 4. On the contrary, in the sarcastic corpus, the trained vector is closer to words like 'misery', and 'sorrow' as shown in the table table 5[6].

For the word sad, the results are a bit interesting. While the semantic neighbors in both cases are negative sentiment words, there is a sharp contrast between the type of words. In table 7, we can see words like 'irony', 'ignorance', 'disgrace' etc. I believe the reason behind this is, sad is being used to show negative emotions regarding a contrastive fact. For E.g. "This is such a sad read. Glad weve got a new super rich PM who has made his wealth from banking and gambling against the pound.", "So sad Billionaires only worth trillion lost billionaire status". Here people are using the word 'sad' to denote their negative feeling towards a seemingly contrastive fact presented as an ironic statement. I think the word 'sad' is an exception due to this kind of use in an ironic statement.

For the rest of the words, the semantic shift isn't significant. Their semantic neighbors also stayed

[6]Remember, except for the said 16 words the word vectors for rest of the words are not changed. So the findings show that the vector for happiness in the sarcastic corpus is closer to words like 'misery' in the general sense.

similar to their non-sarcastic counterparts. Therefore this hypothesis was found to be a failure. As experiment 1 was a failure, the rest of the experiments were not run.

*The code to replicate the analysis can be found here*[7].

## 7 Discussion

I believe the reason for failure is the tendency of people to use overly positive ornamentation to represent sarcasm. Ideally, for my hypothesis, examples such as,

"I love migraines"

i.e. where the word denoting the actual emotion (here 'hate') is replaced with its semantic opposites would have worked the best. Unfortunately, people usually use overly positive terms in sarcasm to make the humor land. One such example is shown in figure 2.

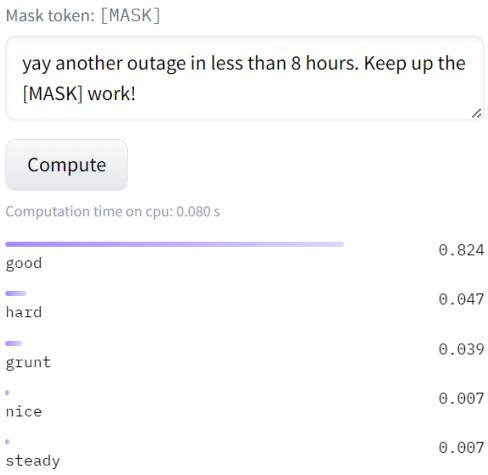


Figure 2: Masked word prediction on a sentence using BERT

In figure 2 BERT large model (Face) was used for masked word prediction. The original sentence was, "yay another outage in less than 8 hours. Keep up the good work'. We know it's a sarcastic statement because of the word 'outage' being used with the sentiment 'good'. Even though the statement shows a negative emotion the BERT model still ends up predicting the correct sentiment word. This is because, if we see the context of 'good', it also has words like 'yay', and 'keep up' which usually denote a highly positive emotion. So the language model can easily predict the correct sentiment word. These overly positive sentiment words will also interfere with the word-embedding training as it also relies on the context words to train the embedding. This is why I believe most of the sentiment words didn't show any semantic shift. Because their neighbors stayed more or less identical even in sarcasm corpus.

It is not to say that the method won't work at all. As we saw from our experiments the word 'happiness' yielded the expected results. But due to the overly positive ornamentation people use with sarcasm, training word-vectors without any modifications might not be 100% reliable.

One possible way to offset this could be to someway identify the words or phrases that describe a situation and give more weight to those cooccurrence counts. (Riloff et al., 2013) does pattern matching and POS tagging to identify these phrases. Using similar techniques in word-embedding training might be worthwhile to explore in the future.

[7] github.com/Samarendra109/Detecting_Semantic_Shift_In_Sarcasm

## References


Santosh Kumar Bharti, Korra Sathya Babu, and Sanjay Kumar Jena. Parsing-based sarcasm sentiment recognition in twitter data. In *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*.

E. Bruni, N. K. Tran, and M. Baroni. 2013. Multimodal distributional semantics. In *Journal of Artificial Intelligence Research 49*, pages 1–47.

Tomer Cagan, Stefan L. Frank, and Reut Tsarfaty. 2014. Generating subjective responses to opinionated articles in social media: An agenda-driven architecture and a Turing-like test. In *Proceedings of the Joint Workshop on Social Dynamics and Personal Attributes in Social Media*, pages 58–67, Baltimore, Maryland. Association for Computational Linguistics.

Hugging Face. Bert large language model. Accessed: 2022-12-11.

Yoon Kim, Yi-I Chiu, Kentaro Hanaki, Darshan Hegde, and Slav Petrov. 2014. Temporal analysis of language through neural language models. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*, pages 61–65, Baltimore, MD, USA. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *ICLR Workshop Papers*.

Abhijit Mishra, Tarun Tater, and Karthik Sankaranarayanan. 2019. A modular architecture for unsupervised sarcasm generation. In *Proceedings of*

*the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6144–6154, Hong Kong, China. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Sara Peters. 2018. Why is sarcasm so difficult to detect in texts and emails?

QuaXP. Simple tweet preprocessing.

Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert, and Ruihong Huang. 2013. Sarcasm as contrast between a positive sentiment and negative situation. In *The 2013 conference on empirical methods in natural language processing*, pages 704–714.

Alex Rosenfeld and Katrin Erk. 2018. Deep neural models of semantic shift. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 474–484, New Orleans, Louisiana. Association for Computational Linguistics.

Cynthia Van Hee, Els Lefever, and Véronique Hoste. 2018a. SemEval-2018 task 3: Irony detection in English tweets. In *Proceedings of the 12th International Workshop on Semantic Evaluation*, pages 39–50, New Orleans, Louisiana. Association for Computational Linguistics.

Cynthia Van Hee, Els Lefever, and V´eronique Hoste. 2018b. We usually don't like going to the dentist: Using common sense to detect irony on twitter. In *Computational Linguistics, 44(4)*, pages 793–832.

Mengdi Zhu, Zhiwei Yu, and Xiaojun Wan. 2019. A neural approach to irony generation. *arXiv preprint arXiv:1909.06200*.

# A GloVe Embeddings

The GloVe objective function to train word-embeddings was introduced by (Pennington et al., 2014). The objective function derivation is explained below.

Say, the word-embedings for the words $W_i$ and $W_j$ are $\hat{w_i}$ and $\hat{w_j}$ respectively. We want to learn a function $F$ such that

$$F(\hat{w_i}^T \hat{w_j}) = P(W_i|W_j) \tag{1}$$

where $P(W_i|W_j)$ is the probability of occurrence of $W_i$ given $W_j$.

For getting good performance on word-analogy tasks, $F$ must obey the following condition,

$$F((\hat{w_i} - \hat{w_j})^T \hat{w_k}) = \frac{F(\hat{w_i}^T \hat{w_k})}{F(\hat{w_j}^T \hat{w_k})} \tag{2}$$

The solution to equation (2) gives $F(.) = \exp$. Empirically $P(W_i|W_j) = \frac{X_{i,j}}{X_j}$. Putting that in equation (1) gives,

$$\hat{w_i}^T \hat{w_j} + log(X_j) = log(X_{i,j}) \tag{3}$$

But, as dot product is reflexive, $F(\hat{w_i}^T \hat{w_j}) = P(W_j|W_i)$ is also true. Putting the corresponding empirical value of probability gives us,

$$\hat{w_i}^T \hat{w_j} + log(X_i) = log(X_{i,j}) \tag{4}$$

Hence, the glove training objective is defined as (taking an average of equations (3) and (4)),

$$\hat{w_i}^T \hat{w_j} + b_i + b_j = log(X_{i,j}) \tag{5}$$

where $b_i, b_j$ are bias terms corresponding to the words $W_i$ and $W_j$. Here the bias terms correspond to the normalization factor in the empirical probability calculation.

## A.1 Cooccurrence count

The cooccurrence count is calculated by counting the number of times a word pair cooccurs in a context window. But one can instead increment the count by some function of the distance between the words. This is helpful because we give more importance to words that are closer to each other by using a function that decreases in value as the distance between the word pairs increases.

For this paper, the functions inverse $\left(\frac{1}{d}\right)$, inverse of log $\left(\frac{1}{1+log_{10}(d)}\right)$ and inverse of square root $\left(\frac{1}{\sqrt{d}}\right)$, were explored.

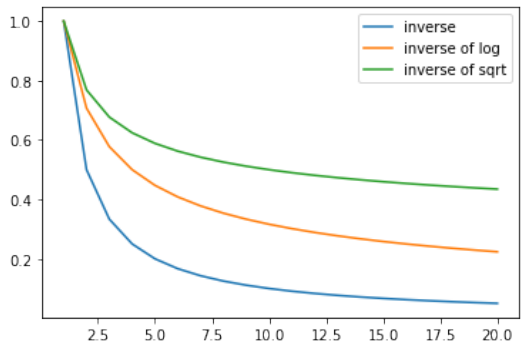


Figure 3: Distance Functions Plot

For the requirement of the paper, a large context window was required. This is because the context word can occur at a longer distance from the sentiment word. The inverse of the log function drops the slowest among all the three figure 3. Hence, the inverse of the log function with base 10 was selected for incrementing the cooccurrence count.

## B Linear Transformation as Model Initialization

In this section, the reasoning behind several design decisions is discussed.

### B.1 Using the output of transformation for shift detection

In the paper (Rosenfeld and Erk, 2018), static word-embeddings are trained on the data, and for each timestamp, a linear transformation w.r.t to these embeddings is trained to generate temporal word-embeddings. Then these transformed word-embeddings are used to detect the semantic shift. But in this paper that approach is not followed.

The reason is, when using a pretrained word-embedding in place of the static embedding, the said approach fails the word switch test. In the word-switch test, we pick two random words, say $W_a$,$W_b$ (preferably with equivalent frequency) and train the word-embeddings. The trained embeddings for $W_b$ should show the semantic neighbors of $W_a$ and vice versa. But the transformation model trained in this paper fails this test, i.e. the words $W_a, W_b$ retain their previous semantic neighbors. To understand the reason, check the example in figure 4.

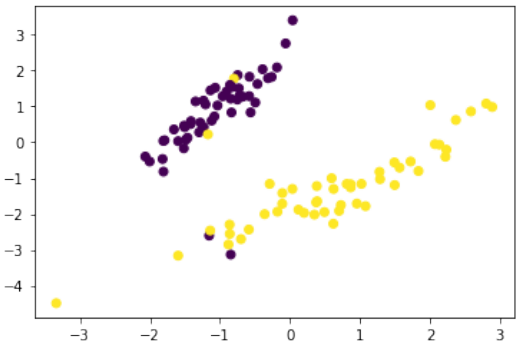


Figure 4: Classification Scatter Plot

If we try to learn a classifier on the data given in figure 4, the four points that are in the wrong clusters will just be ignored by the classifier as noise. I believe, in a similar manner, if a single word is taken out of its semantic neighborhood, then the transformation doesn't prioritize that word. This is because the word's new neighborhood is in complete contrast with its neighborhood in the input space, i.e. the pretrained vector embeddings. The original paper (Rosenfeld and Erk, 2018) doesn't face this problem, probably because they train their word-embedding from scratch over the complete corpus. But as we are using pretrained glove embeddings we are addressing this issue by selectively training word-embeddings for the words whose semantic shift we wish to track section 4.3.

### B.2 Using pretrained embeddings directly as initialization

One question that may arise is, "why to train a linear transformation on the glove embeddings and why not directly use the pretrained model as initialization?". And this idea was tested out. But once the step section 4.3 was executed, the semantic neighbors of the subset of words were very erratic. I believe this is because we are using a different size context window and different function for cooccurrence count. Learning a transformation to retrofit the glove embeddings on the context window and cooccurrence count used in training before using them as initialization mitigated this issue.

## C Large Context Window

A context window of size 20 was used because for some sentences, the negative situation phrase might be far away from the sentiment phrase. A large context window ensures that the negative situation phrase is not left out of the context window. But given that only 31% of the tweets are larger than 20 words and we are using a diminishing function to count the cooccurrence as the words are farther away, this large context window shouldn't negatively affect the results.