Measuring the Change in Semantic and Sentiment of Neologisms Against An Axis

Brooke A. Taylor
Undergraduate Researcher
Whitman College
Walla Walla, Washington 99362
Email: btaylor3.14159@gmail.com

Brendan Tracey
Mentor, Program Postdoctoral Fellow
Santa Fe Institute
Santa Fe, New Mexico 87501
Affiliation: Massachusetts Institute of Technology
Email: btracey@santafe.edu

Vanessa Ferdinand
Mentor, Omidyar Postdoctoral Fellow
Santa Fe Institute
Santa Fe, New Mexico 87501
Email: vanessa@santafe.edu

Abstract

Are today's neologisms indicating that we are all a bunch of pessimists? New words are produced at the rapid rate of about 15 per day and distributed online. Some words last only a week contained to a small circle of friends while others find a permanent residence in the dictionary. But, as with the words that came before them, they are subject to the force of language evolution. We track a set of neologisms in their first few years of life to see how their semantics and their sentiment may have changed over time by embedding them in a 100-dimensional vector space and comparing their placement with respect to a word axis. We analyze the effectiveness of this type of measurement neologism evolution, and find that specific axes provide a great tool to analyzing neologism sentiment and semantics, and also that neologisms seem more mobile in the vector space that common words.

1 Introduction

Newly coined words, or neologisms, while making up a very small fraction of the one million words in the English language, hold a lot of potential for giving insight into the mechanics of language evolution. According to the Global Language Monitor[1], a neologism is produced about every 98 minutes, totaling to about 5400 new words per year. However, only about 1000 of these ever make it into the dictionary.

This paper investigates ten neologisms that have been made popular in this decade, examining their evolution over their first few years of life, from 2011-2016. Identifying neologisms in a corpus is well

[1]http://www.languagemonitor.com/global-english/no-of-words/

studied [3-5], and there is also much qualitative discussion on why a neologism eventually might become a common word. However, very little quantitative work has been done in the way of seeing *how* these neologisms change before they become common words. We believe that by more closely inspecting a neologism's evolution in the first few years of its use, we could understand why they shift the way we do. Ultimately this might be able to shed light on what might cause a neologism to last in comparison to the ones that do not make it.

We analyze the change in words with a tool called Word2Vec, introduced by Mikolov et al. [1] in 2013. We then measured how neologisms move in a latent vector space by projecting them onto a word axis, such as *good-bad.* This allows us to see changes in sentiment (i.e. moving closer to *good* or *bad*), as well as defining a neologism's semantics, such as using the *boat-together* axis for the neologism *ship.*

2 Word2Vec Review

Word2Vec is a tool for natural language processing algorithms. It creates vector representations, or neural word embeddings, of words from unstructured text. The latent embeddings are designed so that similar words will be clustered closely together in that vector space, whether they are similar in definition or similar in context. For example, we would expect that the words *cat, dog,* and *hamster* might be close together as both mammals and house pets, and likewise that *homework, teachers*, and *stress* might be nearby as they pertain to ideas related to school.

In this project we use the word vectors output from Word2Vec as a tool to measure the change in a neologism over time, that is, how it will move in the vector space. Word2Vec can be run with multiple different models and training algorithms [1], but for this paper we will focus on the ones that were used for our program, provided by Deeplearning4j[2]. Other Word2Vec versions can be found at [6, 7].

2.1 Word2Vec's Skip-Gram Model

In order to learn the neural word embedding of each word from unstructured text, Word2Vec optimizes its learning algorithm using the Skip-gram model, introduced by Mikolov et al. [2]. The objective of the Skip-gram model, seen in Figure 3, is to learn word vectors, or word embeddings, that can accurately predict nearby words.

[2]https://deeplearning4j.org/word2vec

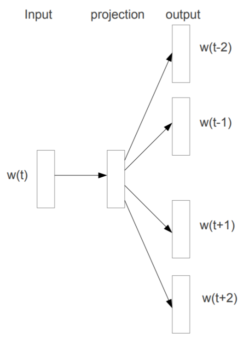
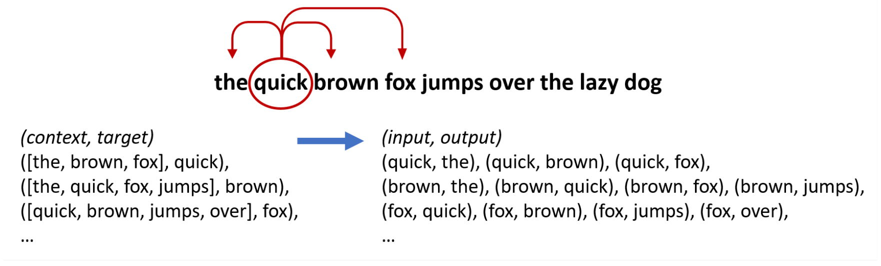


Figure 1: The Skip-gram model

The objective function is defined as

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c}\log p(w_{t+j}|w_t),$$

where T is the number of training words w and c is the context window size. For example, if $c = 1$, this would mean summing over the first word on each side of the center target word w_t, whereas if $c = 2$, this would mean summing over the first two words on each side of w_t. As a more concrete example, take the sentence "the quick brown fox jumps over the lazy dog". For $c = 2$, Figure 2 shows the context-target pairs for the first three words, and then converts them to input-output pairs that will be iterated over in the objective function.



(context, target)
([the, brown, fox], quick),
([the, quick, fox, jumps], brown),
([quick, brown, jumps, over], fox),
...

(input, output)
(quick, the), (quick, brown), (quick, fox),
(brown, the), (brown, quick), (brown, fox), (brown, jumps),
(fox, quick), (fox, brown), (fox, jumps), (fox, over),
...

Figure 2: Skip-gram Objective Function Word-Target Pairs

In the Skip-gram's objective function, $p(w_{t+j}|w_t)$ stands for the probability of the context word given the target word. This probability is assumed to have the following "softmax" form

$$\frac{\exp\left({\mathbf{v}'_{w_O}}^\top \mathbf{v}_{w_I}\right)}{\sum_{w=1}^{W}\exp\left({\mathbf{v}'_{w_O}}^\top \mathbf{v}_{w_I}\right)}.$$

Here $\mathbf{v}_{w_I}$ and $\mathbf{v}'_{w_O}$ stand for the input (I) and output (O) vectors of w, and W is the number of words in the vocabulary. We can see that nearby words will have a higher softmax contribution, making them more likely.

The difficulty with the softmax function is it requires computing the normalization constant, which requires summing over all pairs of words, making the computation time proportional to W, which for our data set was around 18,000-20,000 words. Thus, we instead look for an approximation to lessen the computation time.

2.2 Approximating the Softmax Function

A few suggested approximations to the softmax function include the hierarchical softmax, Noise Contrastive Estimation (NCE), and Negative sampling (NEG) [2]. According to a study done by Mikolov et al. [2], Negative sampling has been the most efficient approximation of the three.

Negative sampling has the objective function J_{NEG} defined as

$$\log \sigma({\mathbf{v}'_{w_O}}^\top \mathbf{v}_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[\log \sigma(-{\mathbf{v}'_{w_O}}^\top \mathbf{v}_{w_I}) \right].$$

Here, $\sigma(x)$ represents the binary logistic regression probability $1/(1+\exp(x))$. The left-hand term calculates the sum over k words chosen from a noise distribution $P_n(w)$, which represents random context words in the data set. In our concrete example "the quick brown fox jumps over the lazy dog", the word *sheep* would be a noise word so the input-output pair *(brown, sheep)* would be one we want to minimize the probability of, as opposed to *(brown, fox)* which is in the data set which we want to maximize the probability of. The goal of Negative sampling is to both maximize the probability of context-target pairs that are in the data set, and minimizing the probability of context-target pairs that are from the noise distribution. The expectation is typically approximated by Monte Carlo sampling.

2.3 Training the Skip-gram Model

The Skip-gram model, along with many other neural network models, is trained using stochastic gradient descent (SGD).

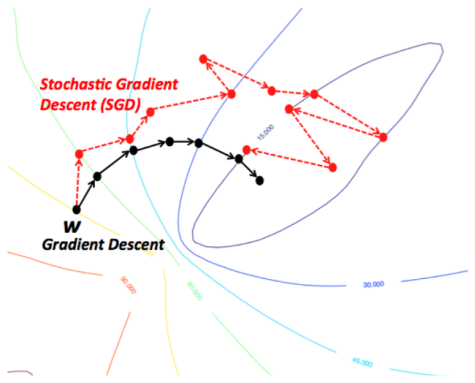


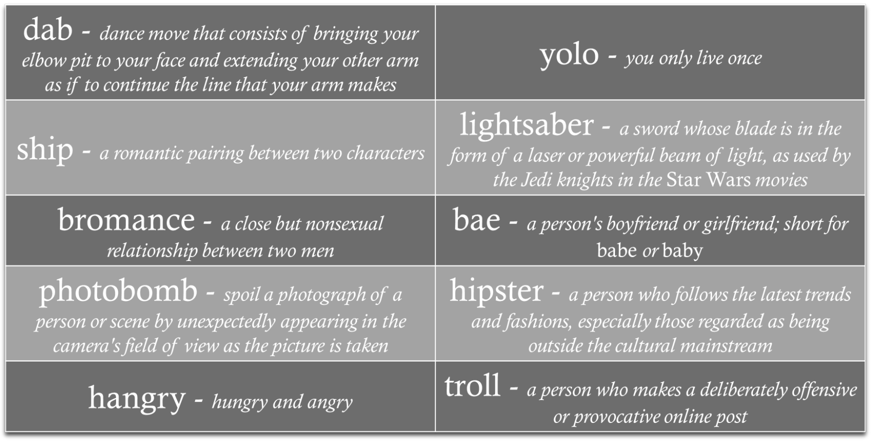
Figure 3: Stochastic Gradient Descent

SGD takes all of the input-output word vector pairs and splits the set into smaller batches of input-output pairs. It will then iterate over one batch, change the vector parameters to make a step toward the maximum of J_{NEG}, then iterates over another batch, and so on and so forth. Unlike normal gradient descent, which iterates over all pairs before making a change in parameters and eventually converges to the maximum, SGD instead converges very close to the maximum with careful implementation. For Word2Vec's objective, close to the maximum is good enough to create accurate word embeddings. Most importantly, this drastically decreases the computation time when working with very large data sets because it does not require iterating over the entire data set for each update to the vector parameters, and instead only a fraction of them.

3 Method

3.1 Selected Neologisms

For this study, we selected ten neologisms that have diverse characteristics described below. We also wanted these neologisms to be popular enough to have a high frequency of occurrences on Reddit. By having a large number of occurrences of these words in our training corpus we would then have more trustworthy word vectors given by Word2Vec.



dab - *dance move that consists of bringing your elbow pit to your face and extending your other arm as if to continue the line that your arm makes*	yolo - *you only live once*
ship - *a romantic pairing between two characters*	lightsaber - *a sword whose blade is in the form of a laser or powerful beam of light, as used by the Jedi knights in the* Star Wars *movies*
bromance - *a close but nonsexual relationship between two men*	bae - *a person's boyfriend or girlfriend; short for* babe *or* baby
photobomb - *spoil a photograph of a person or scene by unexpectedly appearing in the camera's field of view as the picture is taken*	hipster - *a person who follows the latest trends and fashions, especially those regarded as being outside the cultural mainstream*
hangry - *hungry and angry*	troll - *a person who makes a deliberately offensive or provocative online post*

Figure 4: The Ten Selected Neologisms

The neologisms are a collection of nouns, verbs, and adjectives, and each has some interesting properties. The words *dab, ship*, and *troll* are neologisms with double meanings—their definitions as neologisms, but also their definitions before they were neologisms (i.e. *troll* is both an ugly mythical being but also a person who deliberately offends). The words *bromance* and *photobomb* are emerging in popular culture. The word *yolo* is actually an abbreviation, though still treated as a word in its own right. The words *lightsaber* and *hipster* are not popular neologisms within the time frame we studied, but were neologisms in recent history, and were chosen to see how they might differ from the current popular neologisms.

3.2 Word Axis Measurement

Our method for measuring how a neologism changes over time in this vector space is what we will call a 'word axis measurement'. We take two words in our corpus that we believe have a stable relative location in the vector space and create an axis between them, and then compare how the neologism is located with respect to it and how this location shifts with time.

In Figure 5 below, the dashed lines represent the vectors given by Word2Vec for the neologism and two axis words, with locations in space N, A_1, and A_2, respectively. The perpendicular distance of the neologism to the word axis is then located at point I.

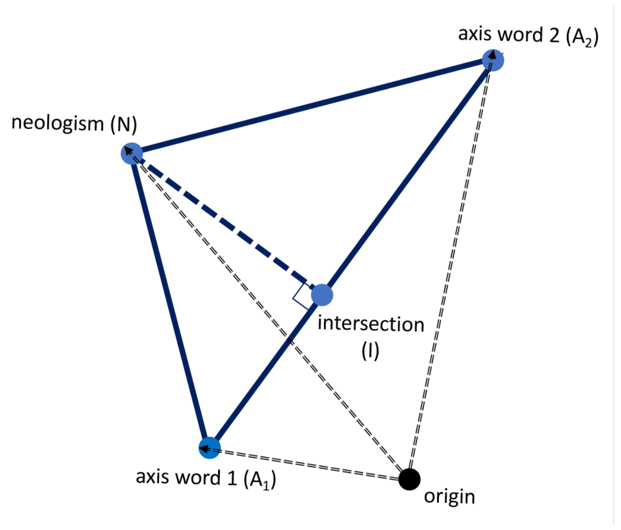


Figure 5: Geometric View of the Word Axis Measurement

The first value we calculate is the fractional distance of the neologism along the word axis:

$$\text{fraction of distance along axis} = \frac{\overrightarrow{A_1N} \cdot \overrightarrow{A_1A_2}}{\left|\overrightarrow{A_1A_2}\right|^2}.$$

If the neologism is oriented between the two axis words (as in the figure above), this returns a value between 0 and 1, where 0 is close to the axis word 1. It can also give us a negative value, indicating a neologism word vector oriented closer to axis word 1 and further away from axis word 2, or vice versa for values greater than 1.

The second calculation is the distance of the neologism away from the word axis:

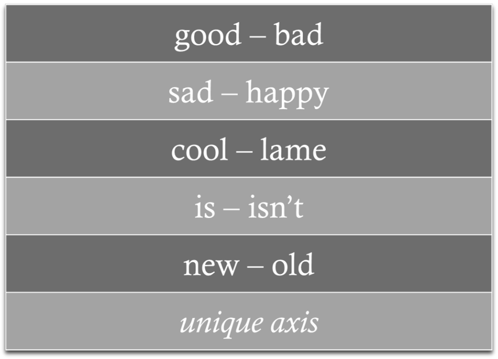
$$\text{fraction of distance away from axis} = \frac{\left|\overrightarrow{NI}\right|}{\left|\overrightarrow{A_1A_2}\right|}$$

This value gives us an idea of if the neologism is close to our word axis at all, because if it is not, then the first value we calculated is essentially meaningless.

By using this kind of measurement, we hope to be able to investigate a neologism's change in sentiment and semantics. We can choose axis words that are polar in sentiment to gauge a neologism's sentiment, or axis words that have different meanings to gauge a neologism's semantics. While this measurement does limit us to only comparing two words at a time, we hope this can lay the groundwork for how we can measure neologism evolution using a vector space.

3.2.1 Selected Axes

For each of the ten neologisms, six axes were chosen, five of which were used across all of the neologisms, and one which is uniquely relevant to the neologism.



good – bad
sad – happy
cool – lame
is – isn't
new – old
unique axis

Figure 6: The Six Selected Axes

The first three axes were chosen as typical positive-negative sentimental measurements. The *is-isn't* axis was meant to be a check to see if any double-negatives were appearing, such as a word being close to *cool* but also *isn't* indicating 'not cool'. The *new-old* axis was chosen to see if the contexts neologisms were used in indicated that people knew it was a new rather than old word. The unique axes for each neologism were chosen to investigate either their semantics or learn some other interesting feature of the word.

3.3 Word2Vec Text Input

Word2Vec trains on a file of unstructured text. While many large text files are available online [8, 9], such as Google News, this project required text input that contains many occurrences of neologisms, which often occurs in less formal contexts than Google News. Other requirements were that our source had an API to access its contents, and that it had a lot of content, as Word2Vec trains more accurately the more data it has to train on[3]. Thus, for this study we drew our data from Reddit[4].

3.3.1 PRAW

The package we used to access contents on Reddit was the Python Reddit API Wrapper[5]. For each file, we used Amazon CloudSearch to search for comments containing a neologism and that were posted in a specific time frame. For example, `(and timestamp:1293840000..1325375999 'bromance')` will search for instances of 'bromance' from January 1 to December 12 of 2011. Reddit would then return a list of subreddits, which we would iterate through, and retrieve all

[3]https://code.google.com/archive/p/word2vec/
[4]https://www.reddit.com/
[5]https://praw.readthedocs.io/en/latest/

submissions in that subreddit, and finally iterate through all the comments in each submission, recording all comments that contained the neologism to a text file.

We collected a total of 60 files: one per each of 10 neologisms and for each of the 6 years 2011-2016. We then ran each of the files through a data-cleaner to rid them of unnecessary punctuation, Reddit user or subreddit references, and hyperlinks to help limit noise data in training Word2Vec.

3.3.2 Compiling the Training Files

While it would be ideal to simply compile the data of all neologisms for one year into one file and train those total 6 files separately to see how the neologisms changed per year, a bit more thought was needed to get accurate results. Word2Vec trains more accurately when it 1) has a large data set to train on, and 2) sees a specific word frequently so it has plenty of references to learn from. Neologisms are much less frequent in the corpus than common words, and a trial run showed that training Word2Vec on the raw collected data resulted in unstable results. This meant that for the same file (i.e. the same year), the neologism's vector position fluctuated due to SGD, thus making it impossible to see how its position changed over time when it was not stable within the same time period.

To help combat these issues, the final input files were compiled as follows: the raw file that contained the specific neologism's comments for one year was doubled, creating a new file that had the original file's contents written twice. This will help alleviate both problems addressed above by providing not only more data but also twice as many occurrences of the neologisms. Then all other text files that did *not* pertain to the neologism (i.e. 55 of the 60 raw files) were appended to the doubled text file. A diagram of the first step of this process is seen in Figure 7 for a data set of four neologisms *hangry, hipster, lightsaber*, and *photobomb*. The diagram is then repeated for *hipster* where the red box surrounds all *hangry, lightsaber*, and *photobomb* files and each individual *hipster* file is added to the red box to make the new file, and then similarly repeated for *lightsaber* and finally *photobomb*.

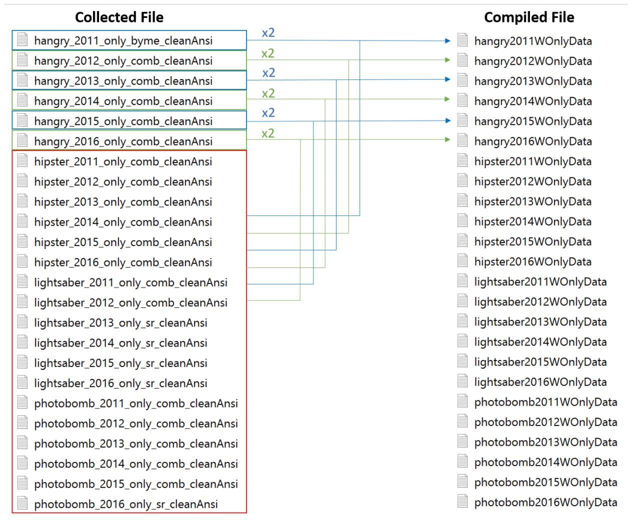

Collected File
hangry_2011_only_byme_cleanAnsi
hangry_2012_only_comb_cleanAnsi
hangry_2013_only_comb_cleanAnsi
hangry_2014_only_comb_cleanAnsi
hangry_2015_only_comb_cleanAnsi
hangry_2016_only_comb_cleanAnsi
hipster_2011_only_comb_cleanAnsi
hipster_2012_only_comb_cleanAnsi
hipster_2013_only_comb_cleanAnsi
hipster_2014_only_comb_cleanAnsi
hipster_2015_only_comb_cleanAnsi
hipster_2016_only_comb_cleanAnsi
lightsaber_2011_only_comb_cleanAnsi
lightsaber_2012_only_comb_cleanAnsi
lightsaber_2013_only_sr_cleanAnsi
lightsaber_2014_only_sr_cleanAnsi
lightsaber_2015_only_sr_cleanAnsi
lightsaber_2016_only_sr_cleanAnsi
photobomb_2011_only_comb_cleanAnsi
photobomb_2012_only_comb_cleanAnsi
photobomb_2013_only_comb_cleanAnsi
photobomb_2014_only_comb_cleanAnsi
photobomb_2015_only_comb_cleanAnsi
photobomb_2016_only_sr_cleanAnsi
x2
x2
x2
x2
x2
x2
Compiled File
hangry2011WOnlyData
hangry2012WOnlyData
hangry2013WOnlyData
hangry2014WOnlyData
hangry2015WOnlyData
hangry2016WOnlyData
hipster2011WOnlyData
hipster2012WOnlyData
hipster2013WOnlyData
hipster2014WOnlyData
hipster2015WOnlyData
hipster2016WOnlyData
lightsaber2011WOnlyData
lightsaber2012WOnlyData
lightsaber2013WOnlyData
lightsaber2014WOnlyData
lightsaber2015WOnlyData
lightsaber2016WOnlyData
photobomb2011WOnlyData
photobomb2012WOnlyData
photobomb2013WOnlyData
photobomb2014WOnlyData
photobomb2015WOnlyData
photobomb2016WOnlyData

Figure 7: Diagram of Compilation of Training File

The files on the right-hand side of the diagram then became the input files which Word2Vec trained on. In total this created 60 separate training files.

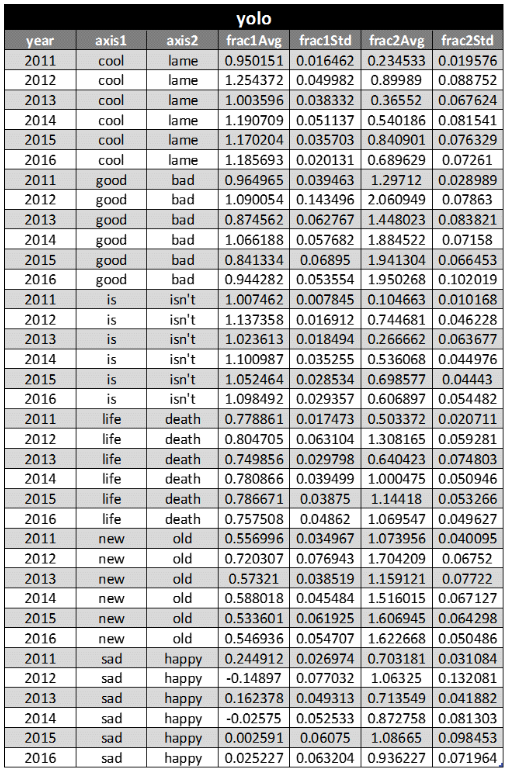
3.4 Creating the Axis Graphs

3.4.1 Word2Vec Output

Word2Vec then trained on these 60 input files using Deeplearning4j. For each file of those 60 files, we ran Word2Vec 15 times, and with each run it wrote a new CSV file of the 100-dimensional coordinates for each word it plotted into its vector space. We used the repetition of the run on the same file as a check to see how stable the words in the vector space remained when no other parameters were changed. In total this had a grand sum of 900 coordinate files, each of which contained around 3.1 million words.

3.4.2 Collecting the Vector Data

The next step was to create a table that would, for each neologism, for each year, for each axis, for each of the repeated 15 file runs, calculate the fractional distance of the neologism along the axis and the fractional distance of the neologism away from the axis. To do so, we wrote a program that collected the vector coordinates for the neologism and two axis words from the coordinate file, and from those calculated the intersection vector and thereby the two fraction values. Finally, since for each neologism for each axis for each year there were 15 runs, the program would take the average of these runs as well as the standard deviation, and the end result would be a table such as in Figure 9 for the word *yolo*, where "frac1" is the fraction along the axis, and "frac2" is the fraction away from the axis, and the n indicates that the neologism was not seen enough times in the training file for Word2Vec to write it in its vector space, and thus was not in the coordinate file.



yolo						
year	axis1	axis2	frac1Avg	frac1Std	frac2Avg	frac2Std
2011	cool	lame	0.950151	0.016462	0.234533	0.019576
2012	cool	lame	1.254372	0.049982	0.89989	0.088752
2013	cool	lame	1.003596	0.038332	0.36552	0.067624
2014	cool	lame	1.190709	0.051137	0.540186	0.081541
2015	cool	lame	1.170204	0.035703	0.840901	0.076329
2016	cool	lame	1.185693	0.020131	0.689629	0.07261
2011	good	bad	0.964965	0.039463	1.29712	0.028989
2012	good	bad	1.090054	0.143496	2.060949	0.07863
2013	good	bad	0.874562	0.062767	1.448023	0.083821
2014	good	bad	1.066188	0.057682	1.884522	0.07158
2015	good	bad	0.841334	0.06895	1.941304	0.066453
2016	good	bad	0.944282	0.053554	1.950268	0.102019
2011	is	isn't	1.007462	0.007845	0.104663	0.010168
2012	is	isn't	1.137358	0.016912	0.744681	0.046228
2013	is	isn't	1.023613	0.018494	0.266662	0.063677
2014	is	isn't	1.100987	0.035255	0.536068	0.044976
2015	is	isn't	1.052464	0.028534	0.698577	0.04443
2016	is	isn't	1.098492	0.029357	0.606897	0.054482
2011	life	death	0.778861	0.017473	0.503372	0.020711
2012	life	death	0.804705	0.063104	1.308165	0.059281
2013	life	death	0.749856	0.029798	0.640423	0.074803
2014	life	death	0.780866	0.039499	1.000475	0.050946
2015	life	death	0.786671	0.03875	1.14418	0.053266
2016	life	death	0.757508	0.04862	1.069547	0.049627
2011	new	old	0.556996	0.034967	1.073956	0.040095
2012	new	old	0.720307	0.076943	1.704209	0.06752
2013	new	old	0.57321	0.038519	1.159121	0.07722
2014	new	old	0.588018	0.045484	1.516015	0.067127
2015	new	old	0.533601	0.061925	1.606945	0.064298
2016	new	old	0.546936	0.054707	1.622668	0.050486
2011	sad	happy	0.244912	0.026974	0.703181	0.031084
2012	sad	happy	-0.14897	0.077032	1.06325	0.132081
2013	sad	happy	0.162378	0.049313	0.713549	0.041882
2014	sad	happy	-0.02575	0.052533	0.872758	0.081303
2015	sad	happy	0.002591	0.06075	1.08665	0.098453
2016	sad	happy	0.025227	0.063204	0.936227	0.071964

Figure 8: Axes Calculations

3.4.3 Making the Graphs

The final step was to plot the above results in a graph. For each neologism there would be six graphs, where each graph would show its word axis along the x-axis, and the neologism's location against the axis would be plotted as separate points for each year. The x-axis coordinate then represents the fractional distance along the word axis and the y-axis coordinate represents the distance away from the word axis. An example of *lightsaber* is shown below.

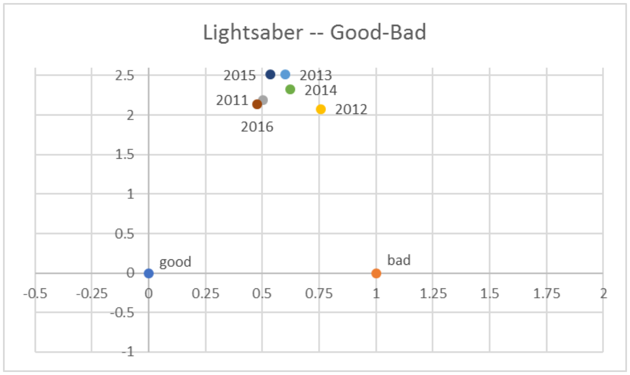


Figure 9: Lightsaber Plot

A typical distance of any randomly-selected word in the corpus away from the word axis would be about 1.5 or higher. In other words, it is just as likely that the word *water* or *purple* will be located at distance 1.5 or higher on the y-axis as the neologism.

4 Results

4.1 Comparing for Word Association

4.1.1 Sentimental Axes

Along the three sentimental axes, we see that the *good-bad* axis provides insignificant results, as the neologisms tend to be located about 1.5 or higher on the y-axis every year. However, the *sad-happy* and *cool-lame* axes provide more significant results. More surprisingly, we see a general trend for the neologisms to be more clustered around the negative words *sad* and *lame*.

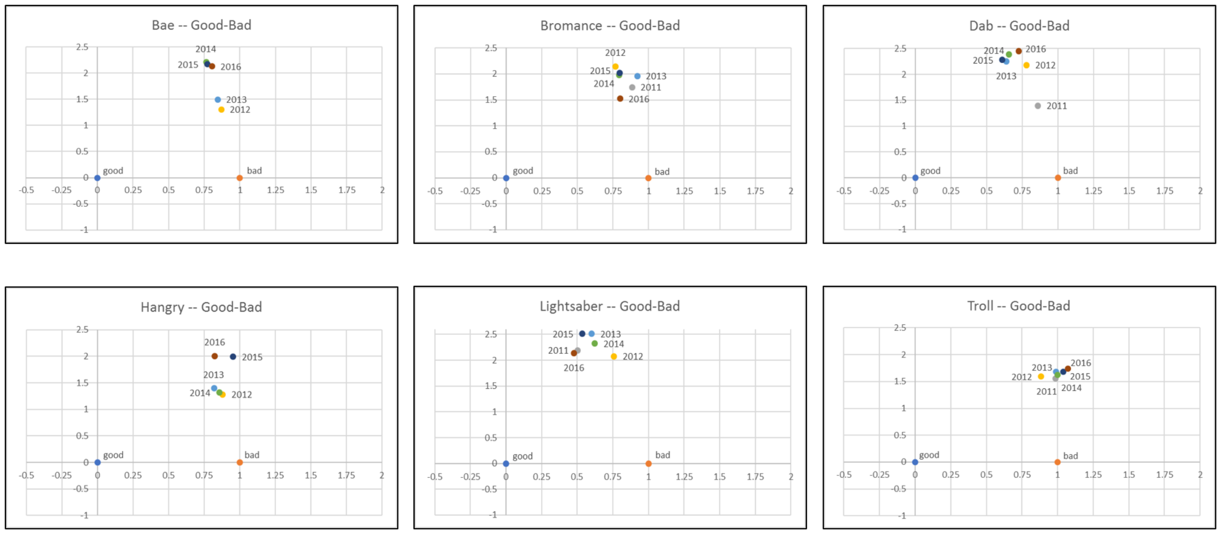


Figure 10: Good-Bad Axis

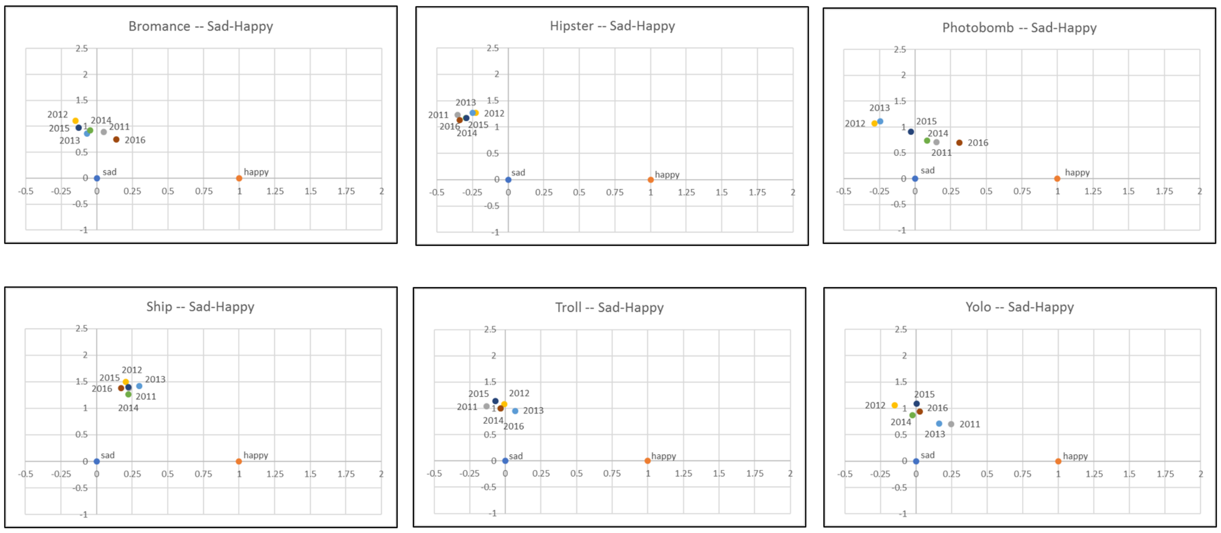


Figure 11: Sad-Happy Axis

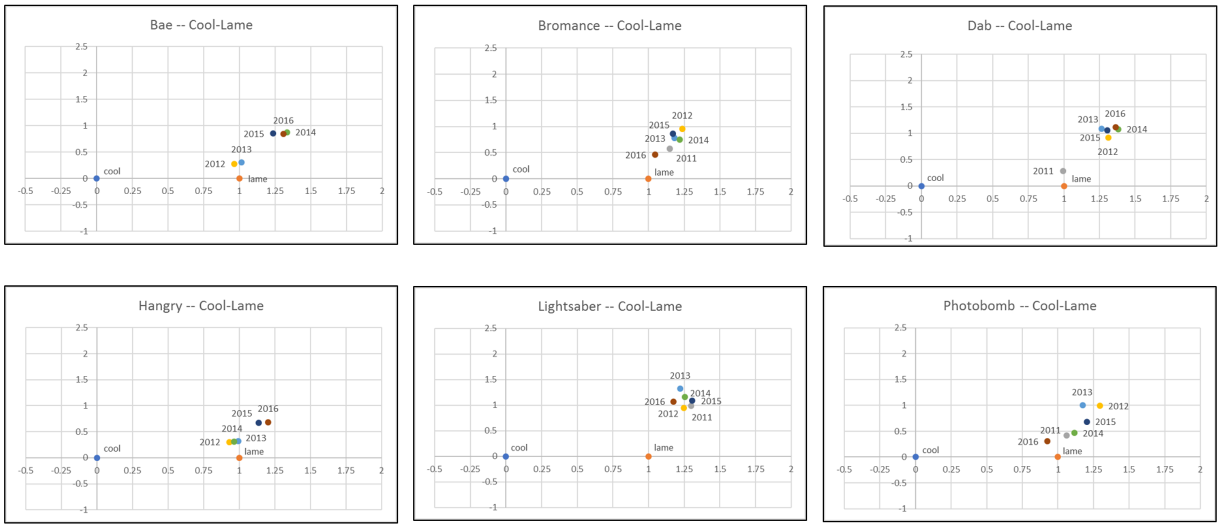


Figure 12: Cool-Lame Axis

There are many possibilities for why these neologisms might be sentimentally negative. It could be that people generally see these words in a negative light. It could be that any randomly-selected word is closer to these negative words, and the neologisms are no exception. It could be that Reddit is simply a negative website, and so all contexts will be negative. More specific inquiries to try these hypotheses would be needed to confirm why we see these results.

However, we might be able to contradict the statement that neologisms are negative when observing the results of the *is-isn't* axis. More significantly than any of the sentiment adjective axes, the neologisms strongly cluster around the word *isn't*. It is possible, perhaps, that while a neologism might be *sad* or *lame*, it may also *not* be. Again, this is a question we would need to explore more deeply to understand.

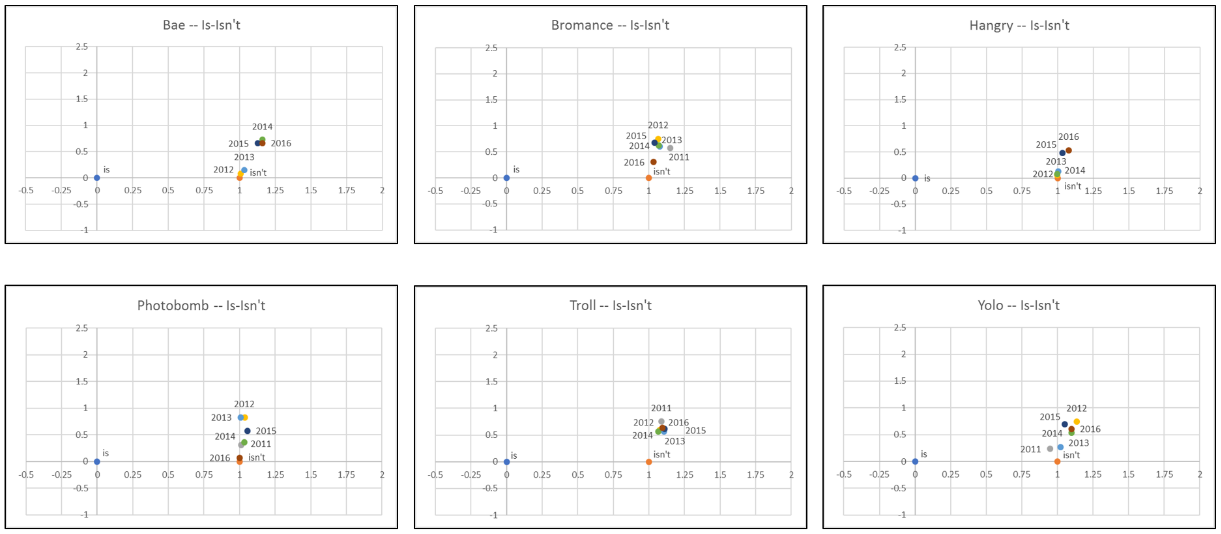


Figure 13: Is-Isn't Axis

4.1.2 Unique Axes

Many of the unique axes for each neologism provided spectacular findings about why using a word axis as a measurement of change is useful.

To begin with, for the three neologisms that have double meanings, *dab, ship*, and *troll*, we saw a much stronger association with axis words closer to their traditional definition rather than their neologism definition. For the most part, this is probably indicitave of how the data was collected, as although the neologisms are popular, they occur much less frequently than their older counterparts, and thus when typing the word into the search bar, it is more likely that the results returned will pertain to the old usage. At the same time, this might also tell us that neologisms are much less frequently *written* about than spoken of than their counterparts.

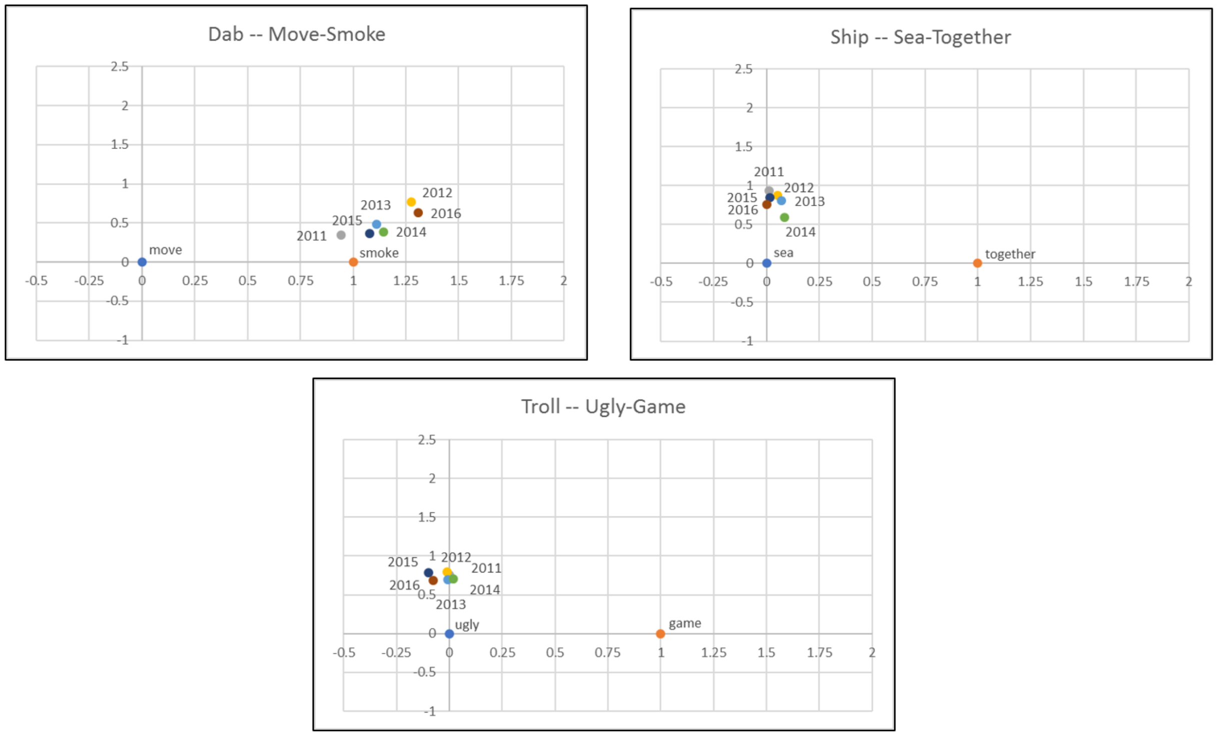


Figure 14: Neologisms with Double Meaning Axes

The unique axis for *hipster* is a perfect example of how the axis measurement can be used to analyze the semantic meaning of a word. A hipster is defined as “a person who follows the latest fashions, especially those that are outside the cultural mainstream”, and furthermore is “typically a young person”. When projected against the *young-old* axis, we see that *hipster* is much closer to the word *young* than the word *old*, and in fact is very far away from *old*.

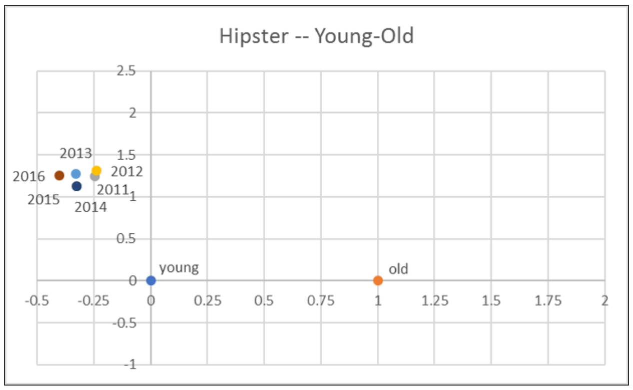


Figure 15: Hipster New-Old Axis

Finally, the biggest advantage of this measurement and using Word2Vec can be seen through the following three plots. In each, we get a glimpse of what type of contexts these neologisms are used in. For *hangry* we see a very strong cluster around *hungry* rather than *angry*—we might expect this because when we are hungry, this can make us angry, but when we are angry, it is less common that this is just because we are hungry. For the unique axis for *photobomb*, we see a trend of people's reactions to its use: it starts off as annoying, becomes funny, and then becomes annoying again. This is an amusing insight because it might suggest that when an activity becomes popular then it is funny, but with time and overusage it starts to simply become annoying. Lastly, the *yolo* axis is incredibly interesting because it shows a battle between context and definition. While *yolo* stands for "you only live once", so we might expect it to be close to the word *life* based on semantics, we instead see it is just as close to the word *death*. This is because *yolo* is often used as a reason to do something one might otherwise not do to try it before we die.

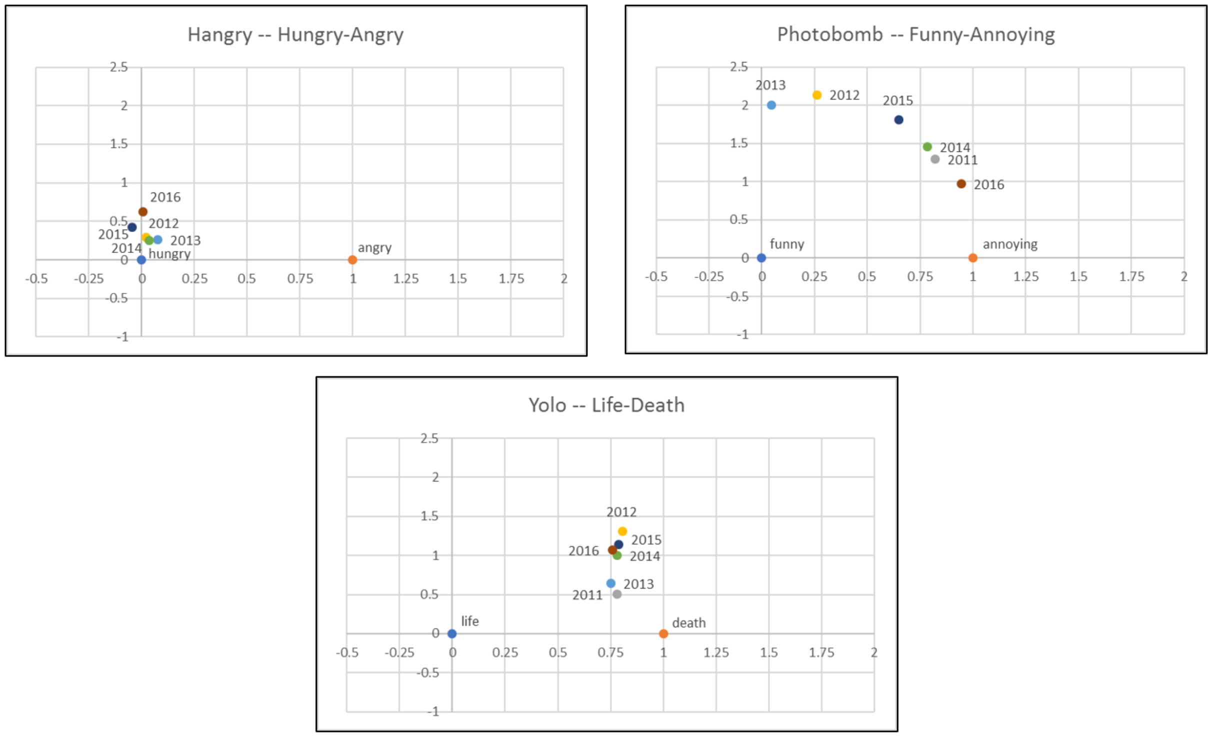


Figure 16: Axes Influenced by Context

4.2 Comparing for Year-to-Year Movement

Now setting aside which axis word the neologisms are closer to, we want to investigate how these neologisms move in space over time. To do so, we ran the same method described above to collect five common words, giving us data on the words *bird, friendship, green, run*, and *save*. We then compared how these words moved over time against the *good-bad* and *cool-lame* axis in comparison to the neologisms. A sample is shown in Figures 17 and 18.

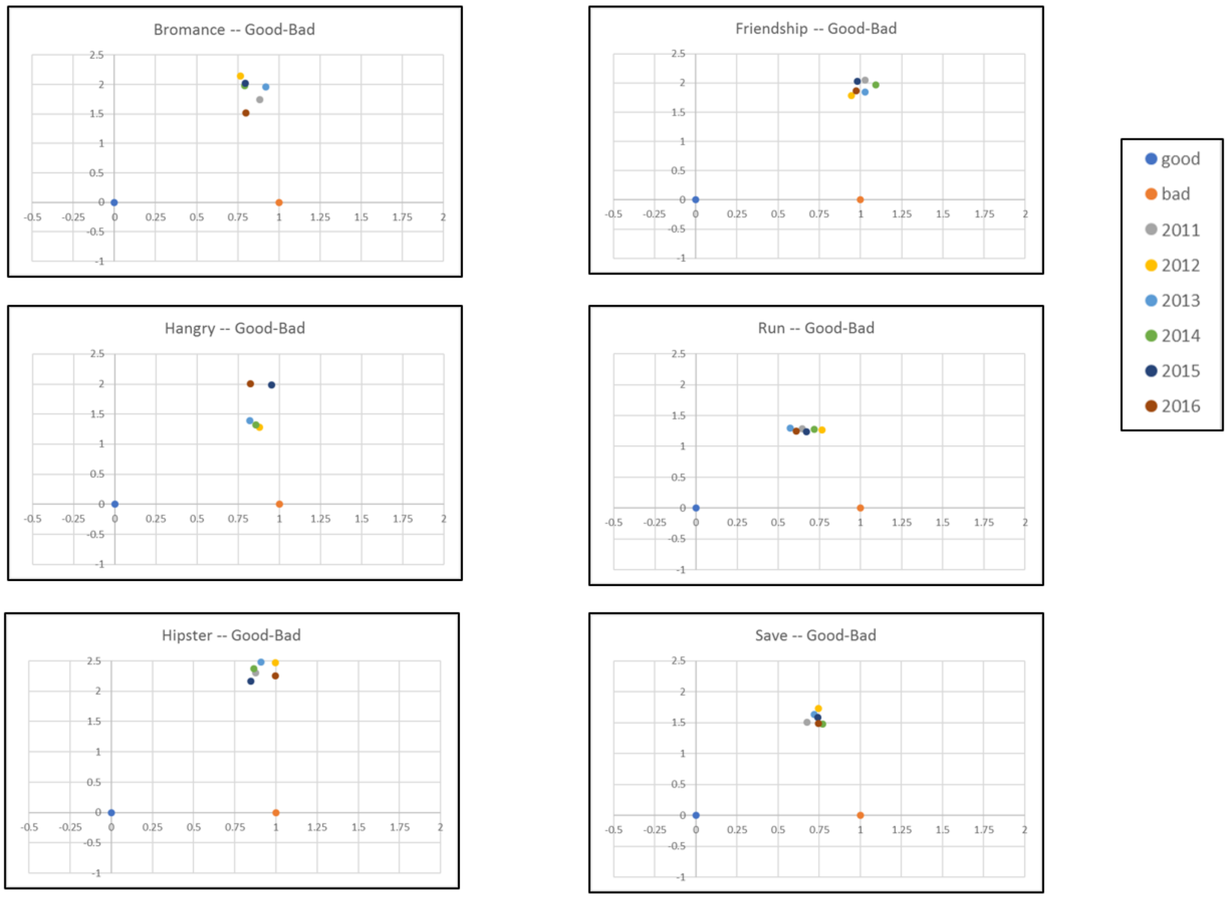


Figure 17: Good-Bad Axis

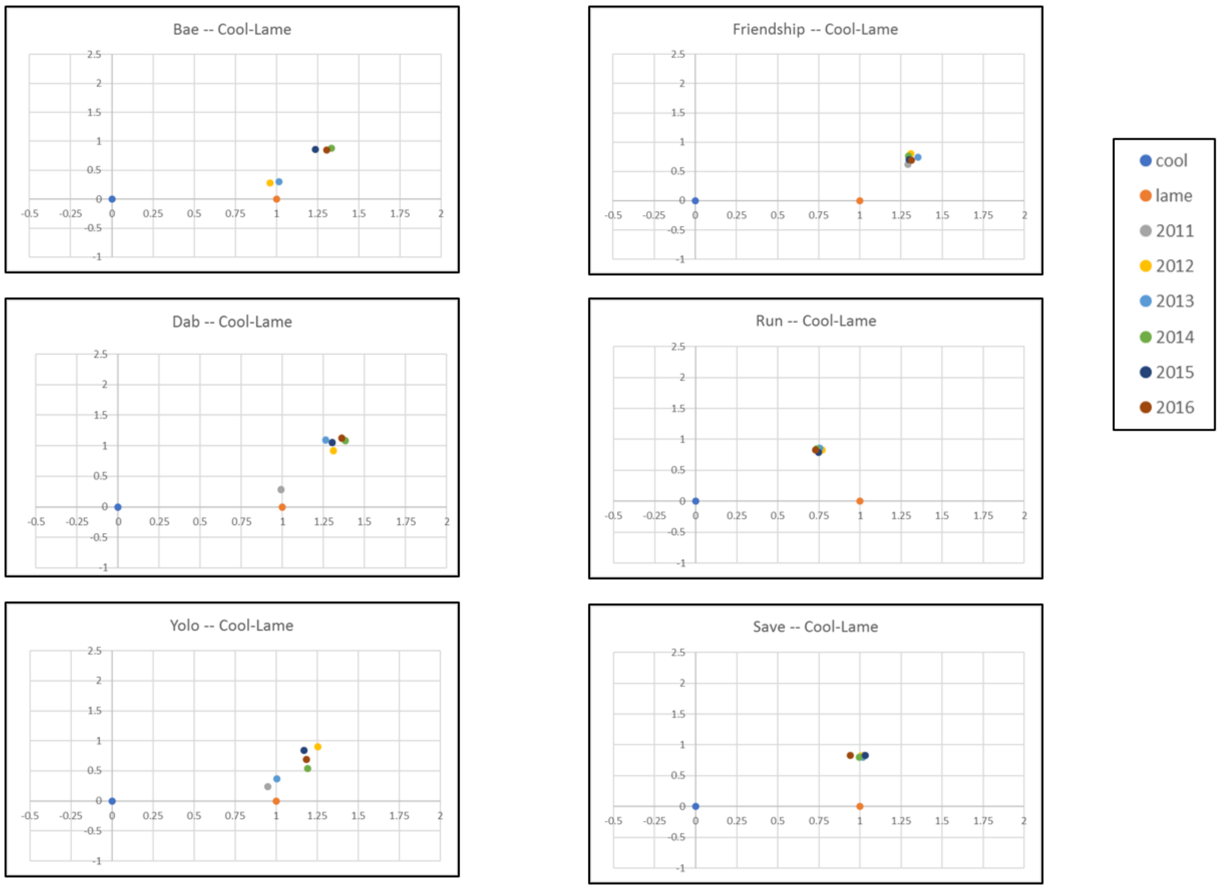


Figure 18: Cool-Lame Axis

As you can see, the common words are much less mobile in the vector space than the neologisms with respect to these axes. These results suggest that neologisms are much more flexible in their sentiment in their first few years of life than words that have been in language for a long time.

5 Conclusion and Future Work

In this paper we investigated how we could measure the change in semantics and sentiment of neologisms in their first few years of life. We treat them as vectors and then measure them against a sentiment or semantic word axis. We observed that this approach can give us some great insight into a neologism's evolution. On the sentiment side, we saw that with a good choice of an axis, we can gauge a neologism's sentiment on a positive-negative scale. On the semantic side, it seems we can also use the axis measurement as a way to help define a word, as we saw with the unique axis for *hipster*.

On the whole, this paper brings into light a new approach on measuring change in neologisms. An important understanding of our work so far is that we will find more insight into our measurement based on the axis we choose to measure the neologism against–for example, *good-bad* gave little insight into the sentiment of the neologisms, but *cool-lame* did. Thus, it would be very important in the future to compare our neologisms against a multitude of different axes to try and discern which axes more precisely define the sentiment or semantics of a word.

Perhaps more importantly than trying other axes would be to draw not only more training data to train Word2Vec on, but also new data from a corpus other than Reddit. Since Word2Vec trains better the more data it has, we can trust our results more the more data we have, and we can also trust that our data is not skewed to certain positive or negative contexts because we have drawn from more than one source. Moreover, some neologisms might have very different vector orientations if drawn from different sites: for example, *ship* is likely to be much more closely associated to its neologism definition than its old one if our data was drawn from a fanfiction website.

Then most obviously, there are hundreds of more neologisms we could study. While for this project we focused on neologisms that were very popular and widely used in our corpus, if we were able to draw from more corpuses then we might also be able to study neologisms that do not have to be as popular and frequently used. While these variables can help us further explore the utility of our axis measurement, there are of course many other ways to explore how a word "changes" over time using our Word2Vec model. Examples might include seeing how the closest words to the neologism in the vector space alter over time; if we could normalize how Word2Vec orients its vector space, we could see how the coordinates of a neologism move; or we could calculate a velocity of a neologism in the vector space.

The ultimate ambitions for this project are to answer some questions that might give us a deeper understanding of language evolution. To start, can we predict which neologisms will stay in the language? Can we predict new words that will be needed in the future? Is it true that most of the new words we are creating are negative, and what might that tell us about our perspective on the world now? Once we have some answers to these questions, we can begin to tackle the questions of *why* a word will change. Eventually, perhaps once we understand how this language evolution happens in small time frames, this will give us great insight into how language evolution occurs as a whole.

References

[1] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. In Proceedings of Workshop at ICLR, 2013.

[2] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. In Proceedings of NIPS, 2013.

[3] Chiru Costin-Gabriel, Traian Eugen Rebedea. Archaisms and neologisms identification in texts. September 2014. http://www.academia.edu/17228418/Archaisms_and_neologisms_identification_in_texts

[4] Piotr Paryzek. Comparison of selected methods for the retrieval of neologisms. *Investigationes Linguisticae* 16 (2008): 163-181.

[5] Daphné Kerremans, Susanne Stegmayr, Hans-Jörg Schmid. The NeoCrawler: Identifying and Retrieving Neologisms from the Internet and Monitoring Ongoing Change. 2012.

[6] TensorFlow. Vector Representations of Words. 19 June 2017. https://www.tensorflow.org/tutorials/word2vec. Retrieved on 11 August 2017.

[7] Google Code Archive. word2vec. https://code.google.com/archive/p/word2vec/. Retrieved on 11 August 2017.

[8] Google News Archive. https://news.google.com/newspapers. Retrieved on 8 August 2017.

[9] Large Movie Review Dataset. http://ai.stanford.edu/ amaas/data/sentiment/. Retrieved on 8 August 2017.