Determining Semantic Similarity through Source-Driven Context

Mckervin Ceme
A.B. Computer Science, 2016
mceme@princeton.edu

Ananda Gunawardena
Faculty Advisor
guna@cs.princeton.edu

Abstract

The problem of determining whether or not two sentences share the same meaning – semantic equivalence – is a challenging endeavor in linguistic processing. Semantic equivalence has useful applications in natural language processing, including cross-lingual applications as well as paraphrasing large texts. My approach to the problem of semantic equivalence uses the context of where a phrase was written to help uncover its meaning. In particular, I showed that given a corpus that produces the point in a document in which a comment was made, that information could be used in improving the reliability of WordNet approaches to semantic equivalence. My experiments showed that using an outside context yields a measurable improvement in determining sentence equivalence.

1. Introduction

The problem of ascertaining semantic equivalence between phrases in natural language processing (NLP) has been researched in various methods. Like most natural language processing problems, the complexity and variance of the meaning of a word varies largely, depending upon the context. A large amount of research methods to measure semantic similarity have involved using things such as bag-of-words models and/or using a large database such as WordNet (Wallace, 2007) [8]. However, context as it applies to semantics is limited because words and small phrases have multiple meanings, depending upon both where it is in a phrase and when the phrase was written, in addition to a word's synonyms. Suppose that the following two sentences are comments on a user's video post on a video-hosting website:

What a lucky dog!

I can't believe that dog did that!

In these sentences, the subject *dog* has multiple semantic meanings, ranging from being a member of the genus Canis to a "despicable man or youth," and even a human who has amassed good fortune on account of circumstance. There is an empirical "distance" between the different concepts (meanings), or words. This property is called *semantic distance*. Therefore, *semantic equivalence* can be defined when two lexical units (texts, sentences, words) have relatively small semantic distances between them. Many approaches to tackling the challenge of semantic equivalence have used a *corpus* of words and phrases, which contain data about words, their synonyms, and their relation to the other meanings that a specific term may have.

Corpus-based approaches to the problem of semantic equivalence provide useful data that can be combined with metadata from an outside source to increase the reliability of current semantic similarity algorithms. These algorithms generate statistical likelihoods that two terms are semantically similar. Furthermore, iterative statistical models like Latent Dirichlet Allocation (a probabilistic topic model, LDA for short) have shown to be reliable in uncovering hidden topics as well as having importance in machine learning algorithms. LDA in particular can uncover hidden topics in a document. Combined with the data generated from a corpus-based algorithm, topic modeling provides data that is useful in finding the context behind a given word or phrase. Section 2.4 explains how LDA works in much more detail.

The context of a phrase plays an important role in determining the semantic distance between two sentences. More specifically, if the context from the source in which phrases are located in can be extracted, it can be analyzed separately from the words themselves, and thus telling more about whether two phrases are semantically similar. In essence, the context, in combination with the data obtained from a topic model and the words themselves, can be combined to give more reliable results in determining semantic equivalence.

In this paper, I aim to show the statistical usefulness of context in determining semantic equivalence. Specifically, I explain:

- The limits of an LDA & lexical database solutions to the semantic equivalence problem.
- That the combination of topic modeling, lexical databases, and the context from the source of the phrases leads to a higher likelihood of semantic equivalence than the three working independently (Section 4.1).
- How the metadata from the source provides the context (in terms of location) of a phrase or sentence, which ultimately gives valuable information in inferring the meaning of said phrase (Section 2.2).

2. Problem

2.1. Vector-based approach

In similarity with many other natural language processing problems, one of the largest impediments to semantic equivalence is the inherent lack of information we have in determining similarity. Specifically, trying to uncover the context in which a term is used has proven to be a difficult challenge. As a result, large corpora have been created in order to enumerate the relationship between different terms (for example, how orange and banana are both subsets of the general category of fruit). Unfortunately, given the topical information that can be gleaned from large corpora, it is still difficult to understand the context of a phrase without further information, such as when & where the phrase was said in addition to who the phrase was written by. These questions shed light on the nature of a term.

But if we can know the context in which a phrase was said in, then uncovering the meaning of a phrase, and thereby determining sentence equivalence, becomes more reliable. This is true, even if one can only gleam bits of information. Consider the following example of two phrases:

What a bad apple.

He is not a virtuous man.

Many semantic similarity services remove structural words that make up a phrase, so terms such as *a*, *is*, and *he* are removed from the sentence. So, a more accurate depiction of what a semantic equivalence algorithm that does term-by-term (or vector) analysis, as seen in Figure 1.

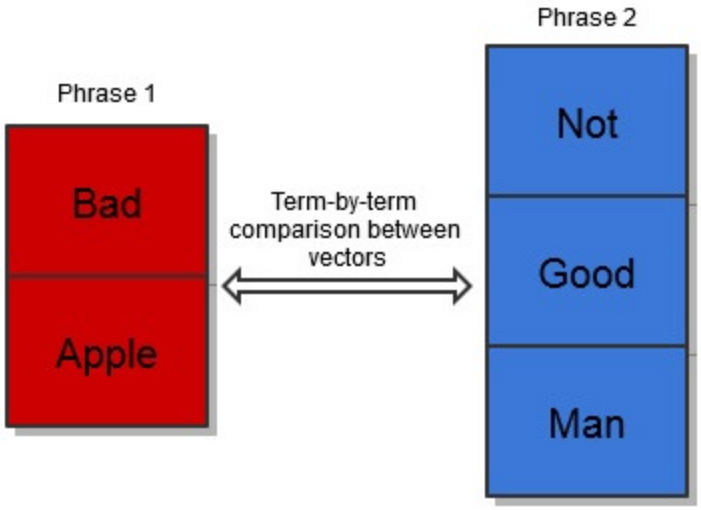

Figure 1: Vector-based approach when comparing phrases.

One implementation of a vector-based model is the UMBC Phrase Similarity Service, created by researchers (Han et. all, 2013) [3]. This service ranges words from 0.0 to 1.0, with a 1.0 being a perfect match. Using the two vectors found in Figure 1, an example output of this analysis can be seen in Table 1. This approach relies heavily on the specificity of the lexicon. Which is why WordNets have so many different synsets. Still, this extremely granular lexical database has seen much criticism, as outlined in (Jing and Tzoukermann, 2001) [4].

	Bad	**Apple**	**Not**	**Virtuous**	**Man**
Bad		0.03860	0.01605	0.14579	0.2066
Apple	0.03860		0.0	0.01595	0.0
Not	0.01605	0.0		0.01114	0.26640
Virtuous	0.14576	0.01595	0.01114		0.05820
Man	0.02066	0.0	0.26640	0.05820	

Table 1: Vector results

Given the nature that a *bad apple* is a colloquialism for an individual who is corrupt/ morally dubious, the similarity scores should have been higher. Based on these two phrases, humans can infer that these two phrases are much more closely related than the data shows. If the algorithm could have known, for example, that these two phrases appeared on a comment on a video, even a video about a specific individual, then that contextual information can provide the information to ascertain not necessarily what a single phrase means, but that two distinct phrases can have similar

meanings. In essence, the problem of uncovering the similarity between two sentences relies on not just whether the terms between the two phrases are related, but also the context in which the phrases exist.

Knowing this, I experimented with using a source of information that I could pull contextual information from in order to improve upon the statistical reliability of other methods. Specifically, I worked with the education platform Classroom Salon, which is a context-driven education platform. When an instructor uploads a video lecture or reading, their students can annotate the shared document with any comments they may have. In large scale "salons," as they're called, there can be thousands of comments on a given text – which can be difficult for a single professor to sift through all of the students' questions and answer everything, especially when many of the questions may in fact be redundant. The utility of semantic equivalence in such a platform is to uncover similar questions or concerns posed by students – which reduces the amount of work for professors with large classes, thereby allowing them to engage more students at any given time.

2.2. Classroom Salon

In order to further improve upon the vector-based and corpus-based approaches to semantic equivalence, I took advantage of the contextual data that Classroom Salon (Barr and Gunawardena, 2012) [1] provides. The following figure shows a typical "salon" page from the Princeton University Computer Science 226 course in the fall of 2014. Notice that the user comments are not just displayed in a scroll-able list, but each annotation has a time stamp affixed to it that shows when in the video the student made a comment. This contextual data, having a time-stamp annotation, is very important information for the instructor, and ultimately provides context for the comments in a given instructional video

The time stamps seen in Figure 2 provide contextually-important information in determining when two sentences are semantically equivalent. The time difference between two comments provides data that shows the likelihood that an event in the video, that generic point in the document, may have triggered both of those comments to be made. The closer in time two annotations are, the

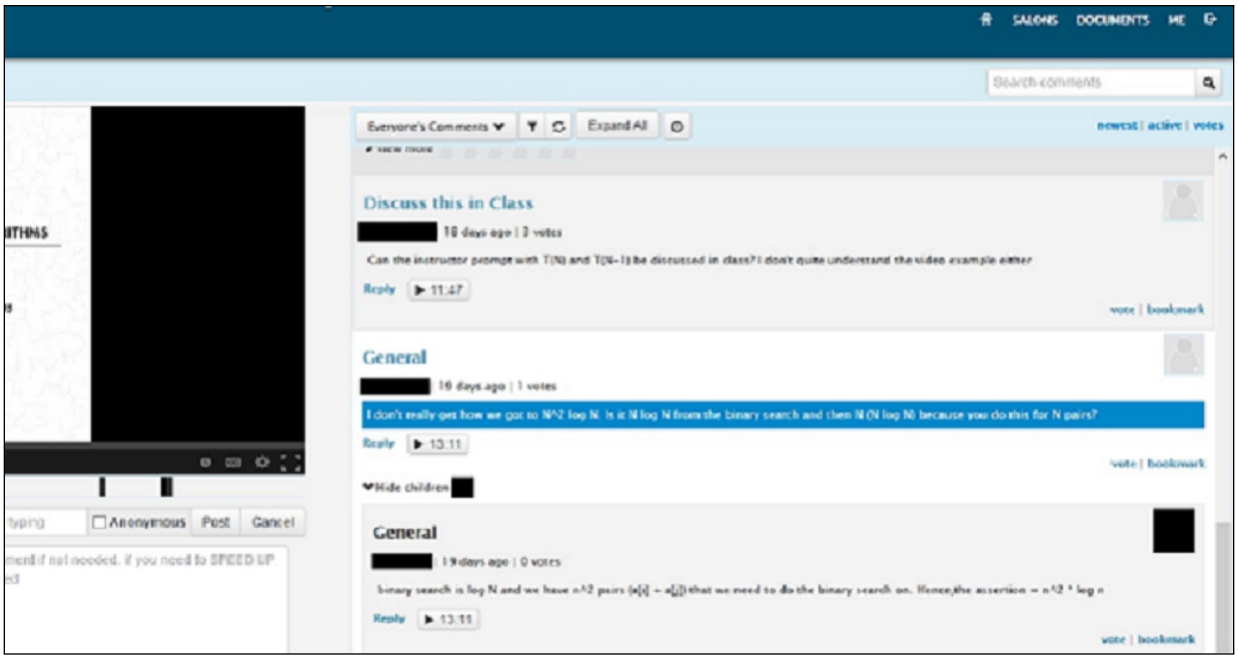

Figure 2: A typical "salon" with user comments alongside the instructor's video.

more likely that the comments are concerned with the same concept in a video. Essentially, rather than trying to solve what each comment means individually and comparing those semantics, I experimented with context (specifically time) to see if the data that Classroom Salon generated could uncover the likelihood of redundancies in the comments section of its classroom.

I made this assertion by assuming that at the moment a student felt confused (or just wanted to make a general comment), they would then write their comment. For instructor-uploaded video content, I assumed that the student would pause the video, write their comment, and then resume. If a teacher, on the other hand, uploads a document as opposed to a video, then the user can highlight the section in the text that is specifically confusing, and write their comment. The context in that case would be the location of the annotation location marker in the text.

2.3. Context-driven solution

Nevertheless, it is important to note that the contextual data driven from timestamps is not enough on its own to determine semantic equivalence. Having contextual data serves as a boost to aid in the quality of the corpus-based and vector-based approaches to semantic equivalence. As a result, my approach to the problem involved three main components.

- Using probabilistic topic modeling, and in particular, Latent Dirichlet Allocation (LDA), to uncover the common topics across all the comments in a given salon. This sorts the comments into sections based on an estimated topic, to increase the likelihood of duplicates.
- Using a semantic equivalence service that does term-by-term analysis take advantage of the information that words and their multiple 'senses' offer to increase the likelihood of equivalence.
- Finally, utilizing the power of time-based context in improving the probability of two phrases being equivalent.

Each component of my approach provided data that, while independently may provide marginal results, together the three steps provided more information for determining semantic equivalence. By putting more emphasis on the*context*, I minimize some of the weaknesses of a vector analysis. The strengths of source-driven context can help offset the weaknesses of term-by-term analysis. This is a more holistic approach that emphasizes the third bullet, and to a lesser extent the first bullet. Source-driven contextual evidence provides data that boosts the overall likelihood of two sentences being semantically similar.

2.4. LDA and corpus-based approaches

Latent Dirichlet Allocation is a probabilistic topic model that uses machine learning to uncover the hidden topics over a given data set. First, it randomly assigns words in a dataset to n user-inputted topics. Then it calculates two probabilities. Equation 1 shows the proportion of words in d that are currently assigned to t. Then, Equation 2 represents the proportion of assignments to t that came from w in every document in your dataset.

$$P(topic\ \mathrm{t} \mid document\ \mathrm{d}) \tag{1}$$

$$P(word\ \mathrm{w} \mid topic\ \mathrm{t}) \tag{2}$$

It multiplies the two probabilities, assigns a word to a new topic, and repeats this process for all the words over many iterations until the approximation is reliable. This is useful in semantic equivalence

because of the fact that a topic model produces hidden topics that are common among distinct phrases. The hidden topics that LDA produces also serve as a type of *context* that narrows down the probability that two phrases are not semantically similar; the semantic distance between two phrases would decrease. With this idea in mind, I used the Machine Learning for Language Toolkit (MALLET) (McCallum, 2002)[5] (Section 3.1)from the University of Massachusetts to uncover the hidden topics from the Classroom Salon dataset. Topics generated by a topic model serve as a way to categorize annotations by hidden topic. While this does not prove that two annotations are semantically equivalent, this does group annotations that may be semantically similar before ever actually looking at the meaning of the terms themselves. Results from an example topic model can be seen in Section 3.1.

In conjunction with the topic models, next I postulated that performing a vector-based similarity algorithm on a topically-organized data set of annotations would unpack more information about the equality of two sentences. The benefits of creating a topic model prior to a vector analysis are as follows:

- The amount of comparisons one has to perform is reduced. It would be wasteful to compare two different annotations whose topics vary wildly.
- The topics serve as a type of context for the vector analysis. While it does not have that contextual information as it is looking up terms on WordNet, from the end-user's perspective working on some form of contextually-based dataset would yield more reliable results.
- Since the topic model is trained based on word probability, the likelihood that similar words will appear in two phrases increases, thereby making the term-by-term analysis more accurate. ...

In order to perform a semantic text similarity analysis on the annotations in their respective topics, I used a web API created by researches from the University of Maryland, Baltimore County (Han et. all, 2013)[3]. The results from the semantic similarity outputs are highlighted in Section 3.3. The time-based context serves as a critical element in calculating a total score (percentage) of the likelihood of sentence equivalence. If two annotations are close in time (that is, if two users commented near the same point in instructional video), then in the case of Classroom Salon, that

data can be taken into account in ascertaining semantic equivalence. The time stamp difference is a unique contextual artifact to Classroom Salon. Because it is *unique* to every annotation, then similar to the semantic distance would be the property of time distance, which is simply the difference in time between two time points. A low time difference is a very strong indicator of semantic similarity. Generally speaking, instructional videos are organized to go ever an ordered topics over a time period of a lecture. As a result, comments on these videos are more likely to be *a.)* questions, or answers to questions about the source material; *b.)* centered on distinctive moments in a video; and *c.)* are likely to use distinct words to describe their questions. The contextual data one can gleam here is important, and the most readily available contextual data to do math on is the time stamp.

2.5. Semantic Equivalence Calculator

Ultimately, given all of the data amassed, a final percentage needed to be calculated for each pair of sentences in a given topic across all topics. By looking at the following parameters (each with a maximum score of 1):

- The fact that each annotation had a major topic (score of 1).
- In one case, the average of the percent by which each annotation was a member of said topic (0-1).
    1. In the other case, the multiplication of those two averages replaced the major topic score as the probability both phrases were a part of the same topic. In this case, at the end the final result was divided by three as opposed to four.
- The time difference scale (a score of 1 meaning no difference, and 0 meaning as far away as possible).
- Finally, the similarity score from the web service (0-1).

The averaged data will yield a probability of whether or not two phrases are likely to have the same semantic meaning.

3. Multi-tier Data Engine

3.1. MAchine Learning for LanguagE Toolkit (MALLET)

Creating the topics required fine-tuning the parameters for running MALLET. The parameters that were user-dependent included the number of topics n that needed to be produced and the optimization parameter x. The optimization parameters allows the algorithm to weight certain topics more than others based on the frequency of certain words appearing in a body of text. So each run of the topic model would produce a list like the output below in Figure 3 (note that this was produced with $n = 15$ and $x = 10$).

```
0       0.15126         sort is the sorting for so of in it merge quicksort better insertion than quick
way msd stable does
1       0.34500         we to can do the use how need if why a when should in that not you have
be
2       0.18786         the of array is to a number in when size what by that are sorted i an elements
and
3       0.28420         the of to interesting in it see applications very is s and really algorithm
application that can problems real
4       0.12109         tree the a of search binary is trees structure in depth b to implementation bst
nodes for red black
…       …               …
```

Figure 3: Sample run of MALLET (Java-based implementation)

The first column shows the different topics, the second shows the weight calculated from the optimization parameter, and the third shows all the words that made up the topic. Topics with higher weights are based on more common structural words (i.e. articles, conjunctions, and prepositions). Given the complexity of semantic equivalence and the nature of this dataset (of a computer science course covering many topics such as Union Find, Minimum Spanning Trees, Hash Tables, and the like), having too small an n not only ignores the variety in user comments, but it produces topics more likely to be based on structural words than those of higher n. Take for example the hidden topics from Figure 4, a run with $n = 40$ and $x = 10$. In this subset of the output, the weights were lower than those in Figure 3. Although the optimization parameter weighted certain topics more than others, with a larger number of topics, the weights were distributed more. And so, there were

more *unique* topics than in the previous figure, hence why the weights appear to be smaller in Figure 4 than in Figure 3.

1 0.07476 path the shortest bfs edge dfs to vertex edges paths from augmenting finding find vertices all graph source problem
2 0.10345 the of tree node a right nodes to left depth root each size from j because instead parent child
… … …
9 0.06297 to a how with bit code forward little the deal test you bits s files compress by compression looking
10 0.05058 union find connected to quick id component components set weighted p change graph has command objects too nodes two

Figure 4: More sample data from MALLET

The graph below in Figure 5 shows the proportion of structural-component based topics compared to more unique topics. Even as *n* increased, the relative number of structural topics remained flat - meaning although the higher *n* created those topics, it also created others that were very distinctive.

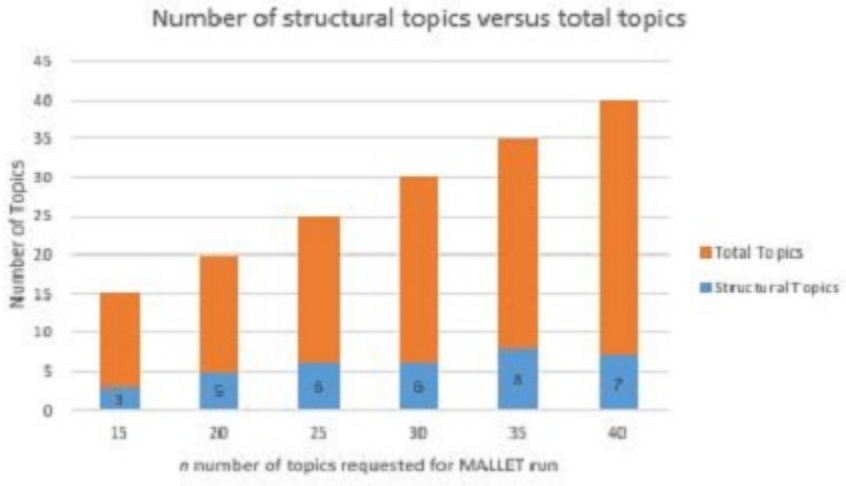

Figure 5: The blue aspects show the total number of topics that resulted from structural artifacts such as 'why,' 'it,' 'a,' etc. The orange is the *n* that was inputted by the user.

However, I still performed experiments on all of these *n* in order to generate a varied data set. With more unique topics, there was a higher chance of specificity. However, I could not make *n* too large in relation to the size of the annotation list, or else each annotation can get treated like its own topic, thereby ruining the advantage gained from LDA.

3.2. Generated topics

The LDA analysis also produced a composition document, with each annotation having its topics sorted by highest probability, like below. Here, each comment has its own ID, its topic (beginning with its major topic), and the topic percentage, followed by subsequent percentages for the other topics. An example of the composition document can be found in Figure 6.

```
...    25253  0    0.28982880797821003    10    0.18073137838869818 ...
...    24811  3    0.16697276383801235    17    0.16604797027798154 ...
...    25050  19   0.45455214486609324    11    0.10920064057877539 ...
```

Figure 6: Sample output of the topics and percentages that each unique annotation is in each of the *n* topics

The major percentages (the highest percentage for each annotation) provided the data needed to calculate the probability that two phrases belonged to the same topic. To compute this value, I needed to multiply the probability of annotation *a* with the probability of annotation *b* (both *a* and *b* must be within the same topic). In a separate run, I also averaged the two percentages in order to average the topical similarity of the two phrases. See Section 4.1 for those results. Another added benefit to a topic model is that is helps limit the amount of work needed to be done while comparing sentences. The dataset that I used contained approximately 5,000 different topics across almost 100 different videos. Rather than having to compare every single topic to one another, only the comments within each topic got compared. The performance boost also helps to make a multi-tiered approach like mine easier to apply in other real-world applications.

3.3. Web-based similarity calculation using WordNet

Given the sorted annotations by hidden topics, the next step involved performing semantic equivalence tests, term-by-term, for every phrase in one topic to every other phrase in the same topic. To accomplish this, I used the web API provided by UMBC researchers that allows for two sentences to be inputted into this distinctive API to produce a decimal ranging from 0-1, with a 1.0 representing perfect similarity (i.e. two identical sentences semantically and structurally). This similarity model uses LSA Word Similarity (Han et. all, 2013)[3] – that words that appear in the same contexts over

experiments are likely to have similar meanings – in conjunction with WordNet – to determine the similarity between two words, as outlined in Section 2.1.

ID 1	ID 2	Web Score
25350	25395	0.497792
25298	25395	0.480571
25396	26055	0.231923
25396	25610	0.00000
26055	25610	0.00000
...	...	...

Table 2: Web results

In Table 2 below, many of the outputs from the sentence equivalence web API test are below 50%. Interestingly, some of the tests did not provide any output at all (a score of 0.0). This is primarily due to the limitations of lexical-based approaches of corpora like WordNet. These vector-based approaches ignore many structural words. Also, some corpora simply do not having enough data inside of them to account for the vast English lexicon. There are 155,287 (Wallace, 2007) [8] unique words in WordNet, and even ignoring pronouns and other words that build sentences, according to a joint experiment performed by researchers from Google and Harvard, there are over 1,000,000 unique words in the English language (Michel et. all, 2011). [6]

This is not to say, however, that phrases that return a 0.0 on the web service all lack any potential for them to be equivalent. In fact, without the prior LDA topic modeling and the use of context that I outline in the next section, then these phrases would have to lack similarity, if the STS Service were the only thing used. Still, those phrases that did provide similarity scores, even low ones, do give insight as to whether or not two phrases are semantically similar.

3.4. Context Calculation

As mentioned before, the contextual information that Classroom Salon provides was one of the driving forces in boosting the reliability of WordNet-based approaches to this problem. The equation used to calculate the context can be viewed in Equation 3.

$$C = 1.0 - \frac{|T_1 - T_2|}{L} \tag{3}$$

L represents the time (in seconds) of the very last time stamp in a data set. T_1 and T_2 are the times of the two annotations to be measured. The resukt from the calculation ultimately returns the final context, C. By dividing the absolute value of the time difference by L, I found the percentage difference between the time contexts of two annotations. Subtracting one from that fraction results in the percentage of how close two annotations are. Suppose that $L = 900$ seconds (10 minutes), and one had to compare comments at the 2:15 second mark and the 4:47 mark, respectively - 135 and 287 seconds, respectively. Solving the equation yields an 83% closeness rating. A subset of time similarity scores can be seen in Table 3 below.

Annotation 1	**Annotation 2**	**Similarity (%)**
26055	25610	98.99
26189	25496	60.61
26189	25469	40.47
25496	25469	86.87
25553	28758	70.10
...	...	...

Table 3: Context results

The context (i.e. time difference) also makes up for some of the inconsistencies in the LDA topic model. Since LDA is *probabilistic* by nature, some comments that are far away from each other (low time similarity percentage) can end up being grouped into the same major topic. If the assumption holds that a student will propose a question about a confusing topic relatively close to the time in which said topic was discussed, then the contextual data from the time stamps, specifically similarities with low percentages, show that comments that are far away are less likely to be semantically equivalent.

4. Results

4.1. Final score comparison

The final similarity score was calculated experimentally under two different assumptions. In the first case of comparing two phrases, I averaged the percentages in their shared major topic, and in the second case, I simply multiplied the two percentages. Moreover, the results varied when the number of n topics was fine-tuned in the LDA analysis. Therefore, the following tables show the results from three different values of n: 15 (Table 4), 20 (Table 5), and 40 (Table 6).The optimization parameter was held constant at $x = 10$ for all runs. Each run was performed on a different video – I ran tests on 95 different videos, each with a large amount (over 50) comments.

Annotation 1	Annotation 2	LDA 1	LDA 2	Web Service	Time Stamp	Result
28160	28240	0.5167	0.3882	0.5625	0.9957	0.7527
27706	28001	0.3997	0.2281	0.2373	0.7913	0.5856
27953	28092	0.5007	0.2363	0.5482	1.0000	0.7292
28060	28137	0.3700	0.5010	0.2130	0.4140	0.5156
28100	27924	0.4573	0.3381	0.5974	0.6022	0.6493
...	...	...	...	...	...	...

Table 4: *n = 15* (Video 1)

Annotation 1	Annotation 2	LDA 1	LDA 2	Web Service	Time Stamp	Result
25025	25255	0.5972	0.6565	0.8598	0.9450	0.8579
27112	27113	0.6553	0.3796	0.6481	1.0000	0.7914
25064	25157	0.4117	0.6103	0.2690	0.9710	0.6772
24856	25143	0.4642	0.4403	0.4627	0.9740	0.7223
25145	25255	0.3601	0.6565	0.2916	0.9840	0.6959
...	...	...	...	...	...	...

Table 5: *n = 20* (Video 2)

The final column, Result, is a measure of the *average probability that these three methods produced*. That averaged value states the mean likelihood that two phrases are duplicates, after having used all methods in sequence. In order to calculate the final outcome, Equation 4 was used.

Annotation 1	Annotation 2	LDA 1	LDA 2	Web Service	Time Stamp	Result
26150	25700	0.3783	0.3135	0.7927	0.9034	0.7606
25574	26042	0.4662	0.2337	0.0000	0.9840	0.5440
26181	25700	0.4311	0.3135	0.5819	0.9153	0.7174
25660	25574	0.2926	0.4662	0.5152	0.7551	0.6625
25856	25540	0.1800	0.1574	0.3865	0.9657	0.6302
...	...	...	...	...	...	...

Table 6: *n = 40* (Video 3)

$$Result = \frac{1.0 + \frac{(A_1 + A_2)}{2} + w + c}{4.0} \tag{4}$$

The 1.0 served as an indicator that we were comparing by topics (for the multiplicative section, Equation 5 was used). A_1 and A_2 are the two averages from the LDA, w is the web service result, and c is the context score. Rather than act as an outlier, the time stamp (or contextual data) made the resulting percentages fit better with the actual annotations. To illustrate this point, consider Row 2 from Table 6.

> *25574: Why is it bad to remove in iterable?*
>
> *26042: Is it possible shed some more light on remove() function in the array implementation of stack iterator*

Both of these comments were explicitly asking to explain some of the nuances of a particular function in an iterable. In these two phrases, the web service was unable to register a similarity score. In addition, the LDA probabilities varied considerably - one was double the other. Therefore, the time stamp context was what gave these two phrases a 54% chance the two were related. Neither the web-based WordNet nor the LDA gave these phrases relatively a high probability of semantic equivalence on their own.

In these experiments, I pulled some of the similarity scores from multiple (three) videos. I was able to extract a few duplicates amongst these runs from Tables 4, 5 and 6. I confirmed this using human intuition (doing manual comparisons from the sample pool). More importantly, some of the confirmed duplicates had low similarity scores from the web service. However, the context (in addition to the LDA percentage) can boost similarity by many percentage points. An example of a

run in Table 5 is shown below, which highlights the limits of a term-by-term comparison method and what the context helped bring to the similarity score.

25064: Very useful!

25157: This demo is really helpful.

Even though the two sentences are very similar structurally, the service returned a very low score. Part of the reason why the WordNet-based web service returned a low score was because it did not account for adjectives and adverbs. The context score, however, was much higher. The contextual score in addition to the LDA assignments showed these two sentences were in fact as semantically similar. In short, the context greatly helped in relating the similarity between these two phrases. Although structural words are too common for a WordNet to be able to decipher meaning, structural words actually provide a type of *phrase-driven* context that may be useful for other types of approaches to this problem.

While the results did show some duplicates, there were also a few negatives that did not have much semantic similarity. For this case, consider this example from Table 4.

28060: Does the shuffling speed up the algorithm enough to make it worth the extra time spent shuffling?Is there an array size for which shuffling is appropriate, below which we should keep the array order?

28137: Why does the partition element in the middle cost less number of compares?

The similarity score from the web service for these two phrases is not negligible. However, the comments were so far apart that the resulting final similarity decreased. And, these two sentences were not in fact semantically equivalent, even though they shared similar words. The first sentence is asking about why there is a speedup in runtime when shuffling for small arrays, while the second is actually a more general question about the nature of a sort that uses a partition. The very low context score helped to drive this score (still, the overall score was above 50%)

In certain cases, some of the duplicates were associated with topics that had larger weights – topics that were developed from more structural words from the LDA analysis in Section 3.1. Although the structural words do not give much information about the meaning, they indicated that many

comments/questions were worded similarly. Further *phrase-driven research* needs to be done to analyze, or somehow understand, what a sentence is trying to convey (its meaning) through its building blocks.

4.2. Multiplicative Results

Table 7 shows the same results as Tables 4, 5, and 6. These final results are all measurably lower than the results from Section 4.1. To calculate this, I used a modified version of Equation 4, shown below in Equation 5. In these experiments, I multiplied the average to become the likelihood that both phrases shared the same topic.

$$Result = \frac{(A_1 \times A_2) + w + c}{3.0} \tag{5}$$

This was done in order to decrease the weight that LDA had in computing the final score. Multiplying probabilities of course decreases the likelihood that both event A_1 and A_2 will occur. This in turn allowed for a more narrowed focus on the effect that the context itself had on the net solution. Essentially, this was done to see how much context the time stamps themselves provided.

In the data from Table 7, many of the comparisons were in fact duplicates. From the data, I concluded that with a lower-weighted LDA and an inconsistent web service score, having a *powerful* contextual score indicator such as the time stamp helped to keep many of the confirmed duplicates at high percentages. The context of time stamps was powerful enough in Classroom Salon to drive the percentages to approximately 50% in most of the tests in Table 7. By multiplying, the results have become better at telling what is a definite negative. In the comparison between 28060 and 28137, for example, the multiplicative percentage was only 27%, which is just under half the percent that it was when calculated the mean percentage. I conclude that context, unlike a WordNet or even LDA, can be so unique to a specific data set that it can identify certain features (in my case semantic similarity) at larger scales. Furthermore, the weight of the context in my calculations was *not driven by word synonyms*, a glaring weakness in WordNet approaches.

Annotation 1	Annotation 2	Results (multiply)
28160	28240	0.5863
27706	28001	0.3733
27953	28092	0.5556
28060	28137	0.2719
28100	27924	0.4514
...	...	...

(a) Video 1

Annotation 1	Annotation 2	Results (multiply)
25025	25255	0.7323
27112	27113	0.6323
25064	25157	0.4971
24856	25143	0.5470
25145	25255	0.5040
...	...	...

(b) Video 2

Annotation 1	Annotation 2	Results (multiply)
26150	25700	0.6050
25574	26042	0.3643
26181	25700	0.5441
25660	25574	0.4689
25856	25540	0.4602
...	...	...

(c) Video 3

Figure 7: Multiplied Results

5. Related Work

5.1. Background research

In beginning my research into semantic similarity, I first came across topic-modeling. Specifically, I looked at the research performed by researchers at Columbia University (Barzilay and Lee, 2003) [2]. Their goal was to paraphrase large bodies of text using a learning algorithm called Multiple-sequence alignment. What I drew from their research for my experiments was the similarities between the *paraphrased* form of a document and the semantics of sentences. In order to paraphrase a text, one must be able to ascertain the unique sentences in a text apart from the ones that are either redundant or do not add anything new. Her team was able to generate good results by focusing on

data that was not contained within the words. With that in mind, this set me on the path to learning how context could aid in experimenting with the semantic equivalence problem..

Next, I began to look at some of the more traditional approaches to minimizing semantic distance. Researchers at Columbia University saw the limitations of WordNet; that a term had a fixed number of senses, and that some of the different senses were actually too specific to be able to find the general meaning of a phrase. What they proposed was that a word would have a *dominant* meaning in the context of the text (Jing and Tzoukermann, 2001) [4]. For example, if the document was an article about financial institutions, the word bank most likely involves the actual financial institutions, as opposed to river banks. Using this intuition, they were able to show improvements by utilizing local document information and a global corpus to calculate the distance in context between two words.

Other research involved approaching involving performing statistical analysis on related concepts as opposed to the words themselves (Mohammad and Hirst, 2008) [7]. Researchers from the Universities of Maryland and Toronto created vectors from thesaurus concepts as opposed to words themselves. The thesaurus they used had a small number of concepts (approximately 1000), and with those concepts in mind, given a word, there would be a small of amount of categories it could fit in (i.e. star can be a celestial body or celebrity, for example), and those categories have a bunch of words in them that are ranked based on their occurrence in a corpus.

This research pushed me to think about ways to experiment with semantic equivalence without having to focus as much on the terms themselves. With that in mind, I tried to look at how given enough contextual information, one could ascertain how close two sentences were semantically. I realized that I did not have to show what a sentence meant, just that two sentences had a low semantic distance from one other. And Classroom Salon offered the contextual data to experiment with the idea of context, and how it could improve WordNet-based approaches to the problem.

6. Conclusion

6.1. Future Work

The data from my experiments showed a measurable improvement in the likelihood of semantic similarity between two phrases. The measured improvement stemmed from having a data source that provided contextual data. Furthermore, the specific data source that I used in my experiments, Classroom Salon, had users create comments at specific points on a given video or text. As a result, the fact that the annotations are more than just text provide useful data in assessing semantic distance.

This is both a strength and a weakness in tackling the problem. The strength is the access to the extraneous data that is very useful. However, this also narrows my approach to the problem. Not all data points will have such a wealth of contextual information (on many websites, their comments sections do not note where in the article/video the user is commenting about). Furthermore, while I used the time stamp data from the source, I did not take into account some of the nuances of videos. For example, some instructional videos may have review sections at the end - so in this corner case, some comments that may be duplicated could be found at far away time stamps. In any sort of semantic equivalence set where one is dependent upon the *source* for the context as opposed to the *terms*, there may be certain artifacts about the source material that make the contextual data a little less reliable. Still, in my paper many of the time points correlated with, if not semantic duplicates, then at least comments that were semantically related (i.e. a reply to another student's question).

For further improvements to this problem, other possible approaches include trying to assess whether a phrase is a question or just a command. A sentence that is a question may have a shared topic with a command, but these two phrases cannot be exactly duplicates; one asks a question, the other makes a statement. By narrowing down the cases for what a sentence can be (command, question, statement, etc.), contextual information can be gleamed before having to look for an outside source. And in combination with as much context about the document as I did in my experiments, the combined information can supplement word sense to increase the likelihood that two sentences are

the same. In many real-life scenarios, phrases rarely-ever lack contextual data. This is how human are able to make sense of phrases. Rather than trying to pull context from the phrases themselves, it is worthwhile to work backwards and find the meaning of sentences through contextual information.

I pledge my honor that I have not violated the Honor Code during this paper. Mckervin Ceme.

References

[1] J. Barr and A. Gunawardena, "Classroom salon: A tool for social collaboration," in *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '12. New York, NY, USA: ACM, 2012, pp. 197–202. Available: http://doi.acm.org/10.1145/2157136.2157196

[2] R. Barzilay and L. Lee, "Learning to paraphrase: An unsupervised approach using multiple-sequence alignment," in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, ser. NAACL '03. Stroudsburg, PA, USA: Association for Computational Linguistics, 2003, pp. 16–23. Available: http://dx.doi.org/10.3115/1073445.1073448

[3] L. Han *et al.*, "Umbc ebiquity-core: Semantic textual similarity systems," in *Proceedings of the Second Joint Conference on Lexical and Computational Semantics*. Association for Computational Linguistics, June 2013.

[4] H. Jing and E. Tzoukermann, "Determining semantic equivalence of terms in information retrieval," in *Natural Language Processing*. John Benjamins Publishing Company, 2001, pp. 245–260. Available: http://dx.doi.org/10.1075/nlp.2.13jin

[5] A. K. McCallum, "Mallet: A machine learning for language toolkit," 2002, http://mallet.cs.umass.edu.

[6] J.-B. Michel *et al.*, "Quantitative analysis of culture using millions of digitized books," *Science*, vol. 331, no. 6014, pp. 176–182, 2011. Available: http://www.sciencemag.org/content/331/6014/176.abstract

[7] S. Mohammad, "Measuring semantic distance using distributional profiles of concepts," Ph.D. dissertation, University of Toronto, 2008.

[8] M. Wallace, *Jawbone Java WordNet API*, 2007. Available: http://mfwallace.googlepages.com/jawbone.html

A.

Experiment Code

```
/*************************************************************************
 * Name:      Mckervin Ceme
 * Login:     mceme
 *
 * Dependencies: java.lang, java.net, java.io, Queue.java,
 *            SeparateChaningHastST.java, UMBC STS Service
 *
 * Description: This program calculates a score that utilizes the web
 *            service provided by UMBC, the percentages of each phrase
 *            in its topic (from the LDA), and the context (time stamp)
 *            when the comment was made to calculate a similarity
 *            percentage.
 *
 * Compilation: javac Score.java
 * Execution: java Score
 *
 *************************************************************************/
import java.lang.*;
import java.net.*;
import java.io.*;

public class Score {

   private SeparateChainingHashST<Integer, Queue<Annotation>> values;
   private double stamp;

   /*************************************************************************
    * Annotation helper class
    *************************************************************************/
   private class Annotation {
      private int id;      // Annotation Number
      private int topic;   // Major topic
      private double score; // Major topic percentage
      private String text; // Annotation text
      private double context; // time stamp of Annotation

      public Annotation(int id, int topic, double score,
         String text, double context) {
         this.id = id;
         this.topic = topic;
         this.score = score;
```

```
      this.text = text;
      this.context = context;
   }
}

/*************************************************************************
 * Fix command line arguments
 *************************************************************************/
private String forwardSlash(String s) {
   return s.replace("\\", "/");
}

/*************************************************************************
 * Cleave sentences for the web service
 *************************************************************************/
private String cleanUp(String s) {
   String result;

   result = s.replace("%", "%20percent");
   result = result.replace(" ", "%20");
   result = result.replace(".", "");
   return result;
}

/*************************************************************************
 * Score constructor. Change documents into iterable list of annotations
 *************************************************************************/
public Score(String topics, String data) {
   values = new SeparateChainingHashST<Integer, Queue<Annotation>>();

   try {
      // open both files
      BufferedReader buckets = new BufferedReader(new FileReader(
         forwardSlash(topics)));
      BufferedReader comments = new BufferedReader(new FileReader(
         forwardSlash(data)));

      String bucketLine; String commentLine;
      while ((bucketLine = buckets.readLine()) != null) {
         String[] temp = bucketLine.split("\t");

         commentLine = comments.readLine();
         // creates annotation and places it inside hash table
         if (commentLine != null) {
            String[] temp2 = commentLine.split("\t");
            String[] clock = temp2[2].split(":");
            double time = Double.parseDouble(clock[0])*60.0;
```

```
            time += Double.parseDouble(clock[1]);

            Annotation annotation = new Annotation(
               Integer.parseInt(temp[0]),
               Integer.parseInt(temp[1]),
               Double.parseDouble(temp[2]),
               temp2[1],
               time
               );

            if (!values.contains(annotation.topic)) {
               Queue<Annotation> q1 = new Queue<Annotation>();
               q1.enqueue(annotation);
               values.put(annotation.topic, q1);
            }
            else {
               Queue<Annotation> q2 = values.get(annotation.topic);
               q2.enqueue(annotation);
               values.put(annotation.topic, q2);
            }
            stamp = time;
         }
         else break;
      }
      buckets.close(); comments.close();
   }
   catch(IOException e) {System.out.println("Failed Here.");}
}

/**************************************************************************
 * One method for determining semantic similarity.
 *************************************************************************/
private void simScoreByTopic(int id, Annotation a1, Annotation a2,
   double simIndex, boolean method) {
   double result = 1.0;
   double temp = 0.0; double avg = 0.0;

   // The percentage of how much the two phrases are in the same topic
   if (method) {
      avg = ((a1.score + a2.score) / 2.0);
      result += avg;
   }
   else {
      avg = a1.score * a2.score;
      result = avg;
   }
```

```
   // The context (time stame for video) difference
   if (a1.context == a2.context)
      temp = 1.0;
   else
      temp = 1.0 - (Math.abs((a1.context - a2.context)) / stamp);

   result += temp;

   // And finally, the similarity score from the web service.
   result += simIndex;
   if (method) {
      result = result / 4.0;
      System.out.format("%d\t%d\t%d\t%f\t%f\t%f\t%f\t%f%n",
         id, a1.id, a2.id, 1.0, avg, temp, simIndex, result);
   }
   else {
      result = result / 3.0;
      System.out.format("%d\t%d\t%d\t%f\t%f\t%f\t%f\t%f%n",
         id, a1.id, a2.id, 0.0, avg, temp, simIndex, result);
   }
}

/*************************************************************************
 * Calculate score by topic
 *************************************************************************/
public void calculateByTopic(boolean type) {
   // the visited array prevents the same two annotations from being
   // compared to one another, once they already have. This also
   // prevents annotations from comparing to themselves.
   int N; boolean visited[][];
   int i = 0; int j = 0;
   String s1; String s2;
   String base =
       "http://swoogle.umbc.edu/StsService/GetStsSim?operation=api&phrase1=";

   for (int id : values.keys() ) {
      Queue<Annotation> q = values.get(id);
      N = q.size(); i = 0; j = 0;
      visited = new boolean[N][N];
      for (Annotation x : q) {
         s1 = cleanUp(x.text);
         if (i == N-1) break;
         for (Annotation y : q) {
            if (i == j) visited[i][j] = true;
            else if (visited[i][j] == false)
            {
               s2 = cleanUp(y.text);
```

```
                try {
                   // the code to call the web service
                   URL comparison = new URL(base+s1+"&phrase2="+s2);
                   BufferedReader in = new BufferedReader(
                   new InputStreamReader(comparison.openStream()));

                   String line;
                   while ((line = in.readLine()) != null)
                      simScoreByTopic(id, x, y,
                         Double.parseDouble(line), type);
                   in.close();
                   visited[i][j] = true; visited[j][i] = true;
                } catch(IOException e) {System.out.format(
                   "Id1: %d\tId2: %d\ti: %d j: %d%n",
                   x.id, y.id, i, j);}
             }
             j++;
          }
          i++; j = 0;
       }
    }
}

/**************************************************************************
 * Testing method
 *************************************************************************/
public static void main(String[] args) throws Exception {
   String topics;
   String context;
   int j = -1;
   int prevContext = 0;
   int comp = 0;
   int total = 0;
   boolean type = true; // if type, then the calculation with the
                        // probabilities does the average, else when
                        // calculating the final result, the
                        // probabiliities are multiplied.

   File fTopics = null; File fContext = null;
   File[] topicsPath; File[] contextPath;

   System.out.println("Beginning Data Mining...");
   System.out.println("------------------------");

   try {
      fTopics = new File(".\\data\\topics");
      fContext = new File(".\\data\\context");
```

```
topicsPath = fTopics.listFiles();
contextPath = fContext.listFiles();

// iterate through two folders. The Topics folders contain files
// organized like so:
/*********************************************
25863 8  0.287596 ......

The first column is the topic id, the second is its major topic,
and the third is its percentage. There is a file for different
LDA runs, as well as files for different documents.
The files are titled <name of document>_n-x.txt

n is the topics, x is the optimization, and the name is a
4-digit number to identify a unique video.

The context folder, however, just has the comments for every
unique video, with files name <integer>.txt. These are
organized like so:

25896 <text> <time context>

The first column is the topic id, the second is the comment,
and the third is its time context - when in the video
it was posted.
*********************************************/
for (int i = 0; i < topicsPath.length; i++) {
   topics = topicsPath[i].toString();
   topics = topics.substring(2, topics.length());

   String params = topics.replaceAll("[^0-9]", "");
   comp = Integer.parseInt(params.substring(0,4));

   if (comp != prevContext) j++;
   context = contextPath[j].toString();
   context = context.substring(2, context.length());

   String numberOnly = context.replaceAll("[^0-9]", "");

   params = params.replaceAll(numberOnly, "");
   params = params.substring(0,2) + "-"
      +params.substring(2,params.length());
   FileOutputStream ans = null;

   System.out.format("File: %d%nCID: %s%nOptimize Pattern: %s%n",
      i, numberOnly, params);
```

```
            try {
               if (type)
                  ans = new FileOutputStream("tests\\"+numberOnly
                                          +"_"+params+".txt");
               else
                  ans = new FileOutputStream("mults\\"+numberOnly+"_"
                                          +params+".txt");

               PrintStream stdout = System.out;
               System.setOut(new PrintStream(ans));
               System.out.println("Content Id: "+numberOnly
                                  + "\tOptimization: " + params);
               if (type)
                  System.out.println(
                     "Topic:\tId1:\tId2:\tSame\tAverage\tContext\tService\tTotal");
               else
                  System.out.println(
                     "Topic:\tId1:\tId2:\tSame\tMultiply\tContext\tService\tTotal");

               Score test = new Score(topics, context);
               test.calculateByTopic(type);

               prevContext = Integer.parseInt(numberOnly);
               total++;
               System.setOut(stdout);
               System.out.println("Finished File: " + i + "!");
               System.out.println("------------------------");
            }
            catch (IOException e) { System.err.println("Failed"); }
            finally { if (ans != null) ans.close(); }
         }
      } catch(Exception e) { e.printStackTrace(); }
      if (total == 1) System.out.println(total + " file mined.");
      else System.out.println(total + " files mined.");
   }
}
```