Learning and Analyzing Topics in Human Language

New Mexico
Supercomputing Challenge

Final Report
April 3, 2013

Team 47
La Cueva High School

Team Members:

Ari Echt-Wilson
Eli Echt-Wilson
Justin Sanchez

Teacher:

Samuel Smith

Table of Contents

Executive Summary

In this project, we used statistical analysis to identify and analyze topics in human language.

Starting with a set of training data, our algorithm splits the words into bags using either a speaker-based or a proximity-based approach. Several changes are applied to "stem" the words so that similar words (plurals, etc.) are processed together. Then we calculate Pearson's Correlation between the word counts in corresponding bags to identify words that typically appear together. We implemented this algorithm in Java. For this report we applied our method to politics, using 21 Presidential debates and speeches totaling 261,000 words as training data. However, our program could be applied to arbitrary sources and topics.

In order to reduce outliers and get significant results, we needed large amounts of data and computation. Our training data set took 4.5 hours to run on a desktop computer; applying our approach to more data would require more compute power. So we designed and implemented a parallel program in the form of a Hadoop MapReduce to find correlated words. We ran the MapReduce on up to 100 computers on Amazon Web Services. The speedup was linear with the number of computers used.

We then used correlations between words to group words into topics. We used a straightforward clustering algorithm to find groups of words that had high correlations with each other. Despite the simple clustering algorithm, the resulting topics are surprisingly accurate, reading like a list of hot topics debated in the election.

Finally, we used our learned correlations to analyze whether a candidate in a debate answered a question. Highly correlated words are considered "meaningful," and correlations between the meaningful words in the question and answer determine whether the response has the same topics as the question. To validate our results, we surveyed 36 people on SurveyMonkey to see how well people thought questions were answered. We compared the computer's answers to the results of the survey, and our results were closely related except for one discrepancy.

1. Introduction

Automatic computer analysis of text could have many important uses. For example, fact-checking in political campaigns has become essential in recent campaigns like the 2012 U.S. Presidential campaign. A computer program could help check facts in an unbiased way, or it could determine whether a candidate is answering or avoiding the questions being asked. Finding topics in texts could also help search engines like Google to find articles related to a particular topic. Finding other songs with similar lyrics could help music services like Pandora to be more accurate.

Human analysis of document topics is subjective and time consuming. Computerized methods to analyze text can do so in an efficient and objective manner. In order to do so, computers must be able to learn topics and relationships between words. If this is achieved, the results can be used to categorize documents, evaluate answers to questions, and perform other topic-related calculations.

2. Background

Ari went with her dad, Randy Wilson, to his 25-year college reunion. She attended lectures from many of his classmates on their current jobs and projects. One of these was a presentation by Phil Resnik. Mr. Resnik is a professor at the University of Maryland. He is working on topic analysis to show changes in topic, specifically in text with changes of speakers, like conversations or debates. For example, the work tries to replicate human intuition on which politicians go farther off topic. His work also focuses on *who* changes the topic and therefore who can control the debates [Nguyen, Boyd-Graber, and Resnik].

Prof. Resnik's work uses topics that were learned using complex statistical techniques. Some examples are Latent Dirichlet allocation (LDA) [Blei, Ng, and Jordan] or probabilistic latent semantic analysis (pLSA) [Hofmann]. Using only the collections of words in documents, they find "latent" variables corresponding to "topics," where a "topic" is a probability distribution over the set of known words. A document has a probable set of topics, each of which likely generates the words in the document. (Hence this is called a "generative model.") In LDA, the Dirichlet

distribution encourages each document to have a small set of topics, and for each topic to have a smaller set of words that it generates. Each document is treated as a "bag of words" – in other words, where the words appear in the document or relative to each other is completely ignored. The words are all mixed up in each bag. Complicated statistical programs called "approximate posterior inference algorithms" are used to generate the topics from a large set of documents, for example [Blei and Jordan].

Once topics and their relationships are known, there are lots of papers about how to cluster documents or divide/cluster topics themselves. For example see [Mei] and [Muscat].

Our program is a simplified version of the above research. Our program allows us to explore similar issues with a simpler way. It was encouraging to find that the way we designed the code made sense because other research involved these issues.

We decided to try to analyze debates through a more simple application with correlated words. Like Prof. Resnik, we treat each utterance (for example a debate question or answer) as a bag of words, without attempting to learn anything from where the words appear in the bag. However, we use simpler statistical analysis, namely Pearson's correlation, to try to learn "related" words. We cluster the words using these correlations to form topics. Then we use the word correlations to measure whether the topics in two bags are related or not. We compare the output of our program to human responses to a survey, and we were very encouraged by our results.

3. Learning Related Words

We used a Java program to analyze and learn topics from input files of text. Our program splits up text files into smaller groups of words called "bags", and then finds correlations between words in the bags. When adding learning data, the text is automatically separated into "bags" and the bags are added to a list of all the bags in the learning data. The positions of words relative to each other are not considered, which is why they are called "bags." If one word appears in lots of bags usually together with another word, then we decide that those two words are correlated, and that they are somehow related or part of the same topic.

3.1 Cleaning up words

We process the input text to "clean" it in a couple ways. This is also referred to as "stemming" words. First, we ignore some "unimportant" words such as "and," "if," "or," and other words that are obviously unimportant to our model. The list of all words we ignore is shown here:

```
"the", "i", "me", "in", "at", "my", "thank", "of", "im", "and", "you", "on",
"for", "be", "if", "a", "an", "by", "to", "--", "ive", "it", "ill", "do", "or",
"get", "from", "mr", "he", "she", "oh", "we", "go", "weve", "got", "as", "your",
"were", "what", "will", "with", "would", "then", "they", "theyre", "cheer", "but",
"now", "that", "when", "any", "make", "made", "instead", "isnt", "put", "this",
"thi", "wont", "are", "also", "mr", "have", "been", "our", "not", "want", "just",
"only", "going", "applause", "laughter", "about", "all", "because", "can", "each",
"every", "let", "more", "must", "one", "out", "said", "say", "them", "there",
"their", "these", "those", "well", "who", "thing", "stuff", "sure", "than",
"many", "both", "crosstalk"
```

Removing these words in the beginning mainly saves some computing time, since we expect that they would have low correlations. However, when we didn't remove the unimportant words, we actually found some correlations that come from common English phrases like "have been". We also considered removing all words that were less than five letters but this posed an issue that crucial words like "war" would be left out.

We also try to make the same word turn into the same Java String, so that all of the word correlations will get connected. We use the .toLower() function to make all words lower-case, so that a word beginning a sentence and in the middle of the sentence become the same. We also remove all punctuation like "(" and "." and quotation marks. This has the negative effect that it changes "don't" into "dont", but that is a small price to pay.

We also remove a final "s" from words, to try to make plurals look like the singular word. For example, "wars" becomes "war," which is the same topic. This works for most words but it isn't perfect: for example it also changes "taxes" to "taxe" and doesn't change "people" to "person." Computer companies have dictionaries that connect words and plurals, but we didn't have access to those so we had to do the best we could. But we think this step still gains more than it loses, so we left it in. We also relied on the amount of data that we had to make things like this not as important. The nice thing about computers is how much data they can look at in order to draw

conclusions so small outliers become less of an issue.

3.2 Splitting by Speaker

The first way we implemented it, our program splits up a large text file into smaller groups of words spoken by the same person either asking or answering a question. We split the debate this way by finding certain names followed by a colon (":"), like "Lehrer:" and "Obama:". So this method is assuming that each "question" or "answer" in a debate is about a single topic. That might not be true, but we hope that with enough learning data we can still find the good word connections and rule out the bad ones.

The main disadvantage of this method is that it only applies to input text that is explicitly separated by speaker name with a colon. So it's harder to get a large amount of input data for it. It also has the problem that in some debates, the speakers interrupt each other a lot, which splits up the bags incorrectly and has a lot of small bags. For example, here is a section from the second Obama/Romney 2012 Presidential Debate:

> ROMNEY: But that's not what you've done in the last four years. That's the problem. In the last four years, you cut permits and licenses on federal land and federal waters in half.
> OBAMA: Not true, Governor Romney.
> ROMNEY: So how much did you cut (inaudible)?
> OBAMA: Not true.
> ROMNEY: How much did you cut them by, then?
> OBAMA: Governor, we have actually produced more oil --

You can see how this would be difficult to learn topics from, when splitting the words by speaker.

3.3 Splitting by Proximity

Another way to find the correlation between the words that we considered was by using proximity as an indicator that two words are correlated. In other words, if two words appear close to each other, then they should be related. We did this by separating the words into smaller groups, or "bags" of 100-200 words and recording the counts of each word in the bag.

We record the counts of each word in the bag and add half of the counts of the word from the bag before and after it. For example, say we have bags A, B, and C that appear in that order, and word X appears 1, 2, and 3 times in these bags, respectively. When recording the counts for word X in bag B, the program would add 2.0 from bag B, 0.5 from bag A, and 1.5 from bag C, for a total count of 4.0. This helps make it so that closer words (in the same group) have a stronger effect on the correlation than words farther away (in the group before or after).

In addition, if two related words happen to end up one at the end of one bag and the other at the start of the next bag, we don't want our program to determine that those two words are totally unrelated. Adding half the counts from the adjacent bags is a good compromise to take this into account while keeping the integrity of the "proximity" idea.

The advantage of using proximity to determine correlation over our original method is that it utilizes a much simpler method of dividing the words into groups. This way, we don't have to worry about dividing the words up by speaker for a debate versus by paragraph for a speech for example. In fact, this program can analyze many speeches and debates combined in a single text file. While there is some more noise with this method from words at either "end" of a group being left out of the adjacent groups or from groups of words that happen to span a sudden change in topic, the fact that we will be able to analyze a lot more data seems to render this noise insignificant.

Another way we could split words into connected groups would be to detect the end of each paragraph (by a line break) or each sentence (by a period). We haven't tested those methods.

3.4 Calculating Correlations

After cleaning the text and separating it into bags, we calculate how each word in the text correlates with each of the other words. In our project, to say that two words are "correlated" means that they are likely to show up in the same bag of words a similar amount of times. For example, in our analysis we found that "energy" and "oil" had a relatively high correlation. So if in

one response Obama said "energy" three times, then he probably said "oil" in the same response, and vice versa.

We used Pearson's Correlation [Starnes, Yates, and Moore] to calculate this correlation between words. Pearson's Correlation is defined as:

$$r = \frac{1}{n-1}\sum_{i=1}^{n}\left(\frac{X_i - \bar{X}}{s_X}\right)\left(\frac{Y_i - \bar{Y}}{s_Y}\right)$$

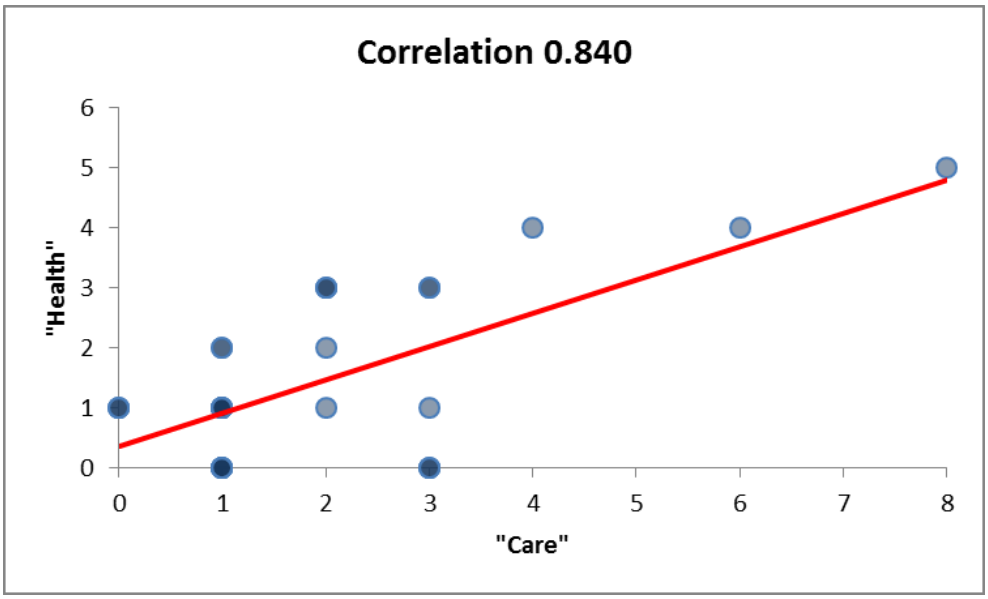
X_i and Y_i are individual values in a set of data on variable X and Y. The means are $\overline{X}$ and $\overline{Y}$ for the different data sets and the standard deviations are s_X and s_Y respectively. n is the number of data samples. Pearson's Correlation measures the strength of the linear relationship between two quantitative variables. It is always a number between -1 and 1. If it is negative it has a negative association and positive a positive association. The values that are closest to 0 are very weak. A perfect linear relationship has the correlation of 1 or -1. We saw very few negative correlations in our data; we think this is because uncorrelated words still appear near each other occasionally, with basically no correlation.

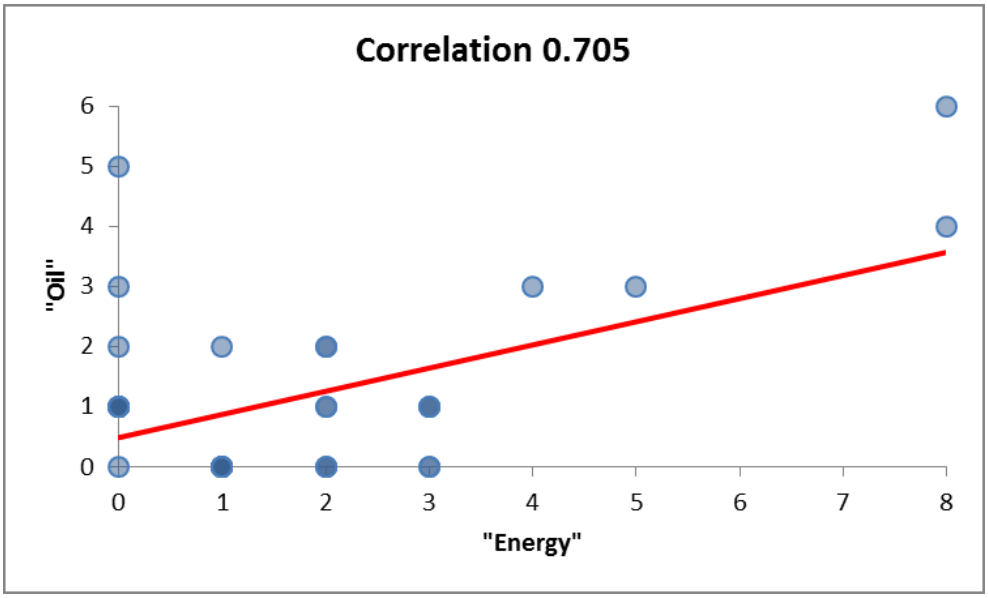
We decided that a linear relationship could be used because if words are correlated, then they should have a linear relationship. If something like "middle" appeared a certain number of times in all the bags, then "class" would appear a similar number of times in all the bags. Dr. Thomas Robey, a Supercomputing Challenge reviewer, suggested that we might use Spearman rank correlation. Spearman has advantages that it is more resistant to outliers and it finds nonlinear correlations. We tried using Spearman rank correlation, but it did not give results that were very different than Pearson's, so we stuck with Pearson's.

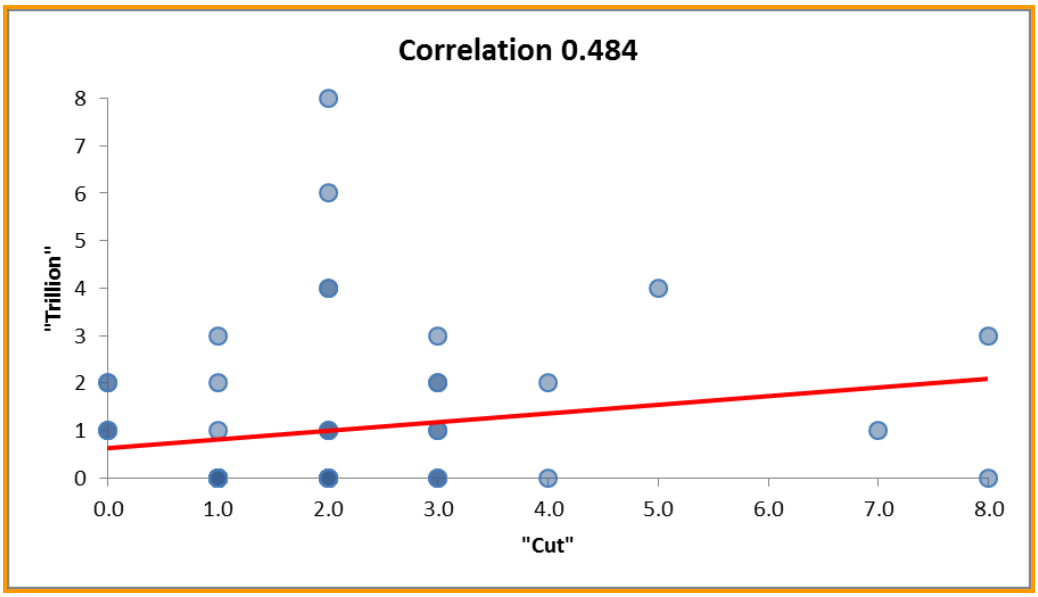
We do not correlate words that only appear in five bags or less. This eliminated the results we were seeing with correlations of 1 because the pair of words only appeared in one bag.

Below are several examples of scatterplots with the correlations between pairs of words. Each axis represents the counts of the respective words within each "bag." A high correlation means that when one of the words has a higher count in a bag, the other word should also appear a high

number of times in that bag, making the scatterplot show a linear relationship.

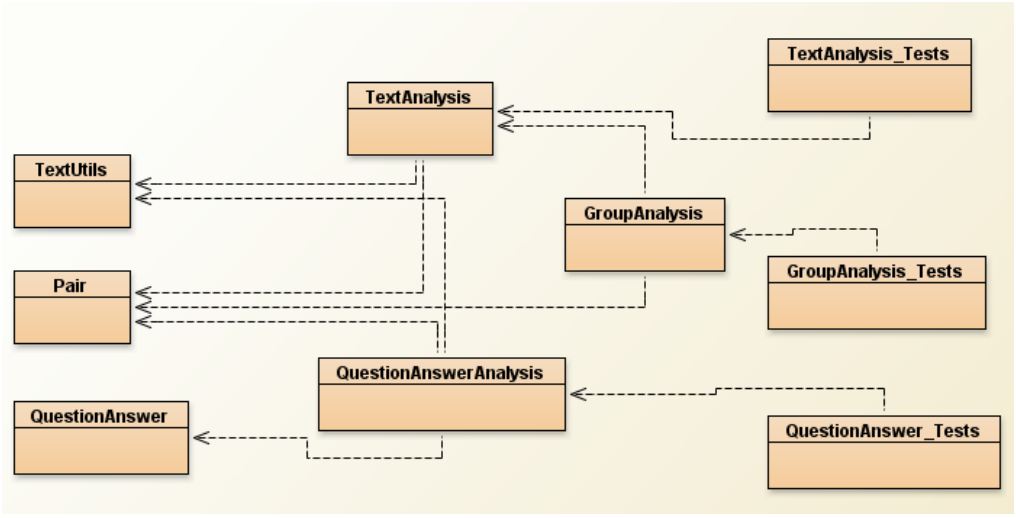






3.5 Program

This figure shows an overview of our program structure from BlueJ, showing the main classes and files. The boxes on the right build on boxes on the left, with an arrow meaning "this box uses that box."



Here are the main pieces and what they do:

- **Pair** represents a pair of words and their correlation.
- **TextUtils** implements "cleaning" of the words (Section 3.1).
- **QuestionAnswer** represents a question and its answer.

- **TextAnalysis** does everything involved in analyzing the text, from separating the words into bags to iterating over all pairs of words and calculating their correlations.
- **GroupAnalysis** uses the correlations between words to form "topics" (Section 5.1).
- **QuestionAnswerAnalysis** finds meaningful words in the question and the answer and calculates the correlation between them (Section 6).
- The three **Test** classes, **TextAnalysis_Tests**, **GroupAnalysis_Tests**, and **QuestionAnswer_Tests** utilize the functions of their respective "testee" classes to run the analyses and print the data to an output.

The full Java code listing is online, and Appendix D links to it.

3.6 Data and Correlations

Our training data included:

- the three Presidential debates in 2012,
- the Vice Presidential debate from 2012,
- 10 Republican primary debates from 2011-2012,
- Presidential State of the Union addresses from 2008, 2010, and 2013, and
- Four speeches by Obama and Romney from 2011 and 2012.

We downloaded all the text from the world wide web and saved it in text files.

Our debate input training data only used the four Presidential and Vice Presidential debates. That is because it becomes difficult to split the text based on all the different speakers, both debaters and moderators. This training set had 30,554 total words, 3,880 distinct words, and 1,118 bags of words. The program found 17,587 correlations over 0.2, 1,354 over 0.4, 154 over 0.6, and 14 correlations over 0.8.

Using proximity-based splitting we were able to use speeches and additional debates, and we had a total of 261,547 words, 9,282 distinct words, and 1,900 bags. The program computed 1.15 million Pearson's correlations, finding 719 correlations over 0.2, 185 over 0.4, 42 over 0.6, and 13 correlations over 0.8. The correlations seemed to be lower with the proximity version, but also had

less “unexpected” correlations, so you could consider it higher quality. We think that is because of the larger amount of training data that could be used for proximity.

Here are some sample correlations using the full proximity-based training data set. First, some expected high correlations that come from topics and phrases:

bin laden 0.9825529924023757
food stamp 0.7771788337846931
parenthood planned 0.932143074172159

Other strong correlations come from related words in topics:

$716 medicare 0.594464515567752
auto uaw 0.5053263316945532
bankruptcy managed 0.7163429210267714
consulting lobbying 0.49591336527358953
island puerto 0.6636766906748348
nasa space 0.7114747613448698

Some correlated words are artifacts of the training data set:

biden ryan 0.8087900972820965
candy crowley 0.5206227019428403
presidential twominute 0.8093348225428425

And finally a few correlations are unexpected. We think that at least some of these are from the small size of our training data: some correlations will happen by chance or coincidently, but a larger sample will decrease the appearance of them.

chile saving 0.4548156135504716
complete journey 0.7728863218661913
nominee specter 0.6472875395346757

The correlations over 0.4 found in the full proximity data set are shown in Appendix A.

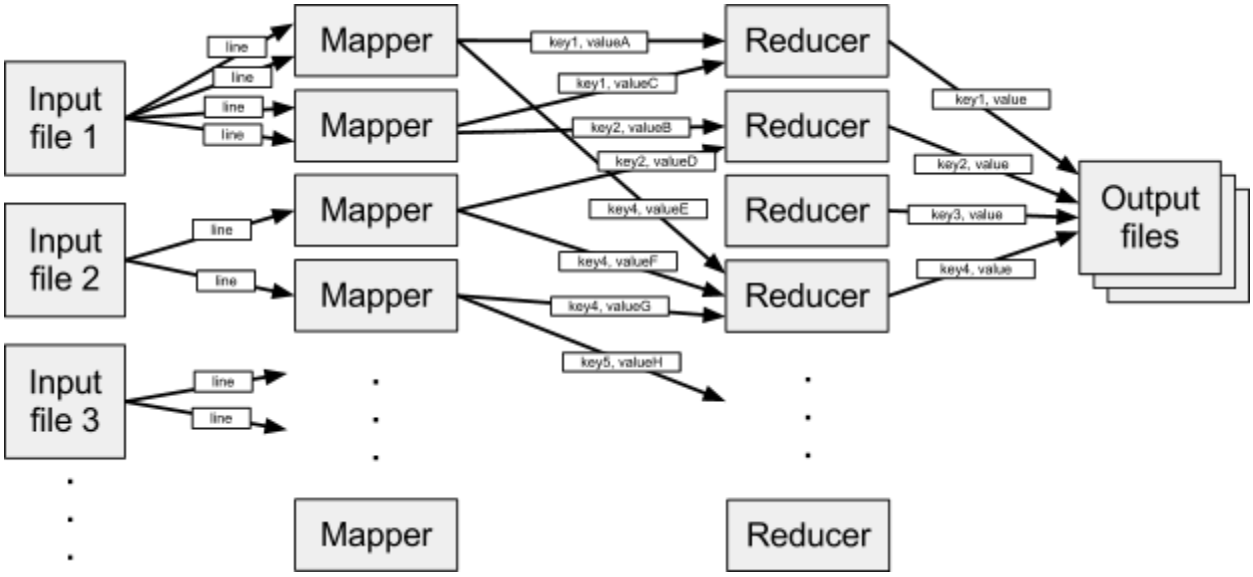
4. MapReduce to Learn Correlations

As we added more data, in order for our program to learn the correlations between words and be

able to analyze topics, it took longer for our computer to find the outputs. Calculating Pearson's correlation for all of the pairs of words in the data was what was taking up the vast majority of computing power. For all 21 debates and speeches, it took 4.5 hours to compute all the word correlations. While it may seem large to us, this data set is very small for a real-world data set. We knew that if we continued adding data, eventually it would take our computer a very long time. To solve this problem, we decided to write a parallel program called a MapReduce for our program. We used Hadoop on Amazon Web Services to implement our MapReduce.

4.1 How MapReduce Works

This diagram shows a basic outline of a MapReduce:



There are four parts to a MapReduce: input, Mappers, Reducers, and output. The input gets split up into pieces – for text files, single lines of text – and passes each piece to a Mapper. The Mapper performs a function on that input piece and passes the Reducers key-value pairs. The key indicates which Reducer it goes to and the value is what value is passed to that Reducer. The key and value can be anything, like strings, ints, or doubles. The Reducer then performs a function on all of the values (from all of the Mappers) for a single key, and produces the output. A single master computer coordinates the computation but doesn't touch any data. The idea is that there can be any number of computers running as Mappers and Reducers. If the program is written right, then with

10 computers, the program runs 10 times the speed compared to having one computer, and with 100 computers, the program runs 100 times as fast. This should be followed with the same application for 1,000 and 10,000 computers. [MapReduce]

Programming a MapReduce makes you put your algorithm into a certain form. Basically you have to split the computation into what is done in the Mapper and what is done in the Reducer. The Hadoop system handles connecting the keys and values from Mappers to Reducers, and organizing which computers process which pieces of information.

4.2 Correlations MapReduce

Our MapReduce takes the input of the bags of speeches and the lists of words and outputs a list of all of the pairs of words and their correlations to each other. Basically, the Mapper handles everything having to do with a single bag of words and sends word pairs and their counts to the Reducer. The Reducer handles the correlation between a single pair of words among all the bags. Hadoop efficiently gets all the information from the Mappers to the Reducers. The mapper takes very little time because it doesn't have much work to do compared to the reducer which has to calculate Pearson's correlation and therefore takes more computing time. The MapReduce speeds up calculating all the correlations because it splits the computing work among a large number of computers.

Below are the descriptions of how the mapper and reducer algorithms work for us to compute correlations between pairs of words:

Mapper: Each mapper gets as input one bag of words, the list of all words, and the counts of each of the words in the bag. The mapper generates all the word pairs in the bag with every word on the list. For every word pair, the mapper sends an output key-value pair to the reducer. The key is the word pair in alphabetical order; in other words, the word that comes first in the dictionary, comes first in the word pair. For example, for "soldier" and "afghan" the key would be "afghan,soldier". The value is a pair of counts of each of those two words in that particular bag. Both the key and the value are represented as strings. The Hadoop system then sends these

key-value pairs to the reducers.

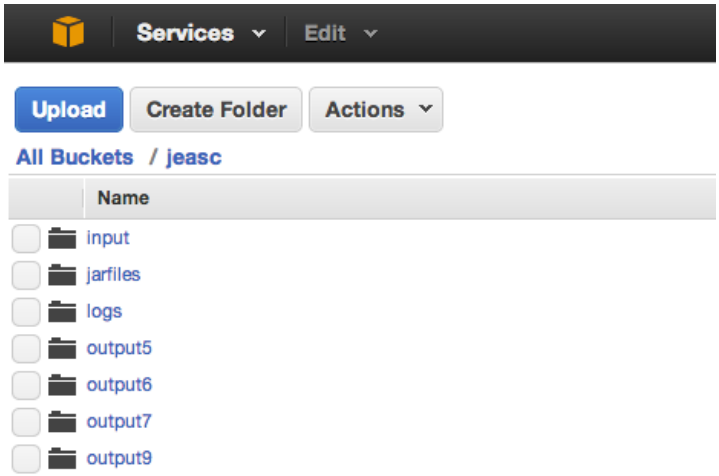
Reducer: Each reducer gets as input all of the key value pairs that have the same key, i.e. the same pair of words. Each reducer then creates two arrays; the first array is for the list of the number of times the first word appears and the second array is for the list of the number of times the second word appears. Pearson's correlation is then run inside the reducer using the two arrays as the two pieces of quantitative data needed to evaluate the strength of the linear relationship between the two words. If the correlation is higher than a predetermined number (for example 0.4) then the reducer sends the word pair and the correlation to the output.

After the MapReduce is run, a list of output files contain all of the word pairs with a certain level of correlation from the input data.

We implemented the MapReduce in Java using Hadoop. The code is listed in Appendix E.

4.3 Running the MapReduce

We used Eclipse for Windows to create a .jar file for our Hadoop program, and then uploaded it to Amazon S3, which is where you store data and programs on Amazon Web Services (AWS). Here is a view of our top-level "bucket" on AWS:



Our Amazon bucket is called "jeasc"; it stands for Justin, Eli, Ari SuperComputing.

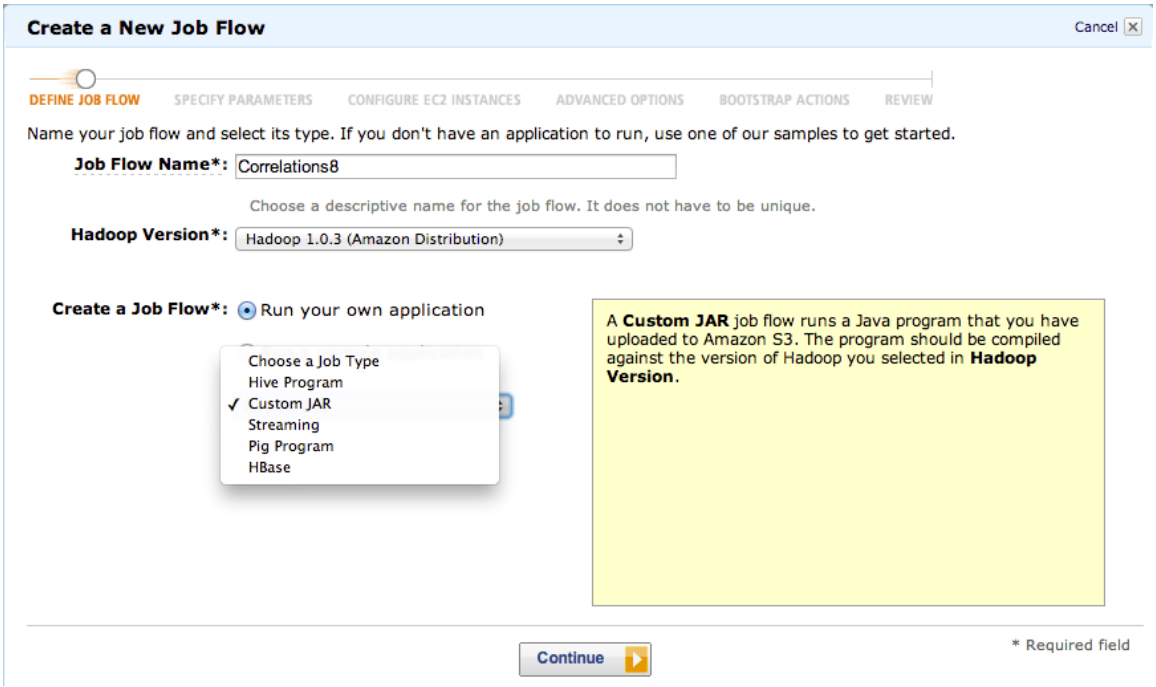
"jarfiles" contains the Java archive (executable program) correlations.jar.

"input" contains the input files.

the output of each Hadoop run goes in a new "output" folder.

"logs" holds log files that help you figure out how the job worked or why it failed.

To run a MapReduce, you use AWS Elastic MapReduce to create a new job flow, which looks like this:



There are several more screens of input that give the inputs and flags to the program, how many Mapper and Reducer computers to use, etc. AWS calls the computers "instances." The computer time costs $0.06 per hour per instance, but you get charged for any fraction of an hour that you use, so even if your program only runs for 5 minutes, you're still charged for several instance-hours.

The MapReduce program to calculate correlations gets exactly the same results as our BlueJ Java version does, no matter how many instances are used.

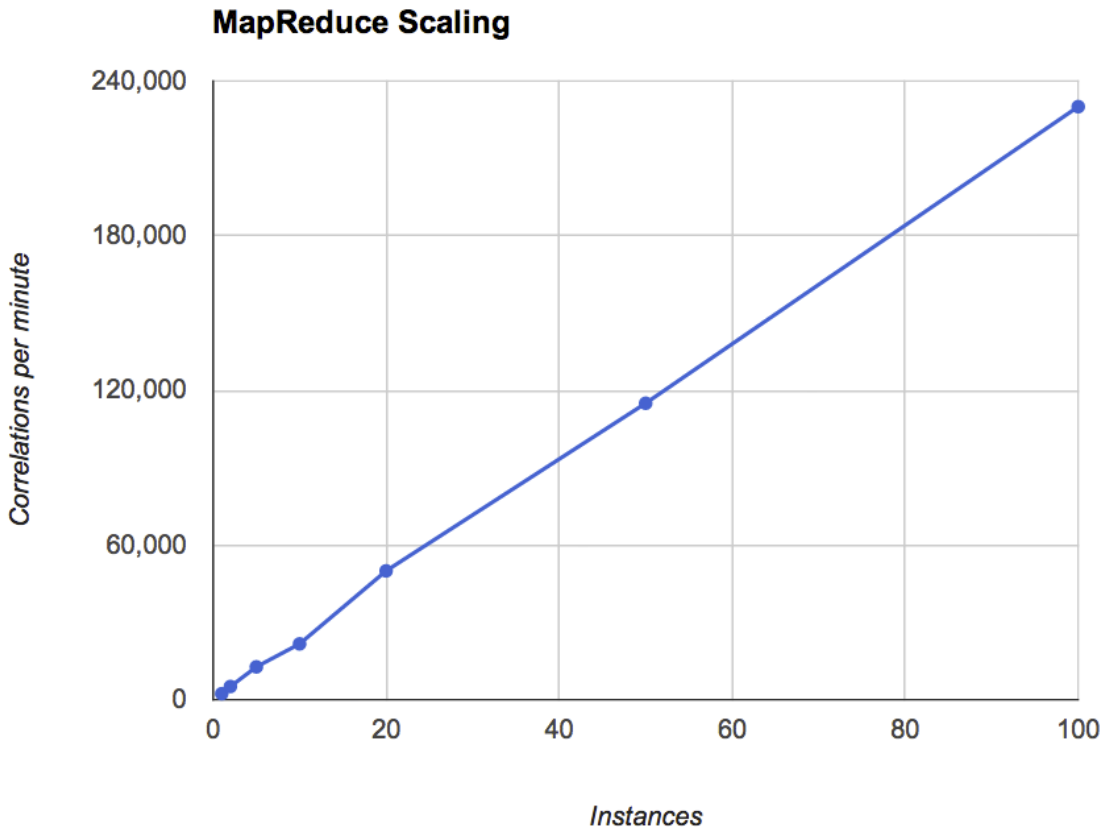
As expected, the program runs faster when more instances are used to run it. Here is a table of

times that we ran it.

Instances	Total time (minutes)	Time after startup	Instance-minutes after startup	Correlations per minute	Correlations per instance-minute
1	488	484	484	2,376	2,376
2	227	223	446	5,157	2,578
5	94	90	450	12,778	2,556
10	57	53	530	21,698	2,170
20	27	23	460	50,000	2,500
50	14	10	500	115,000	2,300
100	9	5	500	230,000	2,300

The first column shows the number of instances used, and the second shows how many minutes it took to run. Even if the mapreduce does nothing (or crashes immediately, which we did several times), it takes about 4 minutes. So the third column shows the compute time after subtracting this fixed startup/shutdown time. The fourth column shows the total instance-minutes of computer time, after subtracting startup time (just multiplying the first and third columns). The fifth column shows the number of correlations per minute the MapReduce tested (1.15 million divided by the total time). Finally, the last column shows the number of correlations per instance-minute, i.e. how many correlations each instance computed per minute.

The graph below shows how our mapreduce scales with increasing computing resources:



With up to 100 computers, our MapReduce scales linearly with the number of instances. Another way to state this is that correlations per instance-minute is constant. This is a very desirable trait in a parallel computer program: it means that if you have more data to process, you can use always use more computers to complete the computation in the same amount of time.

5. Grouping Words into Topics

A topic can be considered a group of related words, that together describe a concept or set of related concepts. Topics can be very useful for determining which speakers or documents are talking about which subjects. We used the word correlations that we found from our learning data, and grouped sets of related words into topics.

5.1 Algorithm

When grouping words into topics, we made the basic assumption that if word A is strongly correlated to word B, and word B is strongly correlated to word C, then words A, B, and C form a topic. In order to create these topics, we started with a single word and found all words correlated to it, at a correlation of at least T, where the correlation threshold T can be chosen experimentally to get the best results. We repeated the process on all of the words in the topic until we found no more correlated words. Then the program chooses another word not part of any current topics, and builds a topic from that, and so on. Using this, the computer created many groups of words that are correlated to each other that we can call a topic.

Our algorithm is risky because there is a chance that a single tie between words in groups that are not related will lump the two groups together (overclustering). However, our results show very few instances where this characteristic is shown. There are much better clustering algorithms that could be the topic of reports by themselves, such as the ones used in [Blei and Jordan], [Hofmann], and [Muscat]. Our simple approach gave very good results, and a more powerful clustering algorithm would no doubt improve on them.

5.2 Clustering Results

Using the proximity-based correlations, we found that a choice of T = 0.2 gave us a very good clustering of words into topics, without over-clustering in most cases. Our results show very promising groups of words that form topics, and they read like a list of hot topics from the 2012 U.S. Presidential election.

These include:
Financial: $16, bond, fannie, freddie, investment, looked, mac, mae, mutual, pension
Abortion: abortion, accept, adoption, catholic, church, conception, conscience, doctrine, embryo, hospital, life, none, rape, regard, religious
Energy: advanced, agency, air, cell, clean, climate, coal, dakota, doesnt, drilling, energy, epa, gallon, gasoline, hampshire, increasingly, keystone, land, license, natural, new, oil, permit,

pipeline, pollution, price, production, renewable, solar, water, wind
Navy: aircraft, carrier, gulf, navy, sea, ship, smaller
Benghazi: ambassador, attack, benghazi, deny, fewer, fighting, force, iraq, iraqi, letter, month, occurred, season, september, stimulu, summer, surge, terrorist, trained, troop, video, youtube
Immigration: amnesty, apply, arizona, become, card, citizenship, come, country, deport, employer, everify, expres, grandfather, grandmother, here, hire, illegal, illegally, immigrant, immigration, legal, legally, lorraine, magnet, pathway, people, permanent, question, residency, resident, sanctioned, selfdeportation, stay, visa

There are a few words scattered in the topics that should probably not be in there, like "looked," "accept," "none," and "regard.". We think those are coincidences in our training data set, and they should go away with even more training data.

Many of the topics we found are clearly recognizable, so we gave them names above. But as [Blei] points out, it is dangerous to assign names to these created topics, because humans might assign incorrect meaning to the topics. It is best to understand and recognize that the computer has formed a topic, but refrain from categorizing the topic under a single name. Instead, the topic is formally defined as simply a set of related words.

The list of clusters based on correlations greater than 0.2 are listed in Appendix B. The program output includes many groups of only 2 words, meaning that those words do not belong in a larger topic. Since this information is redundant in addition to our initial correlations, Appendix B only lists groups of larger than 2 words.

6. Analyzing Whether Responses Answered the Question

One of the practical ways of using our program is to determine whether or not a candidate answered a question. Humans have an obvious bias, but if a computer could be made to do the analysis, then we could control the bias. We implemented a simple method to do this, and got encouraging results.

6.1 Algorithm

In order to find whether the answer answers the question, the first thing we needed to do was identify the "meaningful" words in both the question and the answer. We choose any word that has a correlation to one other word of greater than 0.5 as a meaningful word. We select the meaningful words for both the question and the answer. To discover whether these words match up, we find the correlations from the learning data for every meaningful word in the question compared to every word in the answer. Any word is correlated 1.0 with itself, and if we didn't learn a correlation over 0.2 then we assumed it was 0.0. These correlations were averaged using a mean. We therefore got a value for how well the answer answers the question.

Here is an example question and answer; the important words are highlighted in each:

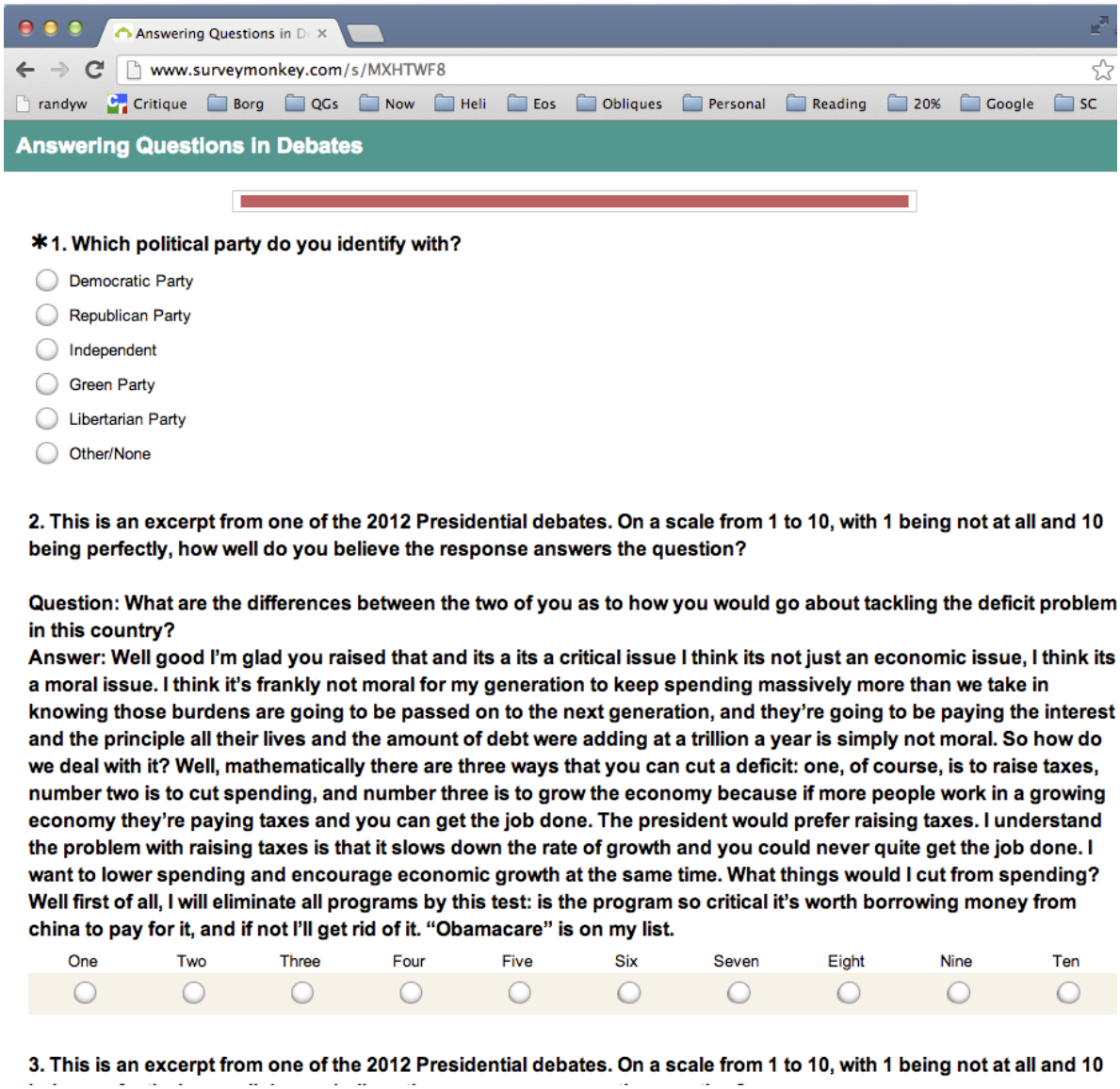
> **Question**: If you were president, and you got a call at 3 am telling you that Pakistan had lost control of its nuclear weapons, at the hands of the Taliban, what would be your first move?
>
> **Answer**: Well obviously, before you ever get to that point you have to build a relationship in that region. That's one of the things that this administration has not done. Yesterday, we found out through Admiral Mullen that Haqqani has been involved with -- and that's the terrorist group directly associated with the Pakistani country. So to have a relationship with India, to make sure that India knows that they are an ally of the United States.
> For instance, when we had the opportunity to sell India the upgraded F-16's, we chose not to do that. We did the same with Taiwan. The point is, our allies need to understand clearly that we are their friends, we will be standing by there with them.
> Today, we don't have those allies in that region that can assist us if that situation that you talked about were to become a reality.
>
> Question meaningful words: [control, nuclear, weapon, hand]
> Answer meaningful words: [build, administration, terrorist, state, instance, sell, today]
> Rating: 0.054268189334015234

One issue that exists with this way of evaluating the answers is outliers. The mean is heavily affected by outliers and if there are a couple important words that aren't completely part of the topic but are mentioned in passing, they will skew the rating we get for the answer.

6.2 Comparing to Human Analysis

In order to test how well our computer program analyzed how the response answers the question, we chose to test it against what people thought. We designed a survey using SurveyMonkey to send out and see on a scale of one to ten, how well the people thought nine questions were answered. (SurveyMonkey limits free users to 10 questions, and we were worried that people wouldn't want to answer lots of questions.) We selected questions from the 2012 Presidential Debates that we thought showed a range of topics as well as some questions that were answered more directly than others (but the survey results were different from what we expected in some cases). The full survey can be found in Appendix C, and the online SurveyMonkey survey can be accessed at http://www.surveymonkey.com/s/MXHTWF8.

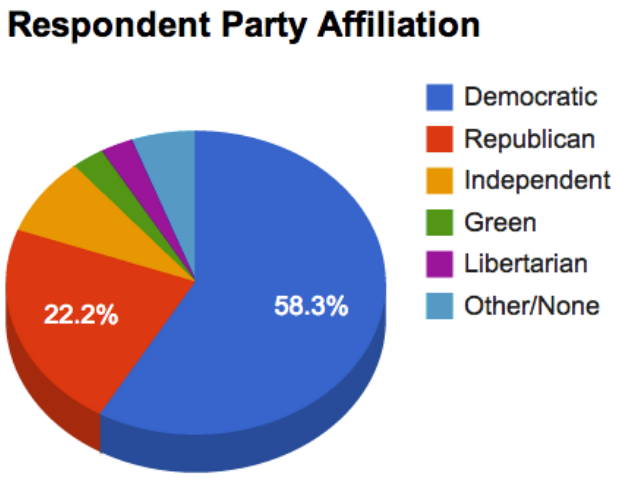
The start of our survey looks like this:



The survey was sent out to approximately 100 people and we got responses from 36. SurveyMonkey collected the results and we were able to analyze the responses compared to our computer program.

As stated in our background information section, we believe that people may be biased towards the political party they represent. Therefore, the first question in the survey asks for which party the person identifies with so we can see whether there is bias in one direction or if the computer can act as a more unbiased guide. The answers are summarized in the figure below. The majority of the respondents were Democratic, which probably reflects the set of people we know to send the

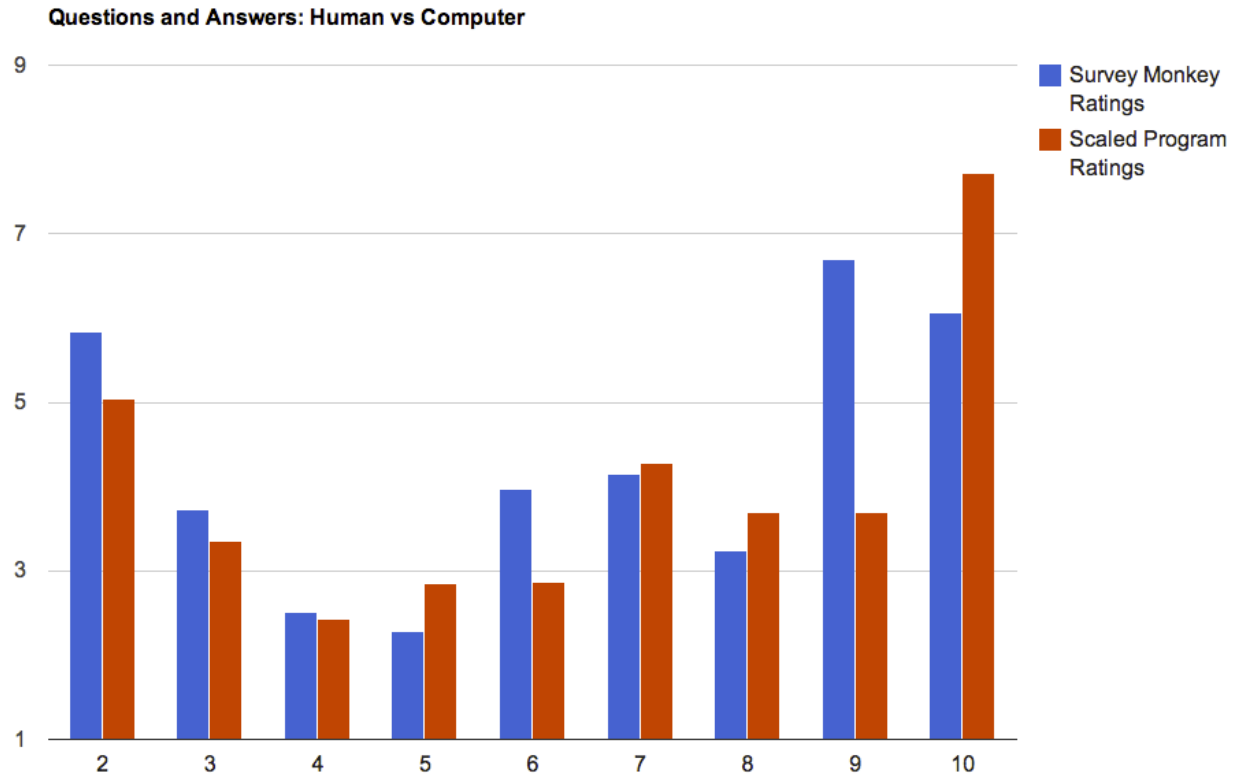
survey to.



We were then able to compare how the computer results changed from question to question to how the survey results changed from question to question. In other words, whether, out of the ten selected questions, the highest rated survey question also got the highest rating in the computer results. The computer results had a much smaller scale than the human results, so we multiplied the computer results by 12, then added 2.2 to make the averages of the two scales the same. The scaling and adding don't change the computed correlation between the human and computer answers.

The following table and figure summarize the results:

Question#	Survey Monkey Ratings	Scaled Program Ratings	Raw Program Ratings
2	5.83	5.044	0.237
3	3.72	3.352	0.096
4	2.51	2.428	0.019
5	2.29	2.848	0.054
6	3.97	2.872	0.056
7	4.15	4.2844	0.1737
8	3.24	3.7	0.125
9	6.69	3.688	0.124
10	6.06	7.72	0.46



Overall, we think our approach does a good job of approximating whether a human would think that the response answers the question. It can be seen from the graph above that the computer does very well on most questions, and comes within 1.0 on seven of the questions. The Pearson's correlation between the human answers and computer answers overall is 0.67, which is good.

Question 9 is an outlier. Without it, the Pearson's correlation is 0.83, which is very good. Humans thought this question was very well answered, whereas the computer did not. Why did this happen?

Here is the computer's debugging output for question 9. We only show the start of the answer; the rest can be seen in Appendix B:

> **Question**: Now let's move to health care where I know there is a clear difference and that has to do with the affordable care act, "obamacare," and its a two minute new segment and that means two minutes each and you go first governor Romney you wanted to repeal the affordable care act, why?
> **Answer**: I sure do. Well in part it comes again from my experience. I was in New Hampshire, a woman came

to me and she said, “Look, I can’t afford insurance for myself or my son. I met a couple in Appleton, Wisconsin, and they said, “We’re thinking of dropping our insurance. We can’t afford it.” And the number of small businesses I’ve gone to that are saying they’re dropping insurance because they can’t afford it.
...
Question meaningful words: [health, care, care, act, “obamacare”, segment, governor, romney, care, act]
Answer meaningful words: [insurance, son, insurance, small, businesse, insurance, cost, health, care, cost, “obamacare”, budget, cost, $2500, year, traditional, insurance, cost, year, down, cost, insurance, $2500, familie, cut, $716, billion, medicare, medicare, senior, small, businesse, “obamacare”, likely, million, rising, unemployment, crisi, table, energy, year, fighting, “obamacare”, fighting, job, killed, job, health, care, state, state, cost, down, $2500, additional, premium]

We think that the reason the computer did so badly on question 9 is because of the many topics included in the answer. The question includes a few off-topic “important words” like “segment,” “governor,” and “romney.” But the reasons that Governor Romney gives for his position have to do with “the big picture,” including medicine but also costs, the budget deficit, the economy, energy, states, and so on. Because of all these other topics that Governor Romney includes, there are lots of topics that don’t “match” the topic of the question in our correlations. As a result, the computer really gets this one wrong.

8. Conclusion and Future Work

In this project, we used statistical analysis to identify and analyze topics in natural language. Using Presidential debates and speeches, we computed correlations to find related words. To handle the volume of training data required, we wrote a parallel program in the form of a Hadoop MapReduce to find the related words, demonstrating that the computation can be linearly sped up with multiprocessing for very large data sets. A straightforward clustering algorithm grouped related words into topics that are surprisingly accurate. Finally, we used the correlations to find meaningful words in questions and answers, whose correlations determine whether a candidate answered a question. The computer’s answers correlated well to the results of a survey of human judgement.

Our solution is only a start on this problem. There are many related questions that we could consider, and things we might want to try next.

We have not sufficiently tested the various tuning parameters of our approach, such as bag size in the proximity approach and the influence of nearby bags. We could also test whether it works better to learn related words by breaking up the input text by paragraph or by sentence.

We could test our code separately with pieces of written work, like speeches, and see the differences compared with pieces of "impromptu" work, like debates. We have learned that in the debates answers are not as well thought through and therefore words are not as carefully chosen as they are in speeches. This could impact or change our results especially with regard to reliability of the learning data as good data. However, this could also have very little impact because the important words for identifying topics are still present.

There are other ways to analyze topics such as looking at transition words and more of the ways that humans identify topic changes. We really wanted to focus our project on actually have the computer learning the way groups of topics appear and change itself. It would be interesting, however, to see if the computer produces better results by learning the topics itself or if a human can tell the computer how to analyze a piece of text with good results as well.

Future work could also include applying the program to figurative pieces of work. For example, we could use this with song lyrics to find songs with similar topics and can be used in things like Pandora. Pandora uses a variety of ways to pick the next song played on a station. One way that Pandora doesn't pick its playlists is by analyzing the lyrics within songs themselves. There could be a way for people to listen to a station based on a particular topic such as "Trees." Something like our program could be used to implement this. Songs could be picked based on high correlations to the chosen topic and what other subjects are included in that. This could be useful for people trying to chose music for a particular subject and could be combined with the typical ways of choosing a playlist such as things like genre.

8. Acknowledgements

The team would like to thank Mrs. Bedeaux for her helpful guidance on English written and

spoken structure. We also thank Dr. Thomas Robey, a Supercomputing Challenge reviewer, for suggesting the use of Spearman rank correlation. We would like to thank Mr. Smith and Randy Wilson for guiding our choice of project and helpful suggestions. We also thank Randy Wilson for teaching us the structure of a MapReduce and paying our Amazon Web Services bill of $26.55.

9. Bibliography

"Amazon Web Services, Cloud Computing: Compute, Storage, Database." Accessed March 27, 2012 at http://aws.amazon.com/.

"Apache Hadoop." Accessed March 28, 2012 at http://hadoop.apache.org/.

Blei, David. "Topic Models." Video lecture from Machine Learning Summer School, September 2009. Accessed January 7, 2013 at http://videolectures.net/mlss09uk_blei_tm/.

Blei, David, and Jordan, Michael. "Variational inference for Dirichlet process mixtures." *Bayesian Analysis* Volume 1, Number 1 (2006), pp. 121-143.

Blei, David, Ng, Andrew, and Jordan, Michael. "Latent Dirichlet allocation". In Lafferty, John. *Journal of Machine Learning Research* **3** (4–5): *pp.* 993–1022, 2003.

"Eclipse - The Eclipse Foundation Open Source Community Website." Accessed March 27, 2012 at http://www.eclipse.org/.

Hofmann, Thomas, "Learning the Similarity of Documents : an information-geometric approach to document retrieval and categorization," *Advances in Neural Information Processing Systems 12*, pp. 914-920, MIT Press, 2000.

"MapReduce." Wikipedia article accessed on March 28, 2012 at http://en.wikipedia.org/wiki/MapReduce.

Muscat, Robert. "Automatic Document Clustering Using Topic Analysis." Computer Science Annual Workshop 2004, pp. 52-59.

Nguyen, Viet-An, Boyd-Graber, Jordan, and Resnik, Philip. "SITS: A Hierarchical Nonparametric Model Using Speaker Identity for Topic Segmentation in Multiparty Conversations." *Association for Computational Linguistics*, 2012.

Nichols, Jeffrey, Mahmud, Jalal, and Drews, Clemens, "Summarizing Sporting Events Using Twitter." 17th International Conference on Intelligent User Interfaces, 2012 ACM 978-1-4503-1048-2. Accessed on December 12, 2012 at http://www.jeffreynichols.com/

papers/summary-iui2012.pdf.

Qiaozhu Mei, Xuehua Shen, Chengxiang Zhai. “Automatic labeling of multinomial topic models.” KDD ‘07, Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 490-499, ACM, NY NY, 2007.

Starnes, Yates, and Moore. "Scatterplots and Correlation." *The Practice of Statistics*. 4th ed. New York: W.H. Freeman, 2010. 150-56. Print.

2012 Presidential Debate at the University of Denver between President Barack Obama and Governor Mitt Romney, moderated by Jim Lehrer. October 3, 2012. Accessed at nytimes.com on November 1, 2012.

Appendix A: Proximity-based Learning Correlations > 0.4

$716 medicare 0.594464515567752
$716 retiree 0.45466289298031565
commercial break 1.0
agreement colombia 0.40680353737793695
air pollution 0.7033910264179355
aircraft carrier 0.829132508427021
airline managed 0.4315277983520347
ambassador youtube 0.5715460291576923
amendment liability 0.5333769280925873
apply card 0.44985389483265115
apply expres 0.41887143773025937
appropriation title 0.5452359935165738
april release 0.5433432497110127
arlen nominee 0.6749562434627363
arlen specter 0.9957620663794557
arlen supported 0.40854857470817985
assad bashar 0.5344184410884772
assad syria 0.434604593026074
auto industry 0.46311945747605665
auto uaw 0.5053263316945532
ayatollah see 0.4165297924597673
ballot language 0.5493891226321143
bank doddfrank 0.4036965858728424
bankruptcy industry 0.4302011357229465
bankruptcy managed 0.7163429210267714
bankruptcy through 0.44407878550392654
bankruptcy uaw 0.5739541627732563
barack obama 0.45401395230309016
bibi netanyahu 0.740300820665865
biden raddatz 0.46117227292574353
biden ryan 0.8087900972820965
bin laden 0.9825529924023757
birth control 0.4477845908978044
birth morningafter 0.5048079824999903
birth pill 0.5220186021053158
black disproportionately 0.40895402114774193
blank check 0.44645365165198947
blank writing 0.4839339338264924
bles god 0.5439027256293146
border secure 0.4535422987958134
branch executive 0.5514956622112166
branch highway 0.4522263073508914
bridge infrastructure 0.4732359575356876

bureaucracy prize 0.4597070788328423
businesse small 0.6519507475874948
cabinet hispanic 0.4348229697033804
cain herman 0.5385325181203288
candy crowley 0.5206227019428403
card expres 0.5268029583201611
care health 0.5078103794880869
carolina south 0.70094232407721
carrier sea 0.4228987088799754
carter jimmy 0.9480468695908524
catholic church 0.6602685507657047
catholic rape 0.680989629443671
cheat vow 0.43196643134166723
chile saving 0.4548156135504716
china tariff 0.41429962480168636
church rape 0.4348951158914722
coal oil 0.43185410610283265
coal production 0.5023504620781358
come illegally 0.4724870692320899
come legally 0.47866625088707904
commercial nasa 0.5959208192659283
commercial space 0.4762355911902087
complete journey 0.7728863218661913
conception life 0.5837604073956864
condition preexisting 0.6949339251220427
conscience religiou 0.507773367468727
conservative rating 0.4006561030741381
consulting lobbying 0.49591336527358953
contraception supreme 0.42507803946371997
control morningafter 0.4587659335043713
control pill 0.5036112815313816
court judge 0.4385457477088248
court supreme 0.5285615845651573
creator endowed 0.4719820217862942
current retiree 0.6171059660063063
customer product 0.41581318737289197
dad mom 0.4969962311556075
deduction middleincome 0.4465746176049027
demonstration happened 0.4723368697725515
dictator minister 0.409049581834181
dictator prime 0.4195176191001859
dividend gain 0.44271802366558727
doddfrank regulation 0.4236195632328255
drilling oil 0.5189590896121132
east middle 0.47221617460047516
employer everify 0.5449233551193988
employer sanctioned 0.6982003656533557
energy solar 0.5734048009973874

english language 0.701748312648381
equal journey 0.48568660707996775
everify sanctioned 0.6250541102117806
fannie freddie 0.5282517189882041
fannie mae 0.7021431534239748
fewer season 0.4038360673229363
fighting season 0.7328016358806204
food stamp 0.7771788337846931
fourth privacy 0.4784965795595932
freddie mac 0.8068362785455369
freddie mae 0.4854687483745465
funding title 0.417894167021299
game saturday 0.40124467443311423
gasoline price 0.4439652702201473
gay hiller 0.6187103469631985
gay orientation 0.5814810020098432
gay samesex 0.4330716741080328
gay voice 0.42304271486566153
governmentrun topdown 0.43961182336420573
governor romney 0.523448033363467
grandiose project 0.4423744308508235
gun hunting 0.5044202943749058
gun owner 0.4828316513170497
happines pursuit 0.40259789150051944
hashtag twitter 0.451445160805272
her named 0.4786350982403044
here illegally 0.5842012767850563
here legally 0.5116469841463162
higherincome lower 0.5312372353745537
hiller voice 0.5200184534085733
hormuz strait 0.9162057947269214
hospital rape 0.5229989174679167
illegal immigration 0.4732999972835906
industry uaw 0.44178866445153425
invented palestinian 0.5116702414405563
iran nuclear 0.5201662855733974
iraq iraqi 0.5328821592213198
islam radical 0.5032266772453493
islamist radical 0.4154998340737519
island puerto 0.6636766906748348
island rico 0.6395399993731343
journey until 0.6807438956866207
kennedy ted 0.6532981631540861
king luther 0.6090116179937284
lady wife 0.40540089059092654
language reinhard 0.5581416561314555
language spanish 0.5638568888944681
liability second 0.41948906703946204

lobby owner 0.4862307726509306
lorraine question 0.4983205744153149
luther martin 0.8359140076371453
mac mae 0.46345616018989394
managed through 0.43511742057044733
managed uaw 0.5993058784653428
manager paint 0.5815205220350446
marriage samesex 0.48217075141611454
medicare retiree 0.4162433696657863
michele newt/romney 0.42675875413114484
mill steel 0.7108524517259686
minimum wage 0.713710064187409
minister prime 0.8672458924093988
moon space 0.446373724879705
morningafter pill 0.5785077182547625
mortgage qualified 0.4353079714808912
nancy pelosi 0.4360902255639095
nasa prize 0.554048775906011
nasa space 0.7114747613448698
nofly zone 0.9130932822538097
nominee specter 0.6472875395346757
nuclear weapon 0.4190705321594113
obama romney 0.4754081231406779
owner pro 0.4916427994898027
pac super 0.7451763434939783
pace voucher 0.4053993061517723
parenthood planned 0.932143074172159
parenthood title 0.41826603909009547
permanent resident 0.5786808138780232
pick robert 0.4355428574149291
planned title 0.42188308864986673
privacy roe 0.4553477908582037
private sector 0.601861654497006
prize space 0.49712756548019
prolife vetoed 0.46169172589737484
puerto rico 0.98328546530836
raddatz ryan 0.5839160118834071
reagan ronald 0.5763441642722894
rose terror 0.496080887475107
security social 0.5464129301036378
senior voucher 0.4262833347323414
shut uaw 0.4394353744020411
specter supported 0.41726422815912795
street wall 0.5834173062057447
subsidie sugar 0.5146265333980821
video youtube 0.40061930293452425

Appendix B: Clustered Topics

Groups with correlation threshold of 0.2:
10: $16, bond, fannie, freddie, investment, looked, mac, mae, mutual, pension,
38: $250000, airline, auto, bail, bankruptcy, busines, businesse, ceiling, code, companie, credit, crushed, cut, debt, deduction, down, exemption, highincome, income, industry, liquidate, loophole, managed, middleincome, million, percent, rate, shut, small, tax, taxe, taxed, taxpayer, through, tough, trillion, uaw, zero,
10: $716, billion, change, current, medicare, pace, retiree, senior, traditional, voucher,
4: 100000, desperately, need, presence,
4: 10th, individual, mandate, unconstitutional,
4: 2010, legislature, prolife, vetoed,
4: 2014, allie, enemie, timeline,
21: [applause], abc, answering, candidate, carolina, charleston, debate, followup, hashtag, king, moderator, podium, port, presidential, rebuttal, santorum, senator, singled, south, tonight, twitter,
3: [commercial, back, break],
15: abortion, accept, adoption, catholic, church, conception, conscience, doctrine, embryo, hospital, life, none, rape, regard, religiou,
43: abstinence, active, alway, amendment, announcer, appoint, appropriation, ban, bill, cheat, commerce, constitution, contraception, court, fourth, funding, gun, however, hunting, judge, justice, legislation, liability, lobby, manufacturer, moine, opposed, owner, parenthood, planned, privacy, pro, roe, second, something, state, stephanopoulos, supreme, title, united, voted, voter, vow,
8: account, chile, increased, saving, security, social, without, younger,
7: actively, cap, look, michele, newt/romney, okay, threw,
4: adding, healthcare, history, inherited,
3: administration, churche, religion,
31: advanced, agency, air, cell, clean, climate, coal, dakota, doesnt, drilling, energy, epa, gallon, gasoline, hampshire, increasingly, keystone, land, license, natural, new, oil, permit, pipeline, pollution, price, production, renewable, solar, water, wind,
4: afghan, afghanistan, haven, pakistan,
7: aircraft, carrier, gulf, navy, sea, ship, smaller,
3: allow, losse, realize,
5: amazing, florida, group, subsidie, sugar,
22: ambassador, attack, benghazi, deny, fewer, fighting, force, iraq, iraqi, letter, month, occurred, season, september, stimulu, summer, surge, terrorist, trained, troop, video, youtube,
13: america, audience, china, computer, currency, honor, latin, putin, restore, rule, tariff, trade, usa,
34: amnesty, apply, arizona, become, card, citizenship, come, country, deport, employer, everify, expres, grandfather, grandmother, here, hire, illegal, illegally, immigrant, immigration, legal, legally, lorraine, magnet, pathway, people, permanent, question, residency, resident, sanctioned, selfdeportation, stay, visa,
5: ann, dad, gift, michigan, mom,
3: april, release, released,
16: arab, assad, bashar, belong, lebanon, nato, neighbor, nofly, opposition, organize, pressure, route, russia, syria, turkey, zone,
3: arabia, qaeda, saudi,
9: arlen, chairman, cover, defended, moderate, nominee, specter, supported, used,
6: asset, castro, cuba, cuban, dictatorship, mile,
30: ayatollah, bibi, credibility, crippling, dictator, difficult, diplomatic, exist, gonna, hama, historically, invented,

iran, israel, israeli, jew, jewish, mind, minister, netanyahu, nuclear, palestinian, prime, rocket, sanction, see, stand, textbook, truth, weapon,
4: balance, balanced, budget, driven,
7: ballot, english, language, learn, reinhard, spanish, teach,
3: bank, doddfrank, regulation,
3: benefit, higherincome, lower,
3: biden, raddatz, ryan,
3: bin, laden, osama,
4: birth, control, morningafter, pill,
7: black, death, disproportionately, drug, minoritie, prison, white,
4: blank, check, countrie, writing,
3: border, mexico, secure,
3: branch, executive, highway,
9: brian, describe, deserve, example, felon, her, named, santorum, vote,
3: bright, settle, track,
7: bureaucracy, commercial, moon, nasa, prize, serie, space,
5: buy, care, health, insurance, provider,
3: cain, identification, patriot,
3: cancer, exact, reward,
3: capital, dividend, gain,
5: carried, carter, electability, jimmy, three,
3: central, chavez, venezuela,
7: ceo, experience, manager, paint, sawyer, someone, talking,
3: citizen, oughta, review,
3: clas, east, middle,
13: college, could, education, failing, graduate, high, janitor, kid, poor, school, student, teacher, work,
5: combat, declared, draft, men, war,
3: commit, fulfill, use,
7: complete, doe, equal, hear, journey, require, until,
3: condition, preexisting, replace,
3: congres, earmark, proces,
4: consulting, contract, lobbying, never,
9: couple, gay, hiller, man, marriage, married, orientation, samesex, voice,
3: course, negotiate, taliban,
3: create, jeremy, job,
4: creator, endowed, happines, pursuit,
6: day, demonstration, happened, rose, terror, tragedy,
4: earn, minimum, pocket, wage,
3: entitie, gingrich, publicly,
3: first, lady, wife,
3: framework, saying, studie,
3: game, saturday, watching,
3: governmentrun, medicine, topdown,
4: governor, massachusett, romney, romneycare,
4: grandiose, majority, project, thought,
3: gregory, perry, uncomfortable,
3: islam, islamist, radical,
3: island, puerto, rico,

3: king, luther, martin,
3: lehrer, obama, romney,
3: mill, plenty, steel,
3: moderator, pick, robert,
3: path, pay, takehome,

Appendix C: SurveyMonkey Questions

1. Which Political Party do you identify with?

2. Question: What are the differences between the two of you as to how you would go about tackling the deficit problem in this country?
Answer: Well good I'm glad you raised that and its a its a critical issue I think its not just an economic issue, I think its a moral issue. I think it's frankly not moral for my generation to keep spending massively more than we take in knowing those burdens are going to be passed on to the next generation, and they're going to be paying the interest and the principle all their lives and the amount of debt were adding at a trillion a year is simply not moral. So how do we deal with it? Well, mathematically there are three ways that you can cut a deficit: one, of course, is to raise taxes, number two is to cut spending, and number three is to grow the economy because if more people work in a growing economy they're paying taxes and you can get the job done. The president would prefer raising taxes. I understand the problem with raising taxes is that it slows down the rate of growth and you could never quite get the job done. I want to lower spending and encourage economic growth at the same time. What things would I cut from spending? Well first of all, I will eliminate all programs by this test: is the program so critical it's worth borrowing money from china to pay for it, and if not I'll get rid of it. "Obamacare" is on my list.

3. Question: This week, the Palestinian Authority brought their bid for statehood to the United Nations. How would you respond to the unilateral declaration of a Palestinian state?
Answer: It starts with an extension of the Reagan philosophy of peace through strength. My philosophy would extend that to peace through strength and clarity. This administration has not made it clear how it stands with Israel. When I was in Israel last month, I met with the deputy prime minister. And he made it shockingly, chillingly clear that, given everything that's going on, number one, Israel will defend itself, with all of the tensions going on in the Middle East.
And he also made it real clear that he wasn't sure how this administration stood when it came to Israel. I made it clear, which -- and I would also make it clear to all of -- our -- the other people in the world, that if you mess with Israel, you're messing with the United States of America. We will stand solidly behind Israel.

4. Question: Help the voters who still have questions about you. What is the biggest misconception about you in the public debate right now?
Answer: I believe that there's a whole question about, what do we need as the person that should be president? What is their position on this issue? Who can be the toughest going after Obama? What we really need, in my opinion, is to say who can lead the country through the kind of fundamental change we have in front of us? And we have people here who all different backgrounds. We've got to restore America's promise in this country where people know that with hard work and education, that they're going to be secure and prosperous and that their kids will have a brighter future than they've had. For that to happen, we're going to have to have dramatic fundamental change in Washington, D.C., we're going to have to create more jobs, have less debt, and shrink the size of the government.

5. Question: If you were president, and you got a call at 3 am telling you that Pakistan had lost control of is nuclear weapons, at the hands of the Taliban, what would be your first move?
Answer: Well obviously, before you ever get to that point you have to build a relationship in that region. That's one of the things that this administration has not done. Yesterday, we found out through Admiral Mullen that Haqqani has been involved with -- and that's the terrorist group directly associated with the Pakistani country. So to have a relationship with India, to make sure that India knows that they are an ally of the United States.

For instance, when we had the opportunity to sell India the upgraded F-16's, we chose not to do that. We did the same with Taiwan. The point is, our allies need to understand clearly that we are their friends, we will be standing by there with them.
Today, we don't have those allies in that region that can assist us if that situation that you talked about were to become a reality.

6. Question: Struggling U.S. workers continue to compete with millions of illegal aliens. Do you support legislation to require all employers to use E-Verify in order to ensure that the people that they hire are actually legally authorized to work in the U.S.? And will you impose penalties against employers who continue to hire illegal workers? The question is, should employers be required to use E-Verify?
Answer: Well, let me say, first of all, I think we would be better off to outsource E-Verify to American Express, MasterCard or Visa, because they actually know how to run a program like that without massive fraud.
Second, the program should be as easy as swiping your credit card when you buy gasoline. And so I would ask of employers, what is it you would object to in helping the United States of America in dealing with the problem involving illegal immigration?
But, in addition, I want to reinforce what Congresswoman Bachmann said. I strongly favor 100 percent control of the border, and I strongly favor English as the official language of government.
And I favor modernizing the legal visa system to make it far more convenient, far easier and far more practical. Here in Orlando, where we have a huge interest in people being able to visit easily for tourism, we have a terribly antiquated legal system while our border is too open for people who are illegal.

7. Question: We're looking at a situation where 40 percent of the unemployed have been unemployed for six months or more. What about those long- term unemployed who need a job right now?
Answer: Well, what you're seeing in this country is 23 million people struggling to find a job, and a lot of them, as you say, Candy, have been out of work for a long, long, long, long time.

The president's policies have been exercised over the last four years, and they haven't put Americans back to work. We have fewer people working today than we had when the president took office. If the — the unemployment rate was 7.8 percent when he took office. It's 7.8 percent now. But if you calculated that unemployment rate taking back the people who dropped out of the workforce, it would be 10.7 percent. We have not made the progress we need to make to put people back to work.

That's why I put out a five-point plan that gets America 12 million new jobs in four years and rising take-home pay. It's going to help Jeremy get a job when he comes a out of school. It's going to help people across the country that are unemployed right now.

And one thing that the — the president said which I want to make sure that we understand — he — he said that I said we should take Detroit bankrupt, and — and that's right. My plan was to have the company go through bankruptcy like 7-Eleven did and Macy's and — and — and Continental Airlines and come out stronger. And — and I know he keeps saying, you wanted to take Detroit bankrupt. Well, the president took Detroit bankrupt. You took General Motors bankrupt. You took Chrysler bankrupt. So when you say that I wanted to take the auto industry bankrupt, you actually did. And — and I think it's important to know that that was a process that was necessary to get those companies back on their feet, so they could start hiring more people. That was precisely what I recommend and ultimately what happened.

8. Question: What if — what if the prime minister of Israel called you on the phone and said: Our bombers are on the way. We're going to bomb Iran. What do you say?

Answer: Bob, let's not go into hypotheticals of that nature. Our relationship with Israel, my relationship with the prime minister of Israel is such that we would not get a call saying our bombers are on the way or their fighters are on the way. This is the kind of thing that would have been discussed and thoroughly evaluated well before that kind of action.
But let me — let me — let me come back — let's come back — let's come back and go back to what the president was speaking about, which is what's happening in the world and — and — and the president's statement that things are going so well.
Look, I — I look at what's happening around the world and I see Iran four years closer to a bomb. I see the Middle East with a rising tide of violence, chaos, tumult. I see jihadists continuing to spread. Whether they're rising or just about the same level hard to — hard to precisely measure, but it's clear they're there. They're very, very strong.
I see Syria with 30,000 civilians dead, Assad still in power. I see our trade deficit with China larger than it's — growing larger every year as a matter of fact. I look around the world and I don't feel that — you see North Korea continuing to export their nuclear technology.
Russia's said they're not going to follow Nunn-Lugar anymore; they're (back ?) away from their nuclear proliferation treaty that we had with them. I look around the world, I don't see our influence growing around the world. I see our influence receding, in part because of the failure of the president to deal with our economic challenges at home, in part because of our withdrawal from our commitment to our military and the way I think it ought to be, in part because of the — the — the turmoil with Israel. I mean, the president received a letter from 38 Democrat senators saying the tensions with Israel were a real problem.

9. Question: Now let's move to health care where I know there is a clear difference and that has to do with the affordable care act, "obamacare," and its a two minute new segment and that means two minutes each and you go first governor Romney you wanted to repeal the affordable care act, why?
Answer: I sure do. Well in part it comes again from my experience. I was in New Hampshire, a woman came to me and she said, "Look, I can't afford insurance for myself or my son. I met a couple in Appleton, Wisconsin, and they said, "We're thinking of dropping our insurance. We can't afford it. And the number of small businesses I've gone to that are saying they're dropping insurance because they can't afford it. The cost of health care is just prohibitive and and we've got to deal with cost and unfortunately when when you look at "obamacare," the congressional budget office has said it will cost $2500 a year more than traditional insurance so its adding to cost. And as a matter of fact when the president ran for office he said that by this year he would have brought down the cost of insurance for each family by $2500 a family. Instead it's gone up by that amount so it's expensive things that hurt families so thats one reason I don't want it. Second reason: it cuts $716 billion from medicare to pay for it. I want to put that money back in medicare for our seniors. Number three: it puts in place an unelected board thats going to tell people ultimately what kind of treatments they can have. I don't like that idea. Fourth: there was a survey done of small businesses across the country. It said what's been the effect of "obamacare" on your hiring plans and threequarters of them said it makes us less likely to hire people. I just don't know how the president could have come into office facing 23 million people out of work, rising unemployment, an economic crisis at the at the kitchen table, and spent his energy and passion for two years fighting for "obamacare" instead of fighting for jobs for the american people. It has killed jobs and the best course for health care is to do what we did in my state: craft a plan at the state level that fits the needs of the state and then lets focus on getting the costs down for people rather than raising it with the $2500 additional premium.

10. Question: Do you see a major difference between the two of you on social security?
Answer: You know I suspect that on social security we've got a somewhat similar position. Social security is structurally sound. It's going to have to be tweaked the way it was by Ronald Reagan and democratic speaker, Oneill, but it is the basic structure is sound. But I want to talk about the values behind social security and medicare and then talk about medicare because thats the big driver.

Appendix D: Java Code Listing

Our final single-computer Java code as it was at the time of filing this final report is here: http://goo.gl/7GwSz .

If we make changes to this code after filing this report, but before final judging, the code will be in the other document linked from inside the above document.

Appendix E: Hadoop Java Code Listing

```
/**
 * This is a Hadoop map reduce to learn word correlations.
 * The mapper finds pairs of words in each bag,
 * and sends word counts (keyed by word pair) to the reducers.  The reducer
 * takes all the word counts for a pair of words
 * and computes the correlation between those words; if the words are
 * correlated, it writes it to the output.
 *
 * @authors Eli, Justin, Ari
 *
 */

package org.lacueva.jeasc;

import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;
import java.util.*;

// This is for Spearman's correlation.
//import org.apache.commons.math3.stat.correlation.*;

import org.apache.hadoop.fs.FSDataInputStream;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

public class LearnCorrelations {

	/**
	 * This is the Mapper for a Hadoop map reduce. It takes one bag, splits it
	 * up into word pairs, counts the instances of the words in the bag, and
	 * stores all of this info in a String. It then sends the Strings out to the
	 * Reducers.
	 */
	public static class BagMapper extends
			Mapper<LongWritable, Text, Text, Text> {
		private ArrayList<String> allWords = null;

		// this is called once when the mapper starts up. It reads the list of all
	words from our Amazon bucket
		public void setup(Context context) throws IOException {
			String allWordsFile = "s3n://jeasc/allinput/allWords.txt";
			URI uri = null;
```

```
		try {
			uri = new URI(allWordsFile);
		} catch (URISyntaxException e) {
			System.out.println("Failed to get URI for allWords.txt");
			return;
		}
		FileSystem fileSystem = FileSystem.get(uri,
context.getConfiguration());
		FSDataInputStream stream = fileSystem.open(new Path(allWordsFile));
		Scanner scanner = new Scanner(stream);
		allWords = new ArrayList<String>();
		while (scanner.hasNext())
		{
			allWords.add(scanner.next());
		}
		System.out.println("Finished setup, allWords.txt has word count: " +
allWords.size());
	}

	// this is called for every line in the input, ie for each bag.
	public void map(LongWritable key, Text value, Context context)
			throws IOException, InterruptedException {
		Text outputKey = new Text();
		Text outputValue = new Text();
		String[] words = value.toString().split(" ");
		HashMap<String, Double> wordCounts = new HashMap<String, Double>();
		for (String wordColonCount : words) {
			String[] wordAndCount = wordColonCount.split(":");
			double count = Double.valueOf(wordAndCount[1]);
			wordCounts.put(wordAndCount[0], count);
		}

		for (int i = 0; i < allWords.size() - 1; i++) {
			String word1 = allWords.get(i);
			double count1 = 0.0;
			if (wordCounts.containsKey(word1))
				count1 = wordCounts.get(word1);
			for (int j = i + 1; j < allWords.size(); j++) {
				String word2 = allWords.get(j);
				double count2 = 0.0;
				if (wordCounts.containsKey(word2))
					count2 = wordCounts.get(word2);
				if (count1 != 0.0 || count2 != 0.0) {
					String pair = word1 + "," + word2;
					String counts = count1 + " " + count2;
					outputKey.set(pair);
					outputValue.set(counts);
					context.write(outputKey, outputValue);
				}

			}
		}

	}
```

```
}

/**
 * This is the Reducer for a Hadoop map reduce. It takes a word1,word2 pair
 * as the key, and all the counts where those two words appear in bags. It
 * computes the correlation between the two words. If the correlation is
 * high enough, it writes the pair of words and the correlation.
 */
public static class CorrelationReducer extends
              Reducer<Text, Text, Text, Text> {
       private Text outValue = new Text();

       public void reduce(Text key, Iterable<Text> values,
                     Context context) throws IOException, InterruptedException {
              double sum1 = 0;
              double sum2 = 0;
              ArrayList<Double> counts1 = new ArrayList<Double>();
              ArrayList<Double> counts2 = new ArrayList<Double>();
              counts1.add(0.0);
              counts2.add(0.0);
              for (Text numsText : values) {
                     try {
                            String[] numStrings = numsText.toString().split(" ");
                            double count1 = Double.parseDouble(numStrings[0]);
                            double count2 = Double.parseDouble(numStrings[1]);
                            sum1 += count1;
                            sum2 += count2;
                            counts1.add(count1);
                            counts2.add(count2);
                     } catch (Exception e) {
                            System.out.println("Caught exception " + e + ": " +
key.toString() + " = \"" + numsText.toString() + "\"");
                     }
              }
              if (sum1 >= 14.0 && sum2 >= 14.0) {
                     Double[] wordOneCounts = counts1.toArray( new
Double[counts1.size()] );
                     Double[] wordTwoCounts = counts2.toArray( new
Double[counts2.size()] );
                     double correlation = getPearsonCorrelation(wordOneCounts,
wordTwoCounts);
                     if (correlation > 0.2) {
                            String value = ": " + correlation;
                            outValue.set(value);
                            context.write(key, outValue);
                     }
              }
       }

       // computes the Pearson Correlation between two arrays of doubles
       public static double getPearsonCorrelation(Double[] wordOneCounts,
                     Double[] wordTwoCounts) {
              double result = 0;
              double sum_sq_x = 0;
```

```
                double sum_sq_y = 0;
                double sum_coproduct = 0;
                double mean_x = wordOneCounts[0];
                double mean_y = wordTwoCounts[0];
                for (int i = 2; i < wordOneCounts.length + 1; i += 1) {
                        double sweep = Double.valueOf(i - 1) / i;
                        double delta_x = wordOneCounts[i - 1] - mean_x;
                        double delta_y = wordTwoCounts[i - 1] - mean_y;
                        sum_sq_x += delta_x * delta_x * sweep;
                        sum_sq_y += delta_y * delta_y * sweep;
                        sum_coproduct += delta_x * delta_y * sweep;
                        mean_x += delta_x / i;
                        mean_y += delta_y / i;
                }
                double pop_sd_x = (double) Math.sqrt(sum_sq_x
                                / wordOneCounts.length);
                double pop_sd_y = (double) Math.sqrt(sum_sq_y
                                / wordOneCounts.length);
                double cov_x_y = sum_coproduct / wordOneCounts.length;
                result = cov_x_y / (pop_sd_x * pop_sd_y);
                return result;
        }
    }

    public static void main(String[] args) throws Exception {
        Job job = new Job(new Configuration(), "learncorrelations");
        job.setJarByClass(LearnCorrelations.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);

        job.setMapperClass(BagMapper.class);
        job.setReducerClass(CorrelationReducer.class);

        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);

        FileInputFormat.addInputPath(job, new Path("s3n://jeasc/" + args[0]));
        FileOutputFormat.setOutputPath(job, new Path("s3n://jeasc/" + args[1]));

        job.waitForCompletion(true);
    }

}
```