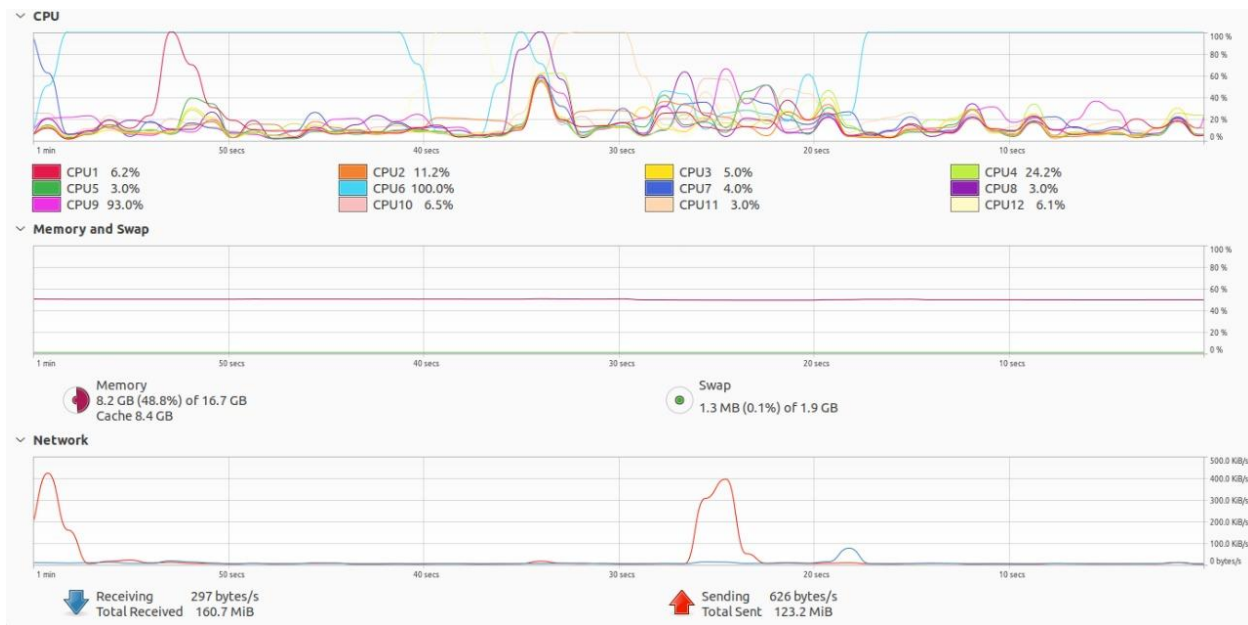


University of Science and Technology at Zewail  
City

Communications and Information Engineering  
Program

CIE 427: Big Data Analytics

Mini-Project 1 Report



**Figure:** Moumen's laptop crying over a whole dataset Mapreduce. Fast forward a few minutes later, his OS runs out of space, Colab fails us, and Alaa's laptop runs out of space.

Alaa Alajmy 201700095

Moemen Gaafar 201700873

## Table of Contents

- I. Introduction
- II. Challenges
- III. Optimizations
- IV. Code Pipeline
- V. Data Analysis

### I. Introduction

Quite a few of this mini-project's crises were solved (or made worse) on Reddit's developer subcommunities. In this project, we analyze Reddit comments from January, 2015. Through a sequence of three Mapreduce jobs we find:

- the most popular subreddits and the topics they discuss
- the most active redditors and what they like to say
- January, 2015's most popular reddit topics
- its most positively-mentioned topics and its most negatively-mentioned ones
- its numerous controversiality calculation and data archiving failures

### II. Challenges

Through the process of making this project work we were faced by several hindrances including:

- **No Controversiality:** While trying to meet the second analysis requirement (finding how comment controversiality affects its reply rate), we found that all controversialities were set to 0.
- **No Downvotes:** While working on the third requirement, we found that all downvote counts were set to zero.
- **Negative Upvotes:** We additionally found that some comments had negative upvote counts, always equal to their scores (the value of

downvotes subtracted from upvotes. We concluded then that the dataset reported neither upvote nor downvote counts but rather the score of each comment and that this value was written as both the “score” and “ups” value. We couldn’t however confirm this conclusion, so we disregarded all comments with negative upvote values and treated downvote values as zeros for all comments.

- **An Ubiquitous AutoModerator:** After completing our analysis, we found that the Reddit AutoModerator, a bot that “automoderates” all subreddits was the most active redditor. Since the AutoModerator had the largest number of comments, its comments always were the most upvoted and earned the highest cumulative positivity/negativity scores. The AutoModerator’s topics were thus all over our results. To alleviate this problem and find the actual topics redditors most often interacted with or had strong feelings about, we added the word ‘automatically’ to our list of stopwords. We found that unlike “question” or “contact”, which were also very common among the AutoModerator’s comments, “automatically” was not a very popular word among human redditors.
- **Non-English Comments:** We found that several non-English words often distorted our results, so we thought to use a Python library that removes words not in the English dictionary to remove them. The results of this attempt missed names of humans and places, so instead we opted to only remove words that did not have an English first letter. This meant we had to tolerate having non-english words from languages such as French and German which also used English letters.
- **OS Space:** We initially attempted to run our whole dataset MapReduce jobs on Google Colab to avoid space issues, however we found that Colab keeps on failing when run for long durations with an imperfect internet connection. As we ran the MapReduce jobs on our laptops, we found that even when directed to save the output files elsewhere, Hadoop temporarily renders its intermediate outputs in the OS’s file system. Additionally, Hadoop is only permitted to take up only a pre-specified percentage of whichever directory it is rendering data on. After running for several hours, our file systems ran out of space, so we tweaked the

MapReduce configurations to have Hadoop render in our Laptop's external storage drives.

- **Sums vs. Averages:** When trying to calculate the most upvoted, positively-mentioned, and negatively-mentioned topics, we found that choosing the topics with the highest sum of these criteria meant that topics most discussed always had a great advantage over less discussed ones, since they simply were present in more comments. On the other hand, we found that using averages of these qualities rather than their sums mean that topics discussed by specially popular comments were given an unfair advantage and these results were unrepresentative of the values of a topic but rather of specific popular comments. At the end we chose to use sums tolerating their lesser severe disadvantage.

### III. Optimizations

- **Sequential Mapreduce Jobs:** To avoid repetitive mapping that would waste time and resources, we chose to meet all of the project's requirements via three Mapreduce jobs running one after the other. More on this will be discussed in the next section.
- **Compression:** We ran the first Mapreduce job, which takes in the raw data, on the bzip2 compressed file. We tried to have Hadoop return compressed outputs too, but unfortunately, Hadoop streaming does not have that feature.
- **Small Outputs:** To reduce computational complexity and space requirements, we opted to avoid sorting all of the data and chose instead to return only the top 10 best-performers of each question.
- **Extracting Topics:** To extract the topics of comments, we had the option to use algorithms such as LDA implemented by Python libraries. We found these implementations would require so much time especially when run on such a huge dataset. We instead opted to use simple bigrams or trigrams. We found both methods gave equal topic clarity but chose to use bigrams to reduce the size of our intermediate data. Lamentably, our resulting comment topics were still not very clear.

## IV. Code Pipeline

Our project was run via a series of three MapReduce Jobs, each using the output of the previous job as its input. We additionally used flags (lowercase letters) at the start of each printed line to help future functions identify what tasks each line belongs to. All combiners we used were identity combiners.

### 1. MapReduce 1:

The first Mapper processed each comment. It cleans the data by removing stop words, websites, and words with a non-english first letter. It returns more than one topic for each comment, discarding all comments with no topics or with negative upvote counts. The mapper then prints for each comment:

- Subreddit name + 1 (count of comments on this subreddit) as key with no value
- Author username + 1 (count of comments by this author) as key with no valueTopic + subreddit name as key with no value (one for each topic)
- Topic + author name as key with no value (one for each topic)
- Comment id as key + controversy
- Comment parent id as key + 1 (count of replies on this parent comment, here comments whose parent is not also a comment are discarded)
- Topic as key + upvotes, downvotes, positivity score, and negativity score (for each topic of each comment)

The first reducer the uses this data to output:

- Subreddit name as key + count of comments on it
- Subreddit name as key + topic + count of times this topic was discussed on this subreddit (for each topic)
- The same above two lines for each author
- Controversiality of each comment (as key) + number of replies it received

- Topic as key + count of times it was discussed + sum of upvotes it received, sum of downvotes, sum of positivity scores, and some of negativity scores

## 2. MapReduce 2:

Our second Map was an identity mapper. Meanwhile, the reducer printed:

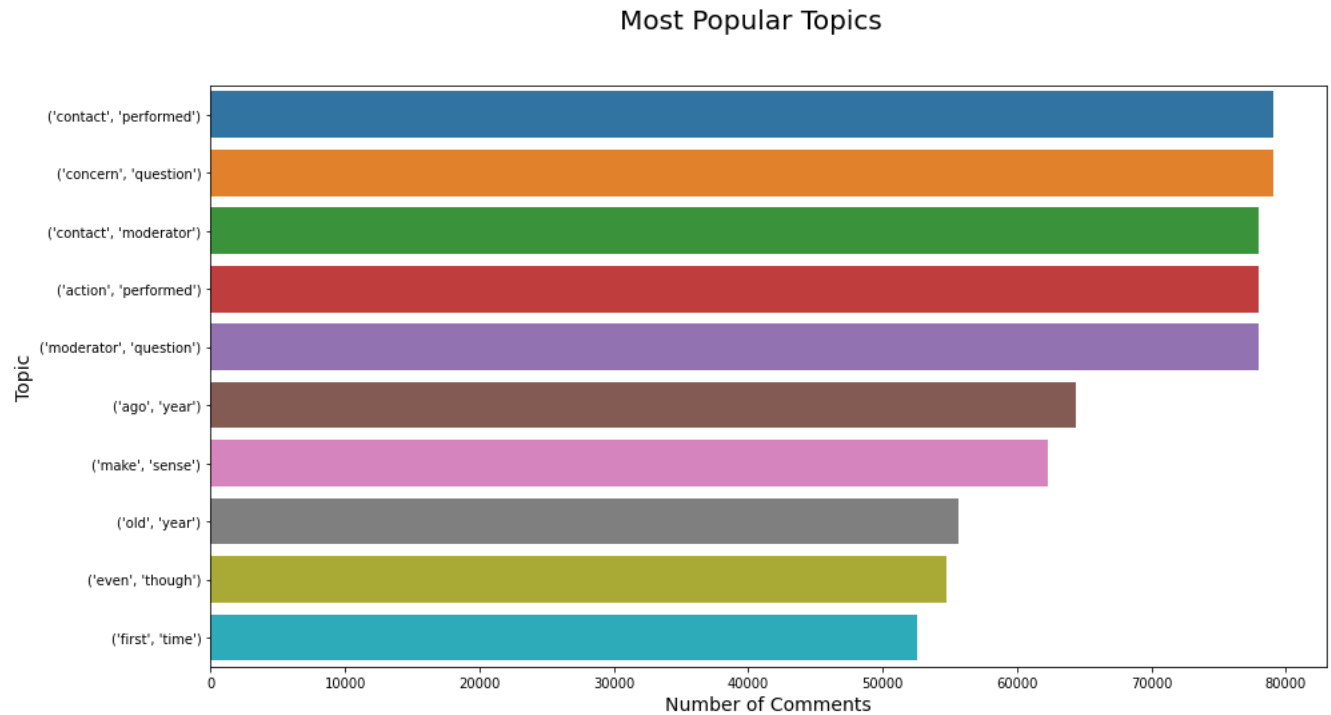
- the count of comments on each subreddit and its three most discussed topics
- the count of comments by each author and his/her/its three most discussed topics
- Each unique comment controversiality score, the number of comments that recieved it, and the sum of replies on them (this was just one line for controversiality = 0)
- The ten most discussed topics and how many times they were discussed
- The ten most upvoted topics and how many upvotes they received
- The ten most downvoted topics and how many downvotes they received (here we found that all received zeros)
- The ten most positively discussed topics
- The ten most negatively discussed topics

## 3. MapReduce 3:

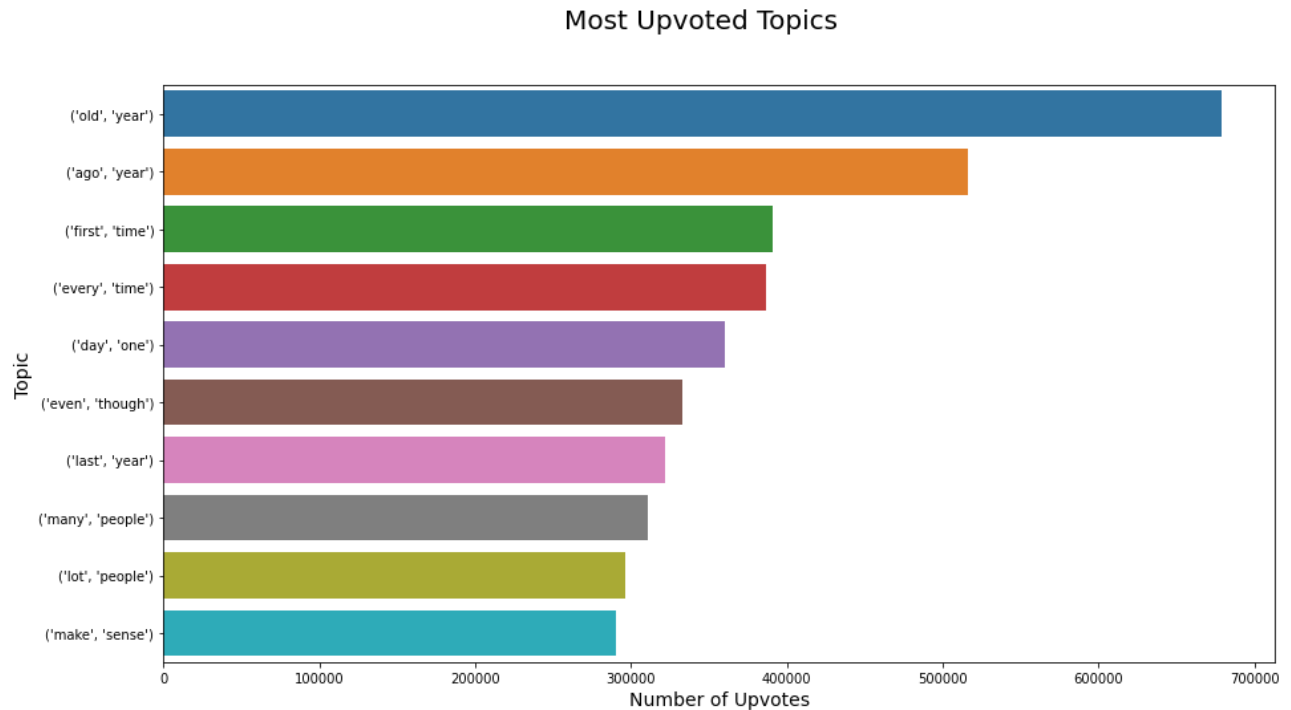
Here again our Mapper was an identity function. This final MapReduce had the sole job of returning the 10 most popular subreddits and the 10 most active authors.

## V. Data Analysis

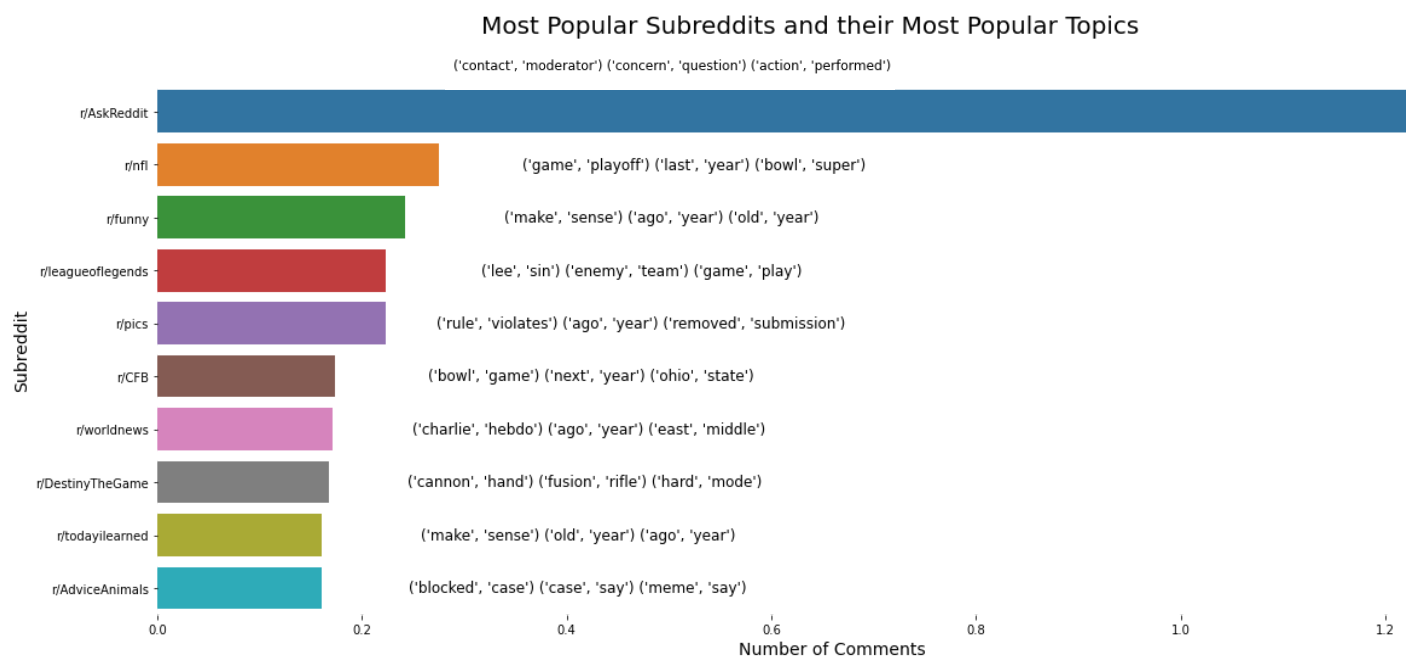
Please behold our insights:



We see that the 5 most popular topics were made so by the ubiquitous comments of the AutoModerator and people's replies to them. Additionally, since the set is of comments made at the start of 2015, people were talking about the past year. Redditors also have an apparent liking to discuss the first time they did things :)



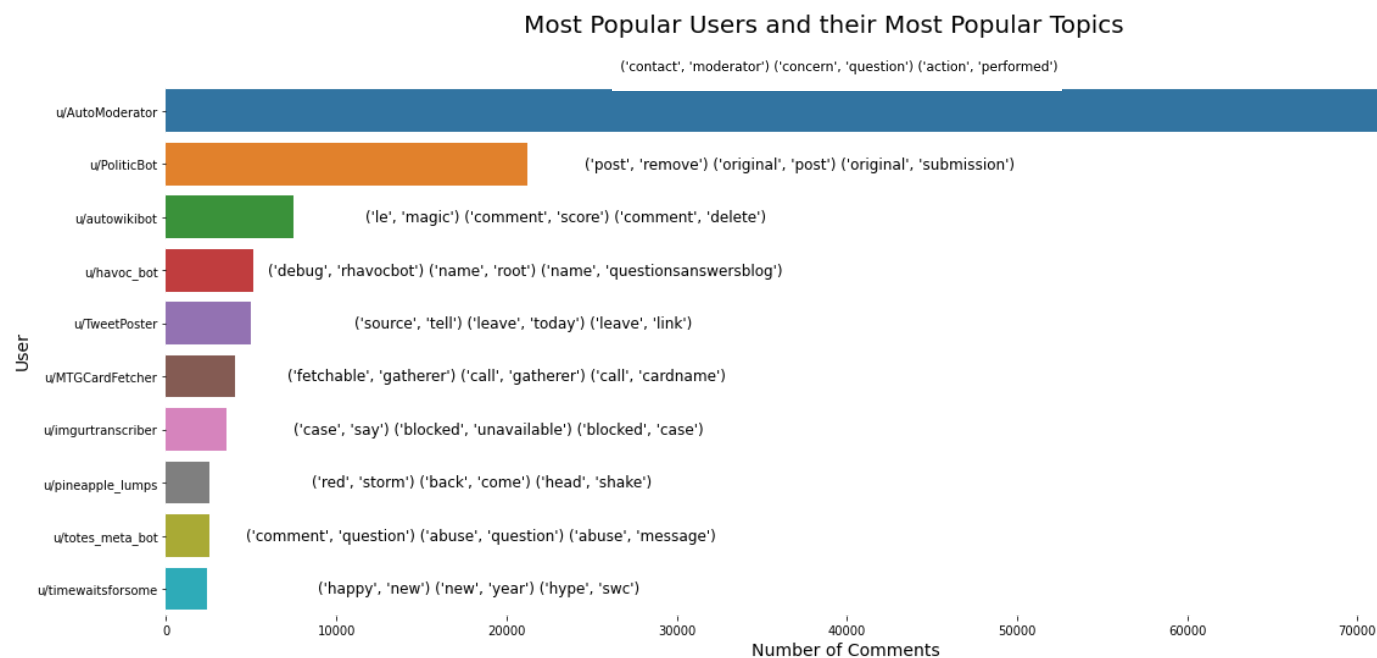
Here we note the parallelism between the most popular topics and the most upvoted ones. This is a result of the “Sums vs Averages” challenge discussed above.



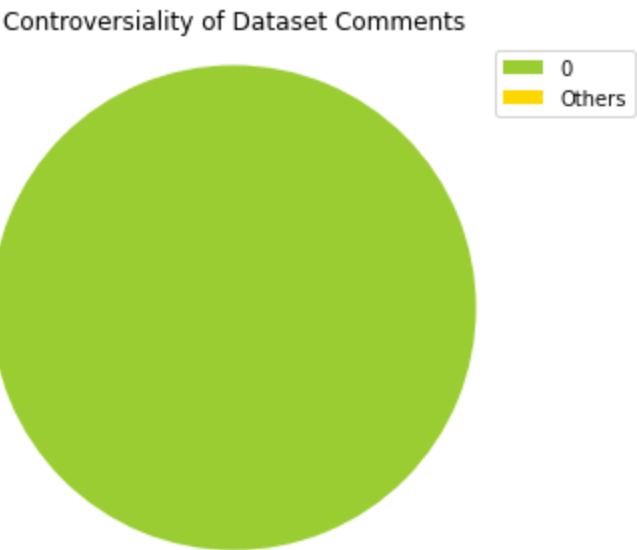
So the top winner of most popular subreddit is r/AskReddit, where the most popular topics also seem to involve the AutoModerator. We also note the CFB



subreddit was profusely discussing Ohio State whose football team had just won the 2014 College Football Playoff National Championship Game.

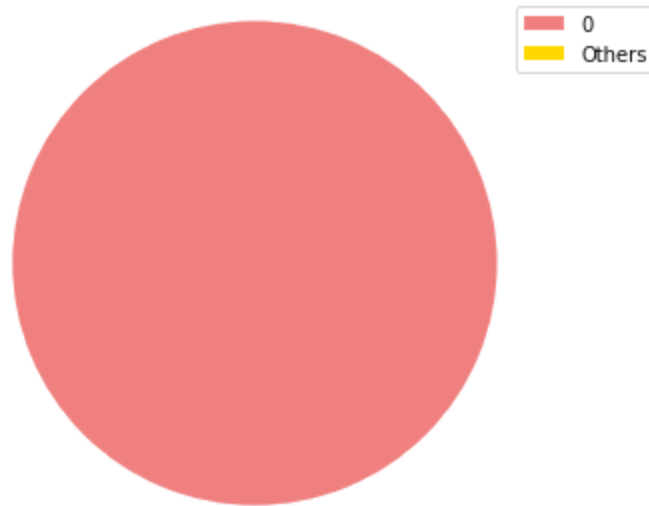


The most active user is the AutoModerator with the most discussed topics as the r/AskReddit subreddit. Additionally all four most active accounts are bots.





Here we note that all comments had 0 controversiality scores.

Downvotes on Dataset Comments



Similarly, all comments had zero reported downvotes.

Finally, these are the most positively-discussed and most negatively discussed topics:

<div>1) ('new', 'year')</div> <div>2) ('much', 'thank')</div> <div>3) ('action', 'performed')</div> <div>4) ('concern', 'question')</div> <div>5) ('contact', 'performed')</div> <div>6) ('contact', 'moderator')</div> <div>7) ('moderator', 'question')</div> <div>8) ('better', 'much')</div> <div>9) ('happy', 'new')</div> <div>10) ('make', 'sense')</div>		<div>1) ('holy', 'shit')</div> <div>2) ('even', 'though')</div> <div>3) ('ago', 'year')</div> <div>4) ('make', 'sense')</div> <div>5) ('old', 'year')</div> <div>6) ('every', 'time')</div> <div>7) ('many', 'people')</div> <div>8) ('lot', 'people')</div> <div>9) ('first', 'time')</div> <div>10) ('people', 'think')</div>	
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------