Christopher Mersman
Homework 3 Written Component

Q1. Consider the case where your will be extending your system so that you can perform sentiment analysis in German. If you have access to a dictionary that translates words (or a set of words) between the English & German how would you go about implementing this capability? [300-400 words]

1) Not much of my implementation would have to change. I would use the same styles that I included in my original but adjust the order of my operations. I would have to store the dictionary that converts the English to German as well as German to English and find the place where I would have to change the words. I would need to create a function that could change the sentences back and forth between the words. A question that would be important to ask is if I would have German sentiment dictionaries or English sentiment dictionaries. There could be an implementation for both of them.

In my implementation I have saved state within my mapper code where I import my dictionaries once so I don't have to do network and file i/o every time I need to use the dictionary. This saves me a lot of time so the extra computation of doing these conversions will not take up so much i/o.

So now depending on if the sentiment dictionaries are in German or English comes in. Really they would be the same implementation just switching roles. I am going to make the assumption of a English sentiment dictionary, German twitter data, and a map that takes words and puts them from English to German and vice versa. You may not need the other way around where you go from German to English unless you are going to want to print the values out back in English which wouldn't make much sense.

From there we will need to find the correct placement for the conversions to happen. In the mapper we will to parse tweet data into a German line. This line may hold some unicode values that would need to be converted at some time but there is probably a way that the Json parse could handle that. Once that is read in you would take each word from that sentence, convert it to English using the fast HashMap and then see which dictionary its in and assign the German word sentiment. You then can send the word onward to the Reducer to have the sentiment compiled more.

Q2. Company AdPlacer is in the business of placing advertisements for products. They are exploring the use of sentiment analysis of web pages to place these advertisements. Describe how you would extend your system to meet the company's requirements.[400-500 words]

2) For the start of this problem I would need to know if I would have to create my own dictionary files. There are probably many different sentiment dictionary files online that could be used however, if I was to create my own I would probably manually go through some dictionary

and assign my person sentiment choice to that word. I could also just sort a large file such as a book and manually move the words by a sentiment of my choice to where I think they would belong. This would take a lot of time but if I was to get paid for it why not?

From there I would want to know what kinds of advertisements. That way I could use some word association to find individual words and subjects that would relative to the advertising that would be requested. After getting the correct sites and locations I would start to scrape words that would be mapped to specific addresses. I would do large strings of words that would found at the websites.

Getting these words would be pretty easy. Using any ‘curl’ or ‘wget’ command could satisfy this request. I would then put those words of different websites into large dictionary files for those specific addresses. I would adjust the way that I would parse the file to allow for this to be easily done such as maybe putting an address as the key for the sentiment. This could strain the Mapper class however. If the Mapper class has too much computation there wouldn’t be any point of doing a mapreduce system. However, the computation into pairing up words is very helpful in the implementation of mapreduce.

You also may want to complete different sentiment for specific addresses at different times. I was originally thinking of running the sentiment analysis on multiple addresses at the same time. I now understand that this could cause a problem in the long run because of trying to compare. You would almost HAVE to run it on individual sites separately to get a ratting on how important and good that site would be for your advertising.

Also a way to make this more efficient would be to use affinity scores instead of sentiment analysis. Or maybe even both. You would want to see if the site of which you are going to want to advertise on is going to have similar content or have a correct target audience and the language on the site would help define that attribute.

Q3. JSON Tweet data contains many fields, including geo-location data identifying the user's location at the time of the tweet. How would you need to modify your code to track this information? [400-500 words]

3) The question I would ask here would be, “What do you want to measure?” Do we want to show that people closer to the ocean show a better sentiment to the word ‘ocean’? Do we want to see there is a total less sentiment in larger cities that could maybe measure the depression of the city? Do we want to measure the location on the fly and output it with the key of a mapper and reducer so that the same word in New York isn’t the same as the word in Los Angeles? These questions can all be answered a with a little different implementation.

When it comes to the idea of measuring words from specific cities it wouldn’t be too hard to ignore all cases where the tweet contains location that are not the same as the one we want to study. What would be a nice part of implementing this is that unlike normal sentences, the GPS will have consistent format for the city and location. When implementing the cases for the words, if you want to count the sentiment of ‘bunny’ then you have to take into account the words ‘bunnies’, ‘bunnys’, ‘bunys’, etc. If those words were to be nouns you would want to map them to the main specific word because that is really what the meaning of the word would be and to get

the most accurate sentiment, those would have to be included. But like I was saying about the GPS locations, we do not have to handle all these cases. Taking only data from that city we can calculate the sentiment of words for that city. We could then compare between cities to see which city likes dogs or cats or any noun more.

For the situation on doing multiple city ratings of words at the same time there could be a way to parse the information and get the location of the tweet and append the location to each word. This would change the key so that the word wouldn't be considered the same word. For example: 'dog San Francisco' would have a different sentiment as 'dog Denver'. And for the case that didn't have a location, we would ignore that because for the idea of comparing geo-locations we would need there to be a defined place for the sentiment to come from. Before things are sent to the Reducer you would have output such as:

'dog San Francisco' '3' '1'
'dog Denver' '3' '1'
'dog Orlando' '3' '1'

This would just be the example for dog output; of course there would also be output for other words from different cities. The second number would be the sentiment that was found within the word and the last number would be the word count. The importance of the word count would be to show the 'popularity'. This would allow you to calculate the sentiment for words for specific cities at the same time.

Q4. How might AdPlacer leverage this geolocation information from Q3 to target advertisements? [300-400 words]

4) This would be very important to AdPlacer because location can be a large key in the marketing world and for figuring out where your target market is. Once a lot of data is collected we can do that calculation of analysing multiple words for multiple towns. So if there is someone who needs a place to put their ad and they are selling dog food. So they are going to look for some area that has a very positive sentiment on dog. I was previously talking about the way that popularity is calculated. From here we would need to have a sentiment divided by the amount of times the word appears. Now there could be a bit of a problem with this in the fact that there may only be one person in a town that is tweeting about something. In these cases the sentiment over occurrence score would throw things off. We would have to make an occurrence threshold that doesn't allow for it to work if the amounts are so little. This would be difficult to implement within this system. Another part of this that would throw things off would be the case where there is only one person that tweets every second about how awesome dogs are. This would give the image the town would be in love with dogs but really it is just one teenaged girl who happens to run into dogs a lot.

Despite the disadvantages that I previously mentioned, this would actually be pretty helpful in finding a good target market city. If could also work with marketing that is already out there and too see if it is working. Analysing the use of your specific company name as the noun and recording the sentiment in different locations. This all could be a really good way to gage the view of a company.

Q5. How would you change your design to support real-time sentiment analysis of streaming data? [400-500 words]

5) This would be probably the most difficult thing to implement. In this sort of system when dealing with real-time we need to take into account the idea of how much data we are receiving, when to disregard data, and how to handle old data versus new data. Another thing to think about is how to reset the counts and where to save the counts. Obviously we would save the counts for sentiment within hdfs but we will need to keep in mind how easy it will be to keep re-accessing this memory.

When it comes the keeping the real time streaming data prevalent we will need to start making a “score”. This score will be used to determine whether or not the data is old along with its popularity, much like facebook’s scoring for news feeds. Having the score lessen as time moves on we can let the amount of influence that data has lessen because it could be old and not relate to the current measurements. Keeping scores on data will also allow us to get rid of data that is not that important and will allow to not have to use up as much memory.

This may bring up an issue however in how we will want to analyze the data again. If we wanted to pay attention to different times and different years we wouldn’t have these scores and instead would build up the sentiment over the time. Also if we don’t want to re-analyze the data later we could throw away that data and be done with it. This would just have us put things into hdfs and then remove them quickly to make space for more.

Another thing that I could do is have it so that there are update folders with months and dates so that we can keep a lot of data for sentiment. This could potentially take up a lot of space especially with the amount of data the twitter is taking in. If we wanted to analyze all of that in real-time we would need a very large distributed system to handle all of the data. With several tens of terabytes coming in, we would need to handle a lot of tasks and make sure that the system is scalable. It would also need a sufficient network due to the crazy amount of communication that would be happening between nodes. The distributed shared memory would be going everywhere and depending on how the hardware to software tradeoffs are implemented we could have a really horrible structure for such data.