Bilkent University

Department of Computer Engineering

CS 425:

Algorithms for Web-Scale Data

Project Final Report

Relevance Checker

ID: 21500269 – Ali Sabbagh
ID: 21503382 – Kanan Asadov
ID: 21400269 – Shahriyar Mammadli
ID: 21500571 – Solehjon Ruziboev
ID: 21500283 – Yalchin Aliyev

Contents

1. Introduction & Problem Definition

It is very common to see irrelevant comments under an Instagram post. These comments are usually advertisements, spam or something completely not related to the picture. The aforementioned comments both create a visual disturbance and make it hard to find and read the relevant comments. Our intention is to find these relevant comments in the post by scoring them according to their relevance to both image and other comments.

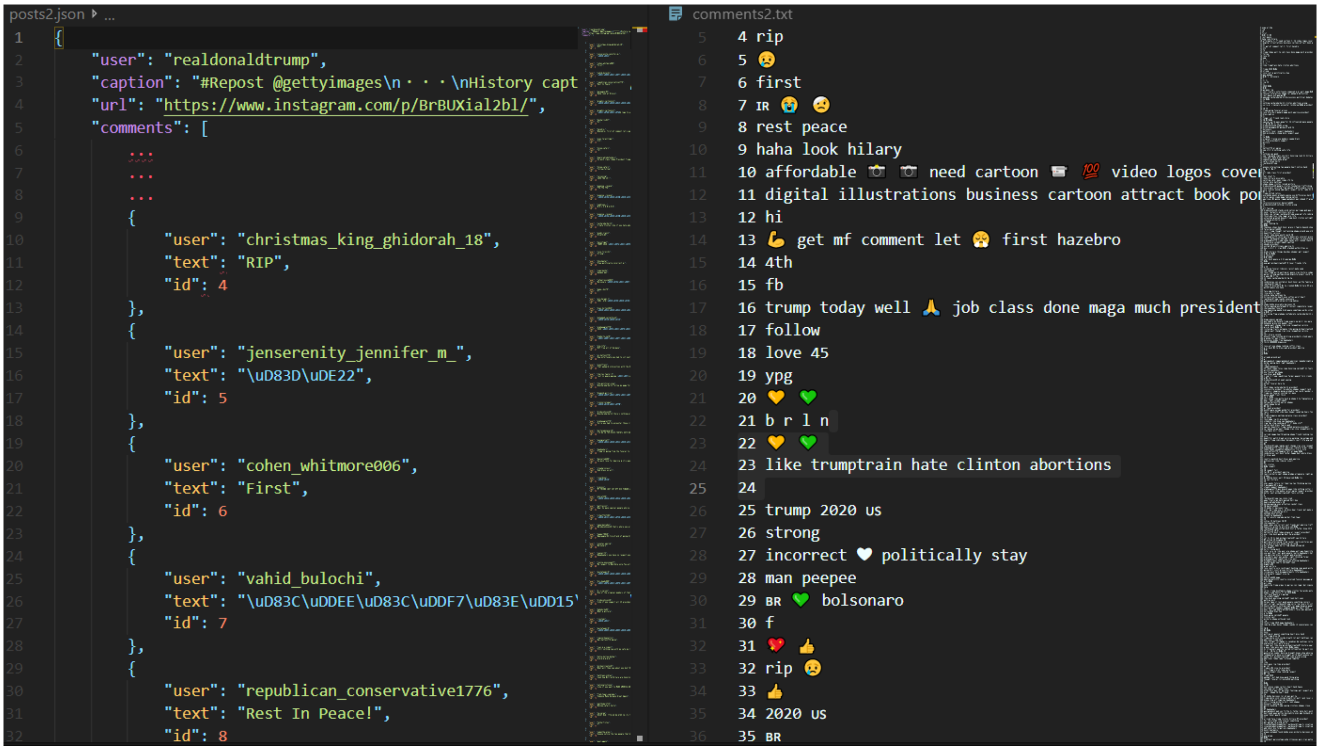
Our first step was to create the dataset and preprocess it in a way that would fit our needs. We started by collecting comments from a post. Moreover, the image and the caption were also obtained to use in the algorithm.

The second step was to analyze the dataset to create our relevance set, a set of words with their individual relevance score. This set would then be used to rank the comments. Here we followed two different algorithms to compare their results and synthesize them to obtain more precise results. First one is to score the comments based on how frequent the words used in this comment appear in other comments. Second one is to score the comment based on the number of important (most used) words this comment uses. Details about the algorithms will be provided in the following sections of the report. After ranking the comments, distance algorithm is applied to consider the mistyped words in the comments. This step helps to improve the correctness of the ranking. Furthermore, keywords that are gathered from the title are also used to score the comments. Google vision API is utilized to find the keywords in the image, after that we improve our relevancy list by the help of keywords, we obtain from the API. The results and correctness of the code will be discussed throughout the report. Moreover, complexity of the algorithm and performance analysis will also be done.

2. The Dataset

2.1. Description

Our dataset is a set of comments under an Instagram post. The main dataset is encoded with JSON format. Apart from the comments, additional information about a post such as who posted it, its caption and a URL to the post is also stored, which is used in the later steps to construct meaningful data according to the results of each algorithm. Here the post URL is unique, and it can be used as an ID when processing multiple post. The dataset also includes the user for each comment. We use the index of the comment in which order it appears under the post as an ID. There are around 20,000 comments on posts from famous people. We have two versions of the same dataset. The second version is the filtered and minimized comments and the IDs. The second dataset is designed in a way that is suitable for map-reduce algorithm to process it in parallel.



Main and Minimized Datasets

2.2. Creating the Dataset

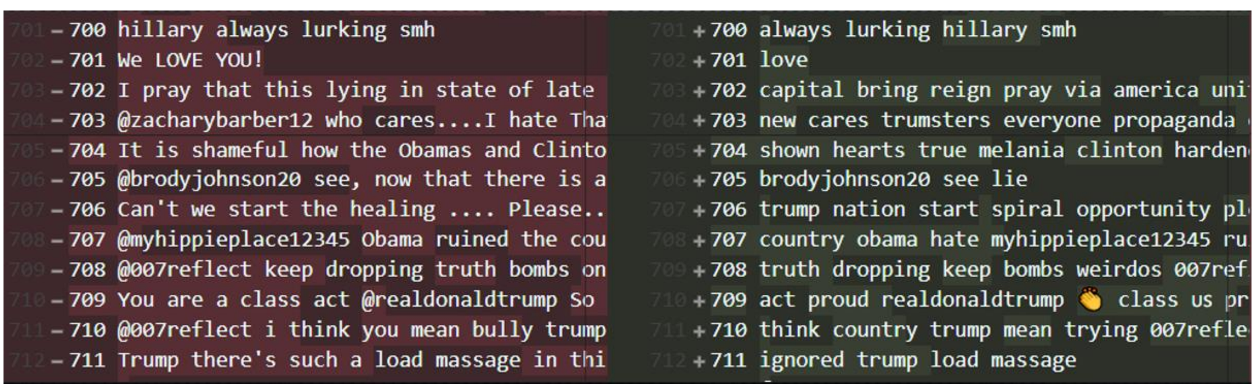
The web page for an Instagram post is single page dynamically loaded web application. To automate the collection of the data, A simple HTML parser is not enough, since it cannot load the dynamic data sending HTTP requests. Furthermore, it is almost impossible on Instagram to create and send requests manually, because they use a query hash to identify if the requests are from a valid session. That is why, Selenium Web driver is used to automate a browser to simulate actions on a web page humanlike. When the web page for a post is loaded for the first time, the comment section only contains around 20-30 comments.

Instagram Post

For each click on the “Load more comments” button, it loads 20-30 more comments. Our web driver waits 500ms after each click to act like a human, so in theory it takes around 10 minutes to load 20,000 comments. But in reality, the elapsed time is much more than 10 minutes because the browser gets loaded to much that it starts slowing down.

When all of the comments are loaded, the source HTML code to feed to a Jsoup parser to extract the useful information. We use this data to construct our main dataset. Jackson JSON serializer to then used to store the dataset as a JSON file.

Moreover, we filter out non-English words, English stop words and repetitions to make the second filtered and minimized dataset to be used in map-reduce.



Raw vs Filtered Comments

3. Comment Ranking Algorithms

MapReduce model was used to implement two different Scoring Systems. The time complexities were analyzed, and the results were then compared with each other (see Section 3.3, 3.4).

Both algorithms start after receiving the preprocessed comments in the following format:

```
66 ☝
67 hatton1776 al agree
68 new still b nothing safe life
69 ❤
70 nothing say good
71 new flag gang signs one bill honoring look 41 hillary
72 like want make great america
73 someone germany make great
74 jafari hniye hello
75 realdonaldtrump
76 f
77 anyone interesting two people heart notice hand
78 ♡ 🇺 ♥ 🇸 😑
79 🙏
80 act lady class first president
81 ❤ ♥
82 chat chief fs
83 looks killary miserable
84 saluting cater mean jimmy jfk 😂
85 niklas helo kathleen
86 shame obama clinton ruined picture
87 brodyjohnson20 google clinton foundation trafficking child
88 love q pain saluting flag upside 5 photo angle luciferian every wwg1wga
89 love rosalyn young omg years respect carter lady 91 94 god first president realize
90 🐸 😊 🙏
91 trump united latins
92 everybody weeks probably gonna use differences back two puts aside sad time thing
93 carter qanon jimmy maga taking option
94 americans 💡 gonna ripgeorgehwbush man indeed 😑 great 🇺🇸 😃 miss 🙁
95 🙏
96 lolololololololol danielledb88
97 brodyjohnson20 nothing clinton wrong
98 ♥
99 art feeling
100 brodyjohnson20 stands prob nation marriage god gay smh
```

Furthermore, the algorithms rank the comments based on their relevance and send the irrelevant comments for revision, giving them a second chance (see section 4). The following two sections will describe how these systems give relevance score to the comments.

3.1. Weighted Scoring System

The main idea of the Weighted Scoring System is that the more some word is used in different comments, the more relevant it is. For example, if everyone is using the word "President" then that particular word is very relevant, thus, it can give a higher relevance score to the comment it was used in. Basically, the more the word is used, the more weight it has to offer to the comment. Hence, the name: Weighted Scoring. After getting the preprocessed data we start our two-stage MapReduce.

3.1.1. Weighted Scoring First Stage

The goal of the first stage is to find the weight of every word. In order to do this, the mapper maps the comments in the following format:

```
<word> : <id of the comment>
```

The reducer in this stage is used to group the key-value pairs. The output of the first stage is, thus, in the following format:

```
<word> : <ids of the comments using this word>
```

3.1.2. Weighted Scoring Second Stage

The second MapReduce stage does the comment ranking. This process is easier to understand with an example.

Let us assume that this is the first line of the output of the first stage:

```
<president> : <1 14 9>
```

This line implies that the word president was used in comments with ids 1, 14 and 9 (weight is 3). Now, the goal of the mapper is to show that the comments 1, 14 and 9 have a word with weight 3. Therefore, the output of the mapper reading the line above will be like this:

```
1  : 3
14 : 3
9  : 3
```

The reducer will take the individual scores for every word and sum them up to produce an output in the following format:

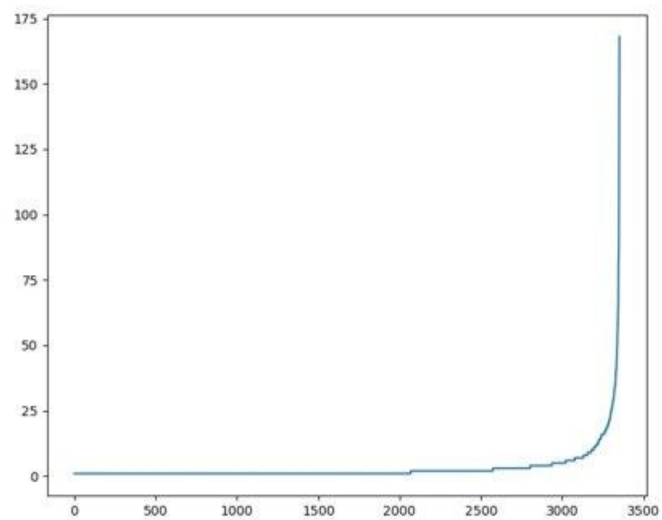
```
<comment id> : <total score>
```

However, Weighted Scoring has a problem. Whenever there is an irrelevant comment containing a word with a large weight, that comment can be considered as relevant which results in some false positives. Constant Scoring was used to solve this problem.

3.2. Constant Scoring System

The main idea of the Constant Scoring is taking not every word and its score for our relevance set (list of keywords used for scoring) as we did before, but only the most important/used ones. Moreover, since the words used in scoring are all important, the score can be incremented by 1 instead of the whole weight.

The main algorithm did not change in this scoring system. However, after the first MapReduce stage the words were sorted by weight in order to find the most important ones.



Layout of Keyword Scores
y axis: keyword score
x axis: keyword index

According to this graph, the last 250 words (3000-3250) can be considered as the most used ones.

The second stage of the MapReduce will be altered so that for every word used in a comment if it is in the relevance set, the comment will get a score of 1.

3.3. Performance

In order to analyze the time complexity of the whole mapreduce stage, the analysis of each sub-mapreduce stage along with additional scripts run in the middle and afterwards will be done.

3.3.1. Time Complexity - Weighted Scoring

Before going into analysis, we will assume that the overhead of mapreduce communication between mappers and reducers is negligible.

The first stage processes all data and renders it in the required way. This gives us `O(N * M)` time complexity, where N is the number of comments and M is the average length of the comments.

The second stage mapper produces N * K total key-value pairs, where N is the same and K is the average number of words in a single comment. The reducer sums up the values of the same key. This gives us the complexity of `O(N * K)`.

The scripts which run after mapreduce stages, produce top keyword lists in descending order and relevant comments along with ids and score in ascending order. The complexities are `O(T * logT)` where T is total number of keywords and `O(N * M * logN)`.

Thus, the total complexity of Weighted Scoring system is

`O(N * M) + O(N * K) + O(T * logT) + O(N * M * logN) => O(N * M * logN)`

3.3.2. Time Complexity - Constant Scoring

The first stage remains the same regarding complexity - `O(N * M)`.

The sorting algorithm run in the middle of the mapreduce stages takes `O(T * logT)` time.

The second stage now produces less key-value pairs since comments don't have as many hits. Consequently, the reducer sums up less values for a given key. The overall complexity becomes O(P), where P is total number of top words occurred in the comments. Note that P << N * K.

The scripts after mapreduce stages produce the same complexity, overall `O(N * M * logN)`.

Therefore, the total complexity of Constant Scoring is

`O(N * M) + O(P) + O(T * logT) + O(N * M * logN) => O(N * M * logN).`

4. Spell Check & Photo labels matching Algorithms

We came up with this step of the algorithm after we thought of and saw scenarios that a comment could have a word that is frequent and relevant to other comments, but it is misspelled, therefore it is not adding to the comment score. Such comments which are previously identified as irrelevant, if run through a spelling checker algorithm against the same keyword list used in comment ranking algorithm, would get a chance to become relevant. We decided to use 2-edit distance technique to catch such misspellings.

Another scenario we considered was when a comment includes a word that matches a word in the post photo caption, or matches something that is actually part of the photo, but no one or not many people mentioned it making it less relevant, while if a word is actually relevant to photo caption or photo labels it should be of highest weight and importance in our algorithm, hence we decided to run the irrelevant comments through a photo caption and photo labels matching checker.

4.1. Algorithm Input

4.1.1. English Dictionary

A comment might have a word that is correct and part of the English dictionary, yet when edit distanced might produce another word that is in the keyword list e.g. “Apply” a word from a comment when edit distanced gives “Apple” a word in the keyword list. We decided that only words that are misspelled in the first place should be checked against the keyword list, hence we use the English dictionary to check if the word from comment is correct or misspelled i.e. part of the dictionary or not. We used a file that has over 466k English new-line-delimited words [].

4.1.2. Photo Caption

From the data creation part mentioned before, a space-delimited file containing the photo caption is produced. The caption is formatted as the comments were i.e only English alphabet allowed and stop words removed, etc.

4.1.3. Image Labels

For getting labels that describe the photo contents we used Google Vision API. After registering our project and obtaining an API key, we coded GoogleVisioner, our class that will take care of establishing the connection with the API, request the label detection of a remote image, and print out the formatted response into a file. The Google Vision API response is in JSON format and provides an array of label objects that have: description, score and topicality attributes. **Description** is the label in text, **score** is how confident the API is that the image is about that label and **topicality** is how much of that image is about that label. As topicality does not reflect relevance or not of a word in a comment, we chose not to include it in output or calculations. If the label description is single word, then it is directly added to output. However, if the label description is made of multiple words, first stop words are filtered out from the description, then description is split into single words each will have the score the full description had. These splitted words are only added to output if they do not match an already added single word label.


```
{
  "description": "officer",
  "score": 0.754245
}
{
  "description": "military officer",
  "score": 0.613217
},
{
  "description": "audience",
  "score": 0.5771099
},
{
  "description": "event",
  "score": 0.5295254
}
```


```
officer 0.754245
audience 0.5771099
event 0.5295254
military 0.613217
```

Google Vision API response formatted and written into our input file

4.1.4. Keywords List

The keywords list that was generated by comment ranking algorithm.

4.1.5. Irrelevant Comments

The output of comment ranking algorithm. A list of comments with score less than the threshold. The file has each comment in a line as: ID Score <Comment>

4.2. The Algorithm

Before going through the algorithm let us define two classes that we will use later.

A Dictionary: a class that takes a file path, reads the words in it hash them into a hashmap, then has a method that given a word <key> checks if the word exists in the hash, returning its <value> or -1 otherwise.

A SpellingChecker: a class that takes file path, reads the words in it hash them into a hashmap,

Edit1: method that given a word, it generates all the other words using 1-edit distance. Changes include: delete, replace, insert, transpose.

Given a word of length n: we get

n words from delete

n-1 words from transpose

(n+1) * 26 words from inserts

n * 26 from replaces

Check: a method that given a **word** calls edit1 and gets all possible 1-edit distance words, then hashes them against the hashmap returns only the ones that collide. If multiple collisions happen the one with highest <value> is saved. Then feeds all 1-edit words to edit1 method again to generate all 2-edit distance words, checks for their collisions and saves the one with highest <value>. Now priority is given to a 1-edit distance match if found return the word <value> otherwise return the <value> of word from 2-edit distance.

The Algorithm

- Read all the algorithm input files
- Hash English dictionary using the Dictionary class
- Hash keyword list words using SpellingChecker class <key> = word, <value> = weight
- When reading keywords save the value of the word with max weight as **maxWeight**
- Hash caption words and image labels as

<key> = word,

<value> = **maxWeight** for caption words

or <value> = (**maxWeight * score)** for image labels *

- Read comment and split into words:
    - Go through the words
        - if word is in English dictionary, check if it matches caption words or image labels and add the value to comment score if successful.
        - if not English = misspelled and of length > 3: check with edit distance if it matches a keyword and/or (caption word or image label) and add value to comment score if successful. **
- Do for all comments and output to new file with possible changed scores.

*: weight of caption words is considered to be as the keyword with highest weight, but image labels weights are the max weight times that label score as the API score reflects certainty.

**: When a word is misspelled, and since we are using 2-edit distance checks, a word of 3 letters or less can get changed totally therefore we only consider words of length > 3.

4.3 Success

4.3.1 Misspellings

Comment 1459 <fush buch> has **buck** word which after 2-edit distance change becomes **bush** which is a word from the photo caption with weight 168 = keyword max weight.

2 ----> 170

```
1407 2 <👀>
1418 2 <waters dom>
1428 2 <mehrab ahmadii>
1438 2 <🙁 ✋>
1459 2 <fush buck>
1495 2 <😦>
1542 2 <0>
1569 2 <💗>
1572 2 <🐕>
```

```
1407 2
1418 2
1428 2
1438 2
1459 170
1495 2
1542 2
1569 2
1572 2
```

comments before and after misspelling check against caption words

Comment 873: has word **presidents** which is plural of **president** a word from the keyword list with weight 168 and can be reached with 1-edit distance.

21 ----> 189

The word president was frequent and repeated in a lot of comments, but presidents was not, such comment was before considered irrelevant while it obviously should be relevant.

```
714 21 <ignorant words myhippieplace12345>        714 21
747 21 <sourpatch dumb comment 84>                747 21
873 21 <remarkable contrast presidents>           873 189
1266 21 <navazandeh hotdog say farhad>            1266 21
391 22 <standing behind everyone>                 391 22
525 22 <netjes hands opgeruimd staat>             525 190
659 22 <loose let ompa lomopa>                    659 22
```

comments before and after misspelling check against keywords

4.3.2 Matching caption

Comment 183: has the hashtag **ripgeorgehwbush** which matches a hashtag in photo caption, it increased comment score by 168 = keyword max weight

4 ----> 172

This comment used a word that not many other comments used, but is in the caption, hence making it relevant.

```
170 4 <third 10000000 dimension>    170 4
183 4 <ripgeorgehwbush>             183 172
210 4 <1a saraa>                    210 4
313 4 <horton4048 kian hey>         313 4
```

comments before and after caption words matching

4.3.3 Matching image labels

Comment 1057: has the word military, a word that is not used frequently in other comments, making the comment score low i.e irrelevant. However, applying the image labels match detection increased the score by 109:

109 = Floor(max weight * military confidence score)

109 = Floor(168 * 0.6494942)

```
1177 12 <security tifaniii line funny>
1234 12 <take l>
1500 12 <♥>
379 13 <wow group>
1057 13 <puck45 rogue military>
1411 13 <death khamenei>
1413 13 <death khamenei>
1511 13 <answer calls dad>
```

```
1177 12
1234 12
1500 12
379 13
1057 122
1411 13
1413 13
1511 13
```

comments before and after labels matching

```
official 0.8018946
crowd 0.6571132
audience 0.5771099
event 0.5295254
military 0.6494942
person 0.6494942
officer 0.613217
```

Image labels and their scores obtained from Google Vision API

5. Results

An Example Post

After running the algorithms on the post above, we obtained the following results.

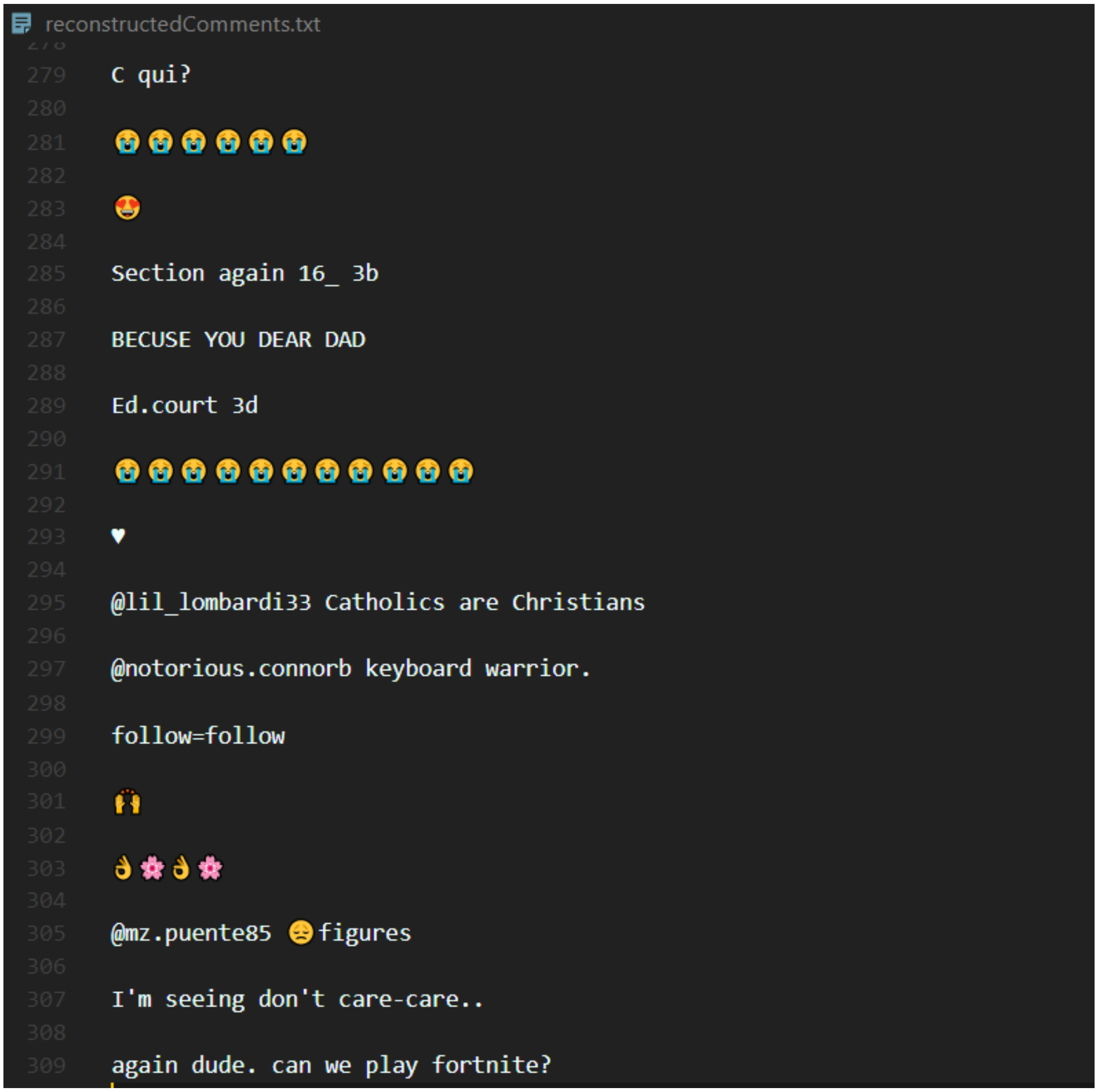
5.1. Weighted Scoring

5.1.1. Relevant Comments

```
reconstructedComments.txt
@josephine.dedominicis well this is about the Bush family....not trump not Clinton’s ....gees in respect for a great man and his family r

@bobby_baccooon because chances are if it came between who the Bush’s wanted at the funeral the Obama’s would be their one and only choic

Sorry, President Trump ~
You’re the only one I respect. Just sayin.’ #wwg1wgaus🦅

God Bless you President Trumpususususus👏🏻

Michell and Hillary look so mad and evil... lol!!! Thank you President Trump for being the BEST president EVER! ❤️❤️

God bless president bush and god bless America 🇺🇸🙏❤️🌹

Yoooo💀 OBAMA like I gotta sit next to these people again smelling like figNewtons 🔥😂 RIP G.H.W Bush.🙏🏾

George H. W. Bush was one of the most respected and respectable presidents we've had. Goodbye to a great American and a true patriot. Tha

A great honourable funeral , for a great President. Good on the Bush family for not making it political , not like McCain . The sadness o

Rest In Peace Former President George Herbert Walker Bush❤️us

Rest in peace Mr President us. The world lost a great man and leader. You will walk in the halls of History with the great leaders of the

@sheshesayz I did!!! This was classy and respectful...and a side note..I for one appreciated the respect shown to President Trump for a c

I felt really bad for President Tramp & First Lady Melania....to have to be next to Obama's & Clinton's the way they acted!!! Trump's har

Total class Mr. President and our Beautiful First Lady; Melania ... God Bless 🙏us

Obama, Michelle, Bill, Hillary; God have mercy on their souls! Judgement must be but don’t forget mercy, O God! I LOVE YOU, TRUMP, AND EV

This was sad to watch yesterday I cried at times but he had a beautiful service at the Cathedral. May Former President George H. W. Bush

@stringsmom ok so if u say this that means you’re most likely a democrat or support the democratic party. Your party over the last 200 ye

**please edit post. ‘Flag-draped casket of former US President George H.W. Bush’ should be #41 NOT #43. George W. Bush #43 is across the
```

Example of Relevant Comments

5.1.2. Irrelevant Comments


reconstructedComments.txt

C qui?

😭😭😭😭😭😭

😍

Section again 16_ 3b

BECUSE YOU DEAR DAD

Ed.court 3d

😭😭😭😭😭😭😭😭😭😭😭

♥

@lil_lombardi33 Catholics are Christians

@notorious.connorb keyboard warrior.

follow=follow

🙌

👌🌸👌🌸

@mz.puente85 😞figures

I'm seeing don't care-care..

again dude. can we play fortnite?

Example of Irrelevant Comments

5.2. Constant Scoring

5.2.1. Relevant Comments

reconstructedComments_2.txt

```
@al_hatton1776 what a good boy donnie was today....sat stil for 1 whole hour...what a good boy... geeez r u kidding me. This is about a f

@myhippieplace12345 racist? racist? have u met your party? done any research? your party started the KKK and continues to run it now, war

George H. W. Bush was one of the most respected and respectable presidents we've had. Goodbye to a great American and a true patriot. Tha

@sheshesayz I did!!! This was classy and respectful...and a side note..I for one appreciated the respect shown to President Trump for a c

@sheshesayz Oh yes...it has been an awe moment throughout this entire funeral. His dog lying next to him in mourning too. Just an amazing

It was a historic time for America for "make America Great"again.Handful American Presidents at the same time,together under same roof.Al

This was sad to watch yesterday I cried at times but he had a beautiful service at the Cathedral. May Former President George H. W. Bush

I'm not upset about the Obama's or the Clinton's being in the pic with Trump, because it only shows how despite our history, we can still

Pres.Bush Sr loved one brought...
For time and space can never divide
Or keep loved one from our side.
When memory paints in colors true,
the happy hours that belonged to
PATRIOTIC AMERICANS."
"There are no goodbyes for us.
Wherever you are Sir, you will always be in our hearts."
The angels are always near to those who are grieving,
to whisper to them that their MOST BELOVED PRES.George H.W.BUSH is safe in the hand of God.
"It was not length of life
PRES.George H.W.BUSH lived,
But it was the depth of LIFE."

@3ku111 omg no offense but are u stupid your party is so corrupt and evil. your party supported slavery and jim crow laws and segregation

Thankyou for MR. PRESIDENT for bringing LOVE 💌and God 🦋back into the 🎄Whitehouse ...into the world! Take care! We Love you so! 🦋😊

God Bless former president Bush 41 and Current President Donald Trump. We need Transparency in our Federal Government. What the Special C
```

Example of Relevant Comments

5.2.2. Irrelevant Comments

```
reconstructedComments_2.txt

👉Restart👈👉 iran👈

@flak_tower_supreme_ Ight we’ll figure out how to clone Trump for y’all

@kamrenporche like they all do now be honest.

USUSUS

@zacharybarber12 @carlykpatt

@future.marine_usmc

یاشاسین ایران IRIR💔 💔 IRIR💔 💔 IRIR💔 💔 IRIR💔 💔 IRIR💔 💔 IRIR💔 💔 IRIR💔 💔 IRIR💔 💔

❤️

@bawbeakari .

@bawbeakari .

Which five? I see three

Rip

RIP

❤️😘

Wow!

@macwagner03 Hillary*

Opgeruimd staat netjes.. Hands down.

He will meet all the people he killed in the gulf war.
```

Example of Irrelevant Comments

5.3. Conclusion

As we can see from the results above, both algorithms produce relevant and irrelevant comments almost correctly. However, since constant scoring gives only one point for every important word, it penalizes short comments but does better with longer comments. Moreover, when analyzing the results from both algorithms we can observe that the relevance order of the comments is similar in most cases. Thus, we can conclude that the algorithms provide reliable data that can be used by Instagram to show the most relevant comments when the post is loaded for the first time.

6. References

[1] Dwyl, “dwyl/english-words,” GitHub, 10-Oct-2018. [Online]. Available: https://github.com/dwyl/english-words. [Accessed: 24-Dec-2018].

[2] www.tutorialspoint.com. (2018). MapReduce Hadoop Implementation. [online] Available at: https://www.tutorialspoint.com/map_reduce/implementation_in_hadoop.htm [Accessed 24 Dec. 2018].

[3] Google Cloud. (2018). Vision API - Image Content Analysis | Cloud Vision API | Google Cloud. [online] Available at: https://cloud.google.com/vision/ [Accessed 24 Dec. 2018].

[4] Seleniumhq.org. (2018). Selenium - Web Browser Automation. [online] Available at: https://www.seleniumhq.org/ [Accessed 24 Dec. 2018].

[5] Nltk.org. (2018). [online] Available at: https://www.nltk.org/nltk_data/ [Accessed 24 Dec. 2018].

[6] Anon, (2018). [online] Available at: https://www.researchgate.net/publication/268334497_Damerau-Levenshtein_Algorithm_and_Bayes_Theorem_for_Spell_Checker_Optimization [Accessed 24 Dec. 2018].