TWITTER DATA ANALYSIS

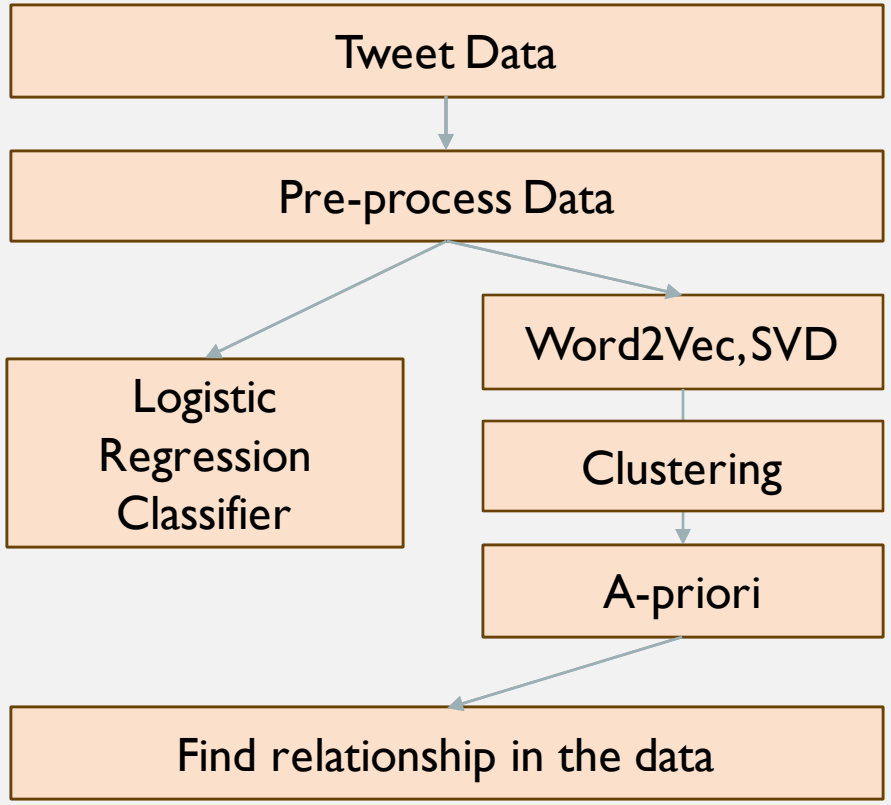
Group 3

David Riemer(202332590)

Olivia Saukonoja

TABLE OF CONTENTS

PURPOSE OF PROJECT



Kaggle Competition	**Twitter sentiment analysis – Determine emotional coloring of tweets**
Goal	Find hidden relationships between tweets
Data format	CSV
Num of Data Columns	2
Using Model	• Word2Vec • Singular Vector Decomposition • K-means • A-priori • Logistic Regression
Performance	Accuracy (prediction only)

METHODOLOGY

Data Preprocessing

Choose Kaggle Data

1) Twitter sentiment analysis
2) Dataset includes 100000 unique tweets

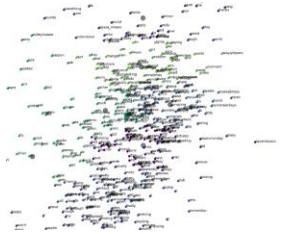
Data Preprocessing

1) Tokenize the words
2) Remove stop words
3) Hash the words to numerical values

Set Database

Run our algorithms on the data

Data Clustering

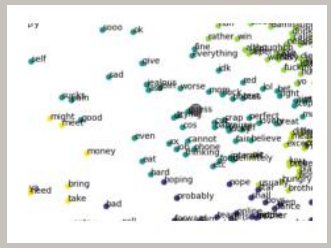


Models Used

1) Word2vec
2) SVD
3) K-means
4) A-priori
5) Logistic Regression

Results and Conclusion

Evaluation of the models



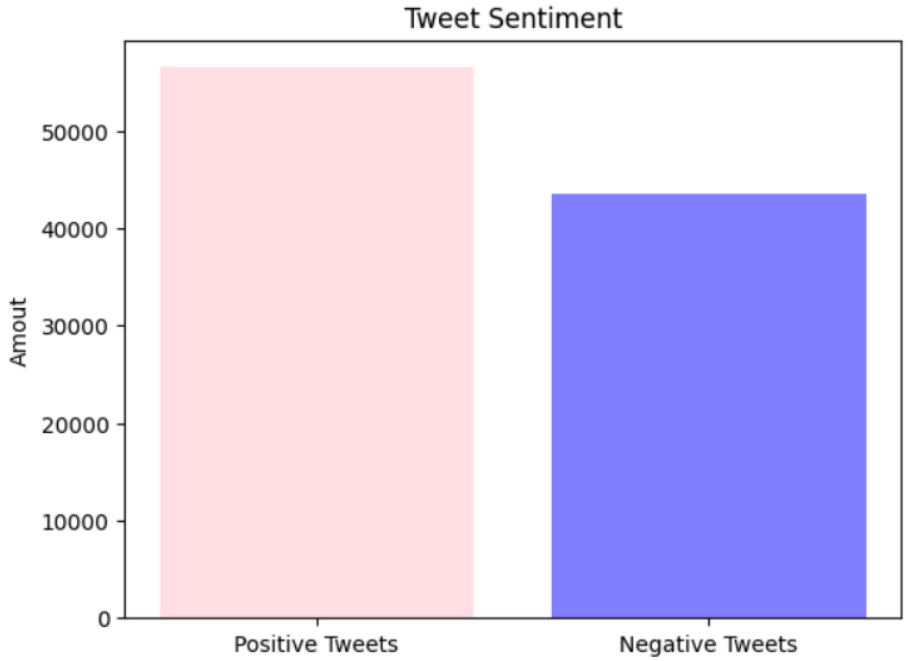
Compare Importance of words

Launch Data Wrangler

	Happy	Sad	Good	Bad
0	birthday	jealous	lovely	sick
1	mother	kinda	nice	horrible
2	mothers	sucks	great	sucks
3	camper	bad	wonderful	crap
4	catirah	cuddlyalex	bexmith	pain
5	belated	amelia_grace	rough	poor
6	bday	shit	_ynnie36	hate
7	fathers	horrible	almienova	kinda
8	tb	face	cglade	unloved
9	wonderful	andreakoeln	perfect	terrible

PREPROCESSING

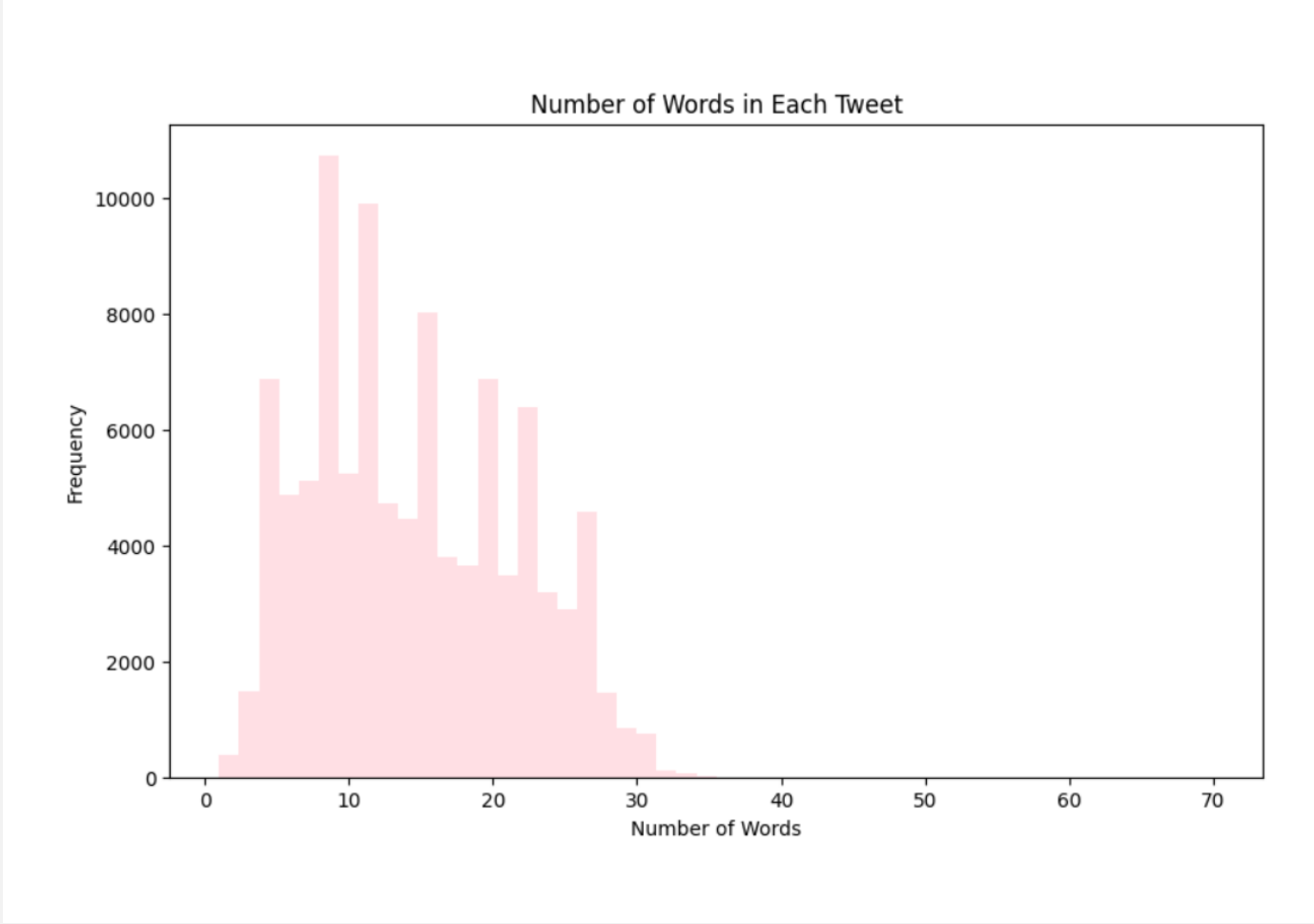
ItemID	SentimentText
1 — 300k	299989 unique values
1	is so sad for my APL friend.....
2	I missed the New Moon trailer...
3	omg its already 7:30 :O
4	.. Omgaga. Im sooo im gunna CRy. I've been at this dentist since 11.. I was suposed 2 jus...
5	i think mi bf is cheating on me!!!



DATA INFORMATION

BASIC VISUALIZATION

- The longest tweet included 70 words, the shortest only 1
- Average number of words: 15



TOKENIZING THE WORDS

```
1  # Add a column with just words
2  regexTokenizer = RegexTokenizer(inputCol="SentimentText", outputCol="words", pattern="\\W")
```

REMOVING STOPWORDS

```
# Remove the stop words
english_stop_words = stopwords.words('english')
english_stop_words.extend(['http', 'https', 'amp', 'rt', 't', 'c', 'the', 'www', 'com'])
stopwordsRemover = StopWordsRemover(inputCol=regexTokenizer.getOutputCol(), outputCol="filtered").setStopWords(english_stop_words)
```

HASHING THE WORDS TO THEIR NUMERICAL VALUES

```
#converts words into numerical values
hashingTF = HashingTF(inputCol=stopwordsRemover.getOutputCol(), outputCol="features")
```

```
1  pipeline = Pipeline(stages=[regexTokenizer, stopwordsRemover, hashingTF]).fit(train)
2
3  dataset = pipeline.transform(train).select("Sentiment", "filtered", "features")
4  dataset = dataset.withColumn("Sentiment", dataset["Sentiment"].cast(IntegerType()))
```

RUN THE PIPELINE AND CREATE THE DATASET

```
1   w2vModel = Word2Vec(sentences=sentences, vector_size=2000, workers=1, seed=1337)
2   w2vModel.save("word2vec.model")
```

CREATING THE WORD2VEC MODEL

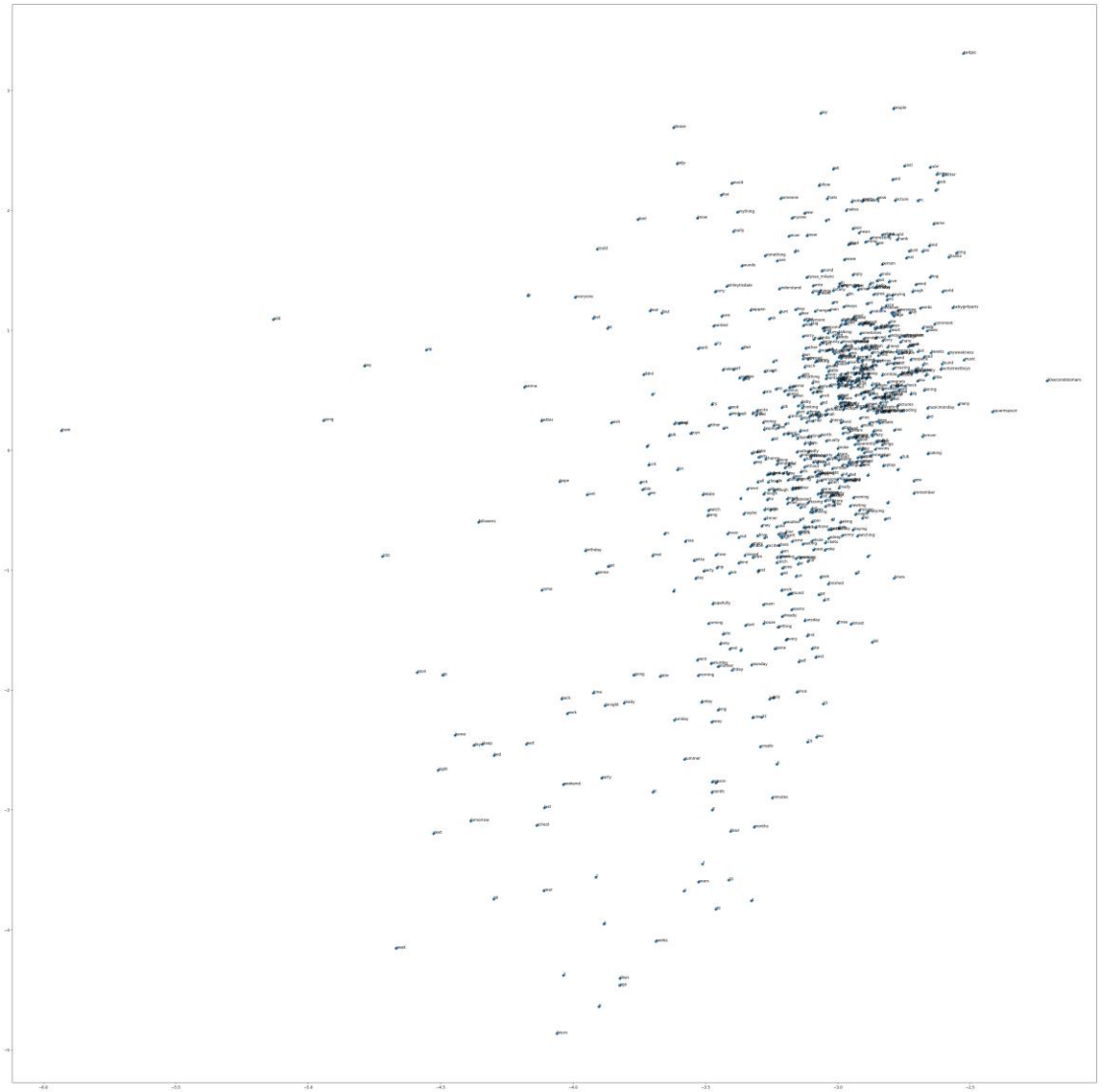
GETTING THE RESULTS FROM WORD2VEC

	Happy	Sad	Good	Bad
0	birthday	jealous	lovely	sick
1	mother	kinda	nice	horrible
2	mothers	sucks	great	sucks
3	camper	bad	wonderful	crap
4	catirah	cuddlyalex	bexmith	pain
5	belated	amelia_grace	rough	poor
6	bday	shit	_ynnie36	hate
7	fathers	horrible	almienova	kinda
8	tb	face	cglade	unloved
9	wonderful	andreakoeln	perfect	terrible

REDUCING THE DIMENSIONS USING SVD

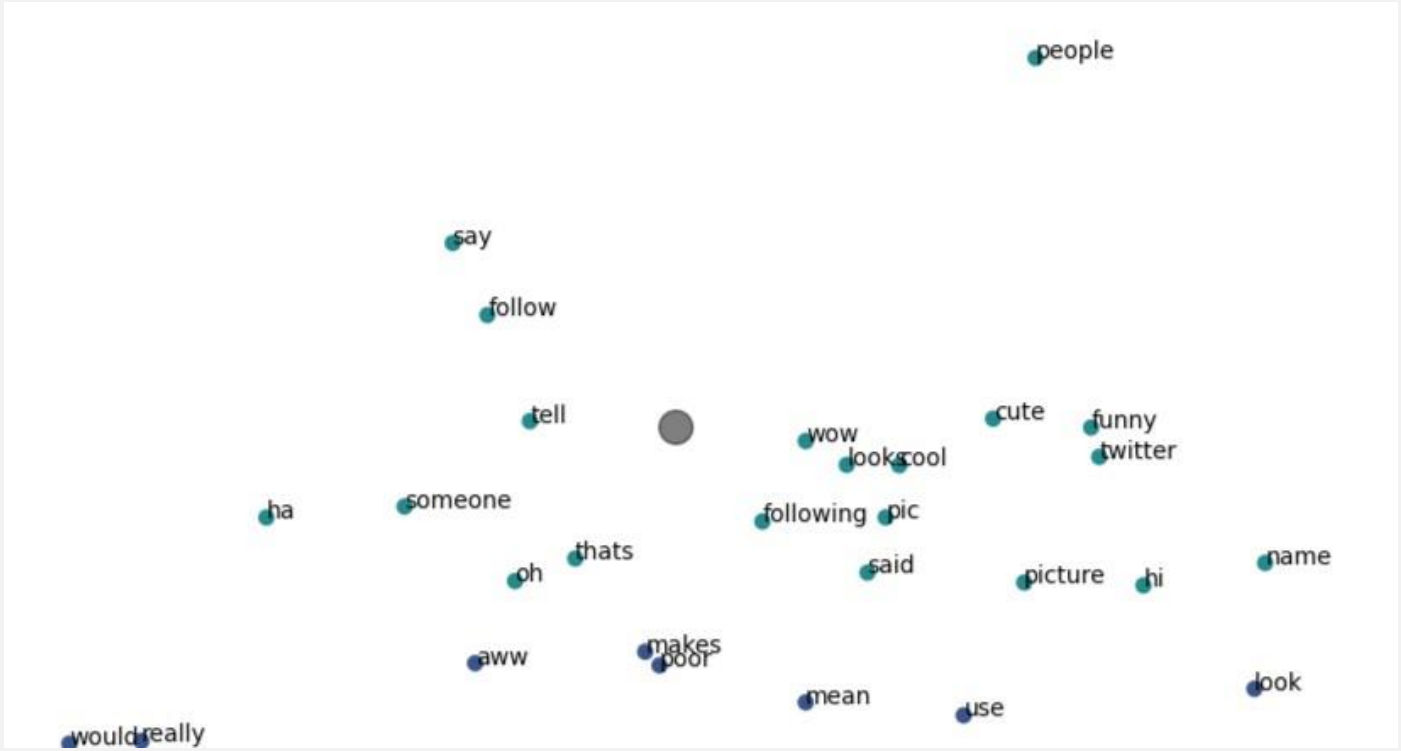
```
from scipy.linalg import svd
from numpy import zeros, diag
#get the most important words (get first 750 rows of vectors)
vectors750 = w2vModel.wv[w2vModel.wv.key_to_index][:750]

U, s, VT = svd(vectors750, full_matrices=False)
Sigma = zeros((vectors750.shape[0], vectors750.shape[1]))
Sigma[:vectors750.shape[0], :vectors750.shape[0]] = diag(s)
n_elements = 2
Sigma = Sigma[:, :n_elements]
VT = VT[:n_elements, :]
# reconstruct
B = U.dot(Sigma.dot(VT))
# transform
SVD_result = U.dot(Sigma)
SVD_resultDF = pd.DataFrame(SVD_result)
SVD_resultDF['x_values'] =SVD_resultDF.iloc[0:, 0]
SVD_resultDF['y_values'] =SVD_resultDF.iloc[0:, 1]
SVD_final = pd.merge(words, SVD_resultDF, left_index=True, right_index=True)
SVD_final['word'] =SVD_final.iloc[0:, 0]
SVD_data_labeled =SVD_final[['word','x_values','y_values']]
```

PLOTTING THE WORDS



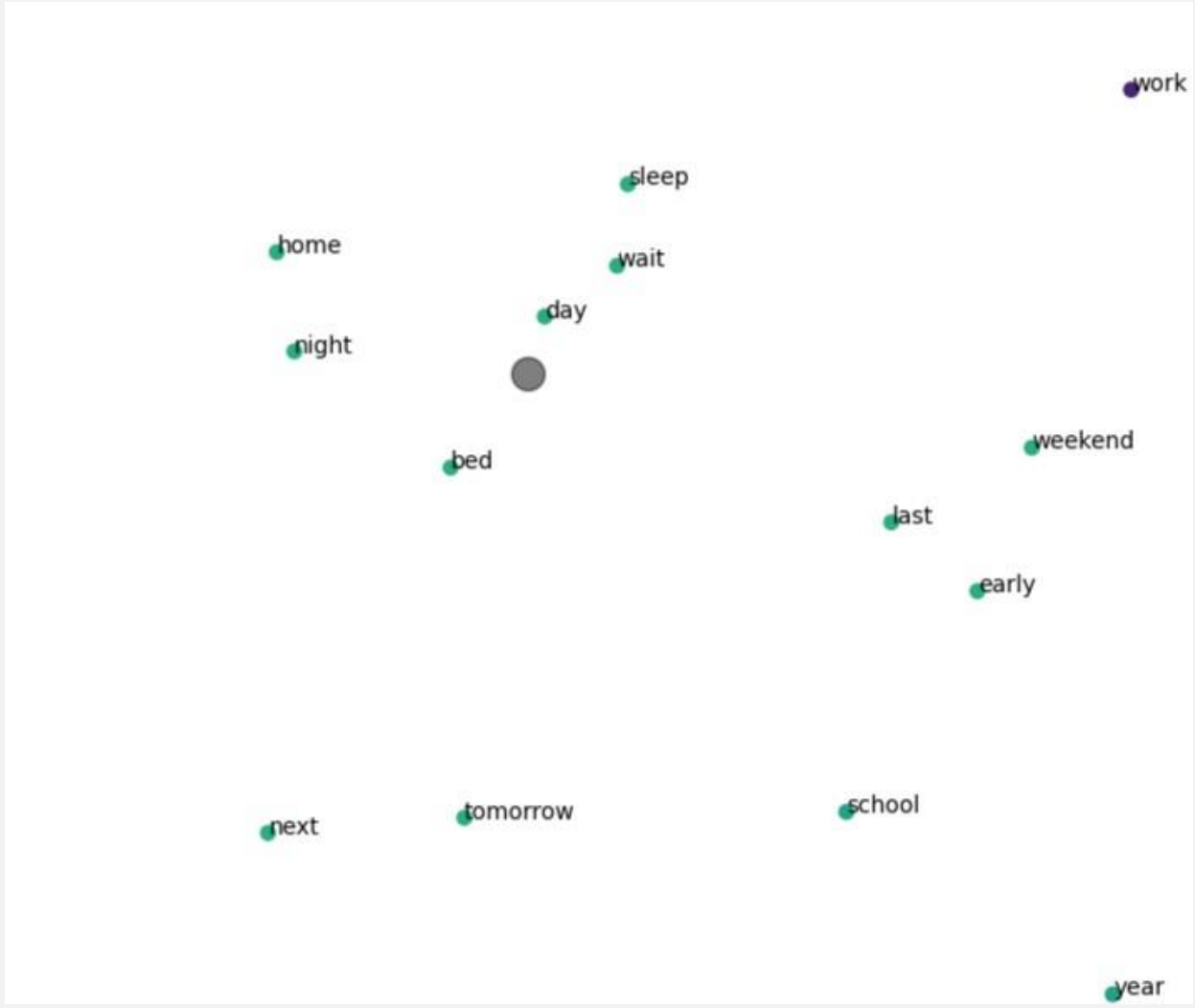
CLUSTER IN THE GRAPH AFTER K-MEANS

- We can see that words which express similar emotions are clustered together



CLUSTER IN THE GRAPH AFTER K-MEANS

- We can see that words which express similar emotions are clustered together



CLUSTER IN THE GRAPH AFTER K-MEANS

- We can see that words which express similar emotions are clustered together

A-PRIORI

SPLITTING THE DATA INTO ONLY NEGATIVE AND POSITIVE SETS

```
#Split the number of tweets into 2 groups: One with positive tweets and one with negative tweets
positiveTweets = dataset.filter(dataset.Sentiment == 1)
negativeTweets = dataset.filter(dataset.Sentiment == 0)
aprioriDFP = positiveTweets.withColumn("wordsWithoutDupes", array_distinct("filtered"))
aprioriDFN = negativeTweets.withColumn("wordsWithoutDupes", array_distinct("filtered"))
#Number of unique words
fpGrowth = FPGrowth(itemsCol="wordsWithoutDupes", minSupport=0.002, minConfidence=0.1)
apriori_modelP = fpGrowth.fit(aprioriDFP)
apriori_modelN = fpGrowth.fit(aprioriDFN)
```

MOST USED WORDS IN

Positive Tweets

items	freq
[good]	4062
[thanks]	3547
[love]	3210
[lol]	3153
[like]	3024
[quot]	2873
[get]	2601
[u]	2503
[know]	2358
[day]	1991

[morning, good]

Negative Tweets

items	freq
[get]	2522
[like]	2260
[know]	2084
[sorry]	2021
[go]	1968
[lol]	1965
[u]	1828
[work]	1619
[good]	1615
[one]	1586

[could, wish]

BUILDING AND APPLYING A LOGISTIC REGRESSION MODEL

CREATING THE MODEL

We decided on a 80/20 split of the data

```
#Split training and Testing
split_data=train.randomSplit([0.8,0.2])
trainSplit=split_data[0]
testSplit=split_data[1]
trainSplit = pipeline.transform(trainSplit).select("Sentiment", "filtered", "features")
testSplit = pipeline.transform(testSplit).select("Sentiment", "filtered", "features")

#Count the number of training tweets and testing tweets
print("Number of training tweets: " + str(trainSplit.count()))
print("Number of testing tweets: " + str(testSplit.count()))

trainSplit.show(5, truncate=False)
testSplit.show(5, truncate=False)
```

```
1  lr = LogisticRegression(labelCol = 'Sentiment', featuresCol='features', maxIter=1000, regParam=0.01)
2  lrModel = lr.fit(trainSplit)
3  print("Done")
4
5  raw_prediction = lrModel.transform(testSplit)
6  raw_prediction.printSchema()
```

APPLYING THE MODEL ON THE TEST DATA

Number of correct predictions: 13879

Accuracy: 69.26685631581574

DISCUSSION

- In this project, we aimed to find relationships between words used in 100,000 tweets. The used algorithms Word2Vec, Dimension Reduction, k-means clustering, A-priori and Logistic Regression were able to find different kinds of relationships between words

- Even with the reduced dimensions, the words in the graph still had a distinguishable similarity between them
- We also found out which words are most likely to be used in positive and negative tweets
- However the ambiguity of some words our results still present conflicting findings

- For instance, it seems contradictory that the word ”good” is one of the most frequently used words in negative tweets as well: in future studies, we believe that implementing more advanced preprocessing techniques could yield improved results, not only in classification but also in generating more intuitive visualizations on the graph