US Bank – Trust Index Measurement

***Shaoni Gupta, Dwarkesh Rajput, Vedangini Kala, Pradnya Patil***  
***Dr. Peng Wang, William Bresler***

Contents

1. [Summary 1](#_Toc79080755)

2. [Introduction 3](#_Toc79080756)

3. [Data understanding and description 4](#_Toc79080757)

4. [Methodologies 6](#_Toc79080758)

5. [Analysis / Results 7](#_Toc79080759)

[6. Conclusion and Recommendations 14](#_Toc79080760)

# Summary

Trust being an important metric in financial industry. According to 2018 Accenture study, 900 companies found drop in trustworthiness costing $180 billion in revenue in over 2 years. With such a soft factor in play, industries implement measures to guage trust in their brand using various ways.

This project is a small step in research study if social media posts could be used to guage trust in the financial industry. The objective was to define a methodology that can be used to meaningfully benchmark the trust value of customers on US Bank’s services in comparison to other banks. The goal was to build a model that helps quantify trust and can be extended to other business-relevant attributes.

We wanted to see if Natural Language Processing(NLP) methods could be used to identify such a nuaced metrics. We used extensive NLP libraries to get to clean the text from social media posts, find semantic similarities and then converge to topics using Latent Dirichlet Allocations (LDA) to work on text clustering and classification.

While we were able to find out about certain dimensions pertaining to the topics of discussions over social media, it is still very difficult to conclusive categorization of posts as trust and non-trust related.

# Introduction

According to Mintel report 2018 consumers must trust their bank even before deciding which account to establish to feel confident about their finances. Trust has drastically recovered since the recession. but customers are still wary and financial service companies have a lot at stake if privacy and security fears are not placated.

Trust in financial services has emerged as a dominant driver of consumer activity. The two major factors contributing are:

**Tangibility**

Financial services are becoming less rooted in real-world interactions (using cash, visiting a branch, personal relationships, etc.)

**Personal Information**

Consumers are understandably concerned about how their personal information is handled in the digital era.

After the financial crisis of 2008-09 , thuest has slowly on the rise since hitting low in 2011 and 2012. Trust is consistently been ranked as an important factor in an individuals decision to bank.

Top 5 factors for choosing a bank are:

Service ease and convenience (47 percent )

Belief in the brand (45 percent )

Rate/price (43 percent )

Timeliness and quality of service resolution (43 percent )

ATMs having a large network reach (40 percent )

Trust is ranked higher than price. Of all the factors mentioned trust may be the most abstract one to try and tackle for companies in the space. The Mintel report also states that while consumer may we wary of the trust in the industry they are generally quick to forgive their own providers while doubting the financial industry.

Because of all these reasons it is rather pertinent for financial institutes to find how well are they trusted across industries and if there are ways in which they could help their customer develop more trust in the brand.

Social media has recently been an effective way for people to raise their voice and get themselves noticed. Also, it is an great platform for brands to directly interact with their audience.

# Data understanding and description

For measurement of trust, the team at US Bank gave us access to a software called Talkwalker. This software’s AI-powered analysis provides real-time insights into what's happening on all social channels and online media, across 187 languages. This enables one to quickly identify issues and complaints before a crisis hits. Benchmark your brand and campaigns with our proven KPI frameworks. Measure sentiment and brand health. Connect social efforts to real business results and provide your management with instant reports. Compare your results to the competition, across every channel. Discover what customers think about your brands and products in real-time  
Different sets of filters were provided to us by the team at USBank. Using the filters, we extracted datasets for the following 5 banks-

1. USBank
2. Bank of America
3. PNC
4. Chase
5. Fifth Third Bank

For the extraction of relevant content from the software, we added additional smart filters on Talkwalker to remove unwanted posts related to advertisements, sponsorships, etc.

Data from the following channels were considered for our analysis:

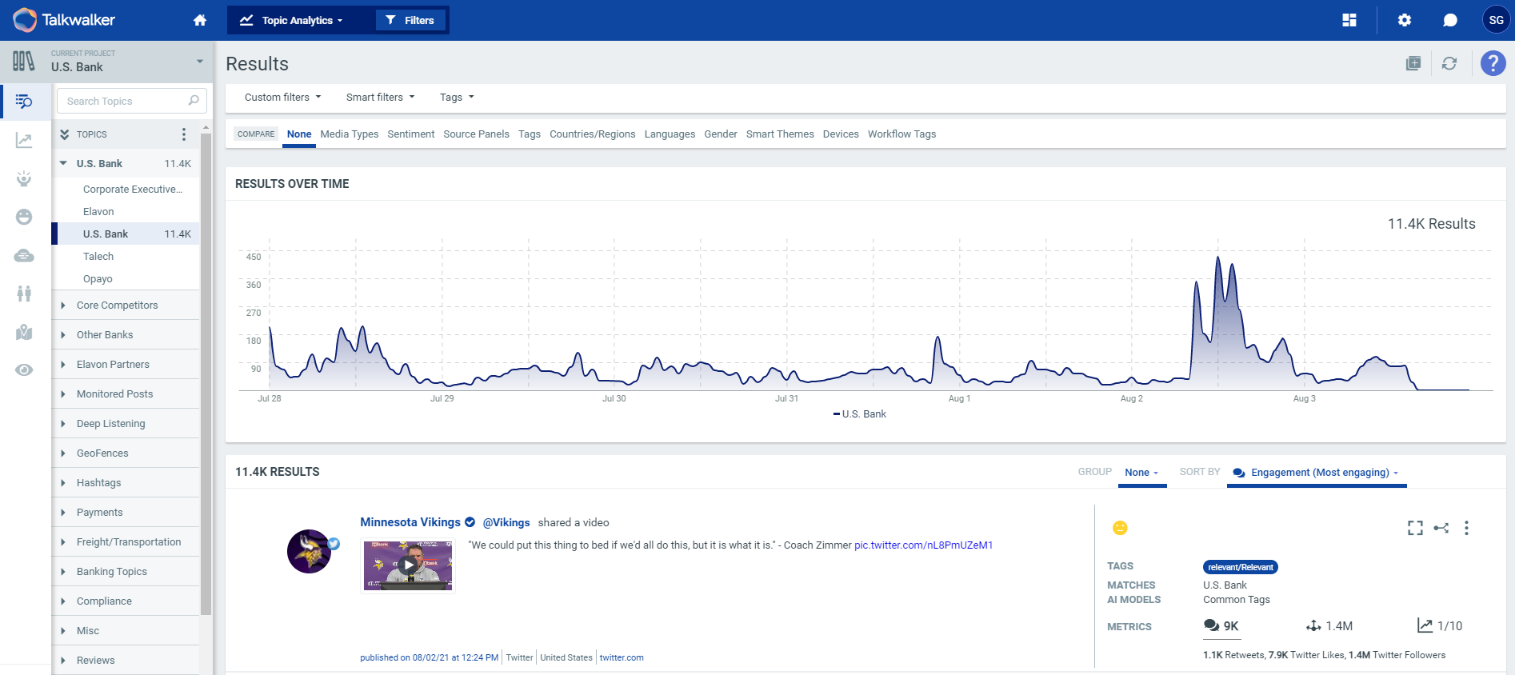
Twitter

Instagram

YouTube

Blogs

Forums (Reddit)

Below is a snapshot of the software used for data extraction   


The data extracted from this software had 99 columns from which we used about 10-15 for our analysis.

For all the social media channels specified, we obtained the date, timestamp, id and the content/title snippet of every individual post. Along with these details, engagement, reach and a sentiment score calculated by the software. The sentiment scores given by the software by –5(negative sentiment), 0 (neutral) and 5 (positive sentiment. For the analysis conducted in approach 1, we used the posts/content with a non-neutral score. The filtered dataset had more relevant content in terms of opinions/complaints expressed by the customers.

Along with the data extracted from Talkwalker, we used the Reddit data.  
On analyzing the data obtained from Talkwalker and Reddit, we could identify major attributes and topics that have a quantifiable impact on Trust. To use the unstructured text data for modelling, we divided the data into several topics and categories that cover all kinds of opinions that were expressed in different social media channels. To make these divisions/categories, we used some literature to expand our domain knowledge on sentiment scoring. The topics and subtopics identified for calculation are as follows:

* Integrity:   
  Data Safety and Ethics  
  Customer Interests  
  Reaction to data breaches
* Competence:   
  Corporate Social Responsibility  
  Cultural changes and commitment to trust  
  Customer Stories  
  Electronic payment and Brand Partnership  
  Infrastructure support  
  Proactive Issue resolution
* Transparency:  
  Privacy policies  
  Process transparency

4. Methodologies

For analysis, we have considered a content post d=string () After tokenization, it appears as- d=list(T(d)) This tokenized and cleansed text forms the basis of all two approaches explained below:

4.1 Approach 1 – Regression  
  
Convert word representations of each w in ct into vectors. Aggregate the vector for each ct. Based on labeled scores for Integrity, Transparency and Competency, build an SBOW that represents a theme( +ve or -ve) for each st. The semantic similarity between vectorized ‘ct’ and SBOW of theme represents the predictor variable for each theme. Based on this representation, fit a regression model to fit a trust score for a given ct.

Steps:

Preprocess each content text ‘ct’ and find word embeddings e for each word w in ct. Embeddings are vectorized representations of w in GloVe.

val=sum(e). val is intended to represent a text that is semantically similar to ‘ct’. Create a neighborhood of 200 terms(arbitrary) closest to val and store it in vn.

Theme ‘th’ for a ‘ct’ is known. For every ct belonging to th, find the intersect of all vals. This represents a BOW for th.

Thematic score for ct is TS= (Intersection of th BOW and vn)/ (word count of ct). The value of each TS for each ct is a predictor variable.

Regress upon predictor variables TS and Y=trust score to fit a model

On regressing the above identified topics on the trust score, the simplified representation of the model looks like:

Trust score = b0 + b1\*Integrity + b2\*Transparency + b3\*Competence

Challenges observed in this approach:

TS in step 4 is heavily dependent on BOW size and content. If the words in BOW for each th have high semantic variance, the overlaps in BOW set and vn will be sparse or potentially disjoint sets.

* GloVe Embedding computation code takes a large amount of time for each iteration due to high dimensionality of the matrix. Any re-computation significantly slows down the model design process.

*3.2 Approach 2 – Topic Modelling*

# Analysis / Results

*5.1 Approach 1*

*A. Analysis*

*#original code. ignore this block*  
**import** **numpy** **as** **npfrom** **scipy** **import** spatial**import** **matplotlib.pyplot** **as** **pltfrom** **sklearn.manifold** **import** TSNEembeddings\_dict = {}**with** open("glove.840B.300d.txt", 'r', encoding="utf-8",errors='ignore') **as** f: **for** line **in** f: values = line.split( " ") word = values[0] vector = np.asarray(values[1:], "float32") embeddings\_dict[word] = vectortsne = TSNE(n\_components=2, random\_state=0)words = list(embeddings\_dict.keys())vectors = [embeddings\_dict[word] **for** word **in** words]Y = tsne.fit\_transform(vectors[:1000])plt.scatter(Y[:, 0], Y[:, 1])**for** label, x, y **in** zip(words, Y[:, 0], Y[:, 1]): plt.annotate(label, xy=(x, y), xytext=(0, 0), textcoords="offset points")plt.show()

Convert word representations of each w in ct into vectors. Aggregate the vector for each ct. Based on labeled scores for Integrity, Transparency and Competency, build an SBOW that represents a theme( +ve or -ve) for each st. The semantic similarity between vectorized ct and SBOW of theme represents the predictor variable for each theme. Based on this representation, fit a regression model to fit a trust score for a given ct.

Approach: Step 1. Preprocess each content text ct and find word embeddings e for each word w in ct. Embeddings are vectorized representations of w in GloVe.

**import** **pandas** **as** **pdimport** **numpy** **as** **npimport** **osimport** **reimport** **numpy** **as** **npimport** **pandas** **as** **pdimport** **matplotlib.pyplot** **as** **pltfrom** **sklearn.feature\_extraction.text** **import** CountVectorizer**import** **nltk** **import** **stringimport** **reimport** **string**filepath='C:**\\**Users**\\**Dwarkesh**\\**Documents**\\**UC**\\**Summer**\\**BANA7095- Graduate Case Studies in Business Analytics**\\**Scoring**\\**Chase Bank Level 1.csv'doc\_df = pd.read\_csv(filepath)**import** **re**doc\_df['content\_text\_processed'] =doc\_df['content'].map(**lambda** x: re.sub('[,\.!?]', '', str(x)))doc\_df['content\_text\_processed'] = doc\_df['content\_text\_processed'].map(**lambda** x: x.lower())doc\_df['content\_text\_processed'].head()**def** tokenization(text): text = re.split('\W+', text) **return** text*#df['Tweet\_tokenized'] = df['title\_wo\_punct'].apply(lambda x: tokenization(x.lower()))***import** **nltk**nltk.download('stopwords')stopword = nltk.corpus.stopwords.words('english')ps = nltk.PorterStemmer()

[nltk\_data] Downloading package stopwords to[nltk\_data] C:\Users\Dwarkesh\AppData\Roaming\nltk\_data...[nltk\_data] Package stopwords is already up-to-date!

*#Function declarations***def** remove\_stopwords(text): text = [word **for** word **in** text **if** word **not** **in** stopword] **return** text**def** remove\_punctuation(text): no\_punct=[words **for** words **in** text **if** words **not** **in** string.punctuation] words\_wo\_punct=''.join(no\_punct) **return** words\_wo\_punct**def** stemming(text): text = [ps.stem(word) **for** word **in** text] **return** text**def** pre(df,col): df['title\_wo\_punct']=df[col].fillna(' ').apply(**lambda** x: remove\_punctuation(x)) **if** 'Unnamed: 0' **in** df.columns: df=df.drop(['Unnamed: 0'],axis=1) df['Tweet\_tokenized'] = df['title\_wo\_punct'].apply(**lambda** x: tokenization(x.lower())) df['Tweet\_nonstop'] = df['Tweet\_tokenized'].apply(**lambda** x: remove\_stopwords(x)) df['Tweet\_stemmed'] = df['Tweet\_nonstop'].apply(**lambda** x: stemming(x)) *#GloVe based functions***def** find\_closest\_embeddings(embeddings\_dict,embedding): **return** sorted(embeddings\_dict.keys(), key=**lambda** word: spatial.distance.euclidean(embeddings\_dict[word], embedding))**def** compute\_content\_embedding(embeddings\_dict,content,n\_close=100): val=np.zeros\_like(embeddings\_dict['king']) *#print(val)* **for** i **in** content: **if** i **in** embeddings\_dict.keys(): val+=embeddings\_dict[i] content\_sum=find\_closest\_embeddings(embeddings\_dict,val)[:n\_close] **return** content\_sum

Step 1 completed. head of new columns **in** the main dataframe

*#main preprocessing and newly generated columns. Step 1 completed*pre(doc\_df,col='content')doc\_df.iloc[:,-6:-1].head()

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **custom.version** | **content\_text\_processed** | **title\_wo\_punct** | **Tweet\_tokenized** | **Tweet\_nonstop** |
| **0** | NaN | find the latest steady app promotions to work ... | Find the latest Steady App Promotions to work ... | [find, the, latest, steady, app, promotions, t... | [find, latest, steady, app, promotions, work, ... |
| **1** | NaN | jpmorgan chase & co - jpmorgan chase is a lead... | JPMorgan Chase Co JPMorgan Chase is a leadin... | [jpmorgan, chase, co, jpmorgan, chase, is, a, ... | [jpmorgan, chase, co, jpmorgan, chase, leading... |
| **2** | NaN | outlook: positive outlook ceo approval: approv... | Outlook Positive Outlook CEO Approval Approves... | [outlook, positive, outlook, ceo, approval, ap... | [outlook, positive, outlook, ceo, approval, ap... |
| **3** | NaN | outlook: positive outlook ceo approval: approv... | Outlook Positive Outlook CEO Approval Approves... | [outlook, positive, outlook, ceo, approval, ap... | [outlook, positive, outlook, ceo, approval, ap... |
| **4** | NaN | ceo approval: approves of ceo\n\ntakes care of... | CEO Approval Approves of CEO\n\ntakes care of ... | [ceo, approval, approves, of, ceo, takes, care... | [ceo, approval, approves, ceo, takes, care, em... |

**import** **numpy** **as** **npfrom** **scipy** **import** spatial**import** **matplotlib.pyplot** **as** **pltfrom** **sklearn.manifold** **import** TSNEclean\_glove=[]embeddings\_dict = {}i=0j=0word\_dict=[]print(j)

0

1. val=sum(e). val is intended to represent a text that is semantically similar to ct. Create a neighborhood of 200 terms(arbitrary) closest to val and store it in vn.
2. Theme th for a ct is known. For every ct belonging to th, find the intersect of all vals. This represents a BOW for th.
3. Thematic score for ct is TS=(Intersection of th BOW and vn)/(word count of ct). The value of each TS for each ct is a predictor variable.
4. Regress upon predictor variables TS and Y=trust score to fit a model

*Extraction of GloVe Embeddings*

*#decode utf8#line split(" ")#extraction of GloVe embeddings#CAUTION: computationally expensive block. reads 5+ GB of txt#with open("glove.840B.300d.txt", 'r', encoding="utf-8",errors='ignore') as f:# try:* *# for line in f:* *#print(line)* *# values = line.split(" ")# clean\_glove.append(line)* *#print(values)# word=values[0]# word\_dict.append(values[0])# vector = np.asarray(values[1:], "float32")* *# embeddings\_dict[word] = vector# j=len(values)# except:# i+=1# f.\_\_next\_\_()#print(find\_closest\_embeddings(embeddings\_dict,embeddings\_dict["twig"] - embeddings\_dict["branch"] + embeddings\_dict["hand"])[:5])*

*#sample embedding output in vector form*embeddings\_dict["king"][1:10]

array([-0.35068, 0.42923, -0.53825, -0.1848 , -0.31082, 0.29196, -0.7103 , -0.23867, 1.8471 ], dtype=float32)

**import** **csv***#write operation executed once.#CAUTION: This is an expensive operation since output file size>2GB# writing the data into the file#file = open('embed\_dict.csv', 'w', newline ='\n',encoding="utf-8")#with file:* *#write = csv.writer(file)* *#write.writerows([[vec] for vec in embeddings\_dict])# for key in embeddings\_dict.keys():# file.write("%s,%s\n"%(key,embeddings\_dict[key]))*

*#file contains list of words pre-extracted from 1st row of GloVe. Count is 2million +#CAUTION: computationally expensive block. file read takes time#with open("C:\\Users\\Dwarkesh\\Documents\\UC\\Summer\\BANA7095- Graduate Case Studies in Business Analytics\Data Analysis\\glove\_dict.csv", 'r', encoding="utf-8",errors='ignore') as f:# reader = csv.reader(f)# data = list(reader)#from itertools import chain#flatten\_values = list(chain.from\_iterable(data))#print((flatten\_values[-1]))#print((word\_dict[-50:-25]))*

**---------------------------------------------------------------------------KeyboardInterrupt** Traceback (most recent call last)**<ipython-input-7-7ffcf5859304>** in <module> 1 **with** open**("C:\\Users\\Dwarkesh\\Documents\\UC\\Summer\\BANA7095- Graduate Case Studies in Business Analytics\Data Analysis\\glove\_dict.csv",** **'r',** encoding**="utf-8",**errors**='ignore')** **as** f**:** 2 reader **=** csv**.**reader**(**f**)----> 3** data **=** list**(**reader**)** 4 **from** itertools **import** chain 5 flatten\_values **=** list**(**chain**.**from\_iterable**(**data**))~\anaconda3\lib\codecs.py** in decode**(self, input, final)** 317 **raise** NotImplementedError 318 **--> 319 def** decode**(**self**,** input**,** final**=False):** 320 **# decode input (taking the buffer into account)** 321 data **=** self**.**buffer **+** input**KeyboardInterrupt**:

*#Top 6 words semantically close to "king" using custom function*print(find\_closest\_embeddings(embeddings\_dict,embeddings\_dict["king"])[1:6])

['kings', 'prince', 'King', 'queen', 'throne']

*# Importing library#saving all glove words to a new file# data to be written row-wise in csv fil#data = [['Geeks'], [4], ['geeks !']]# opening the csv file in 'w+' mode#file = open('glove\_dict.csv', 'w+', newline ='\n',encoding="utf-8")# writing the data into the file#with file:# write = csv.writer(file)# write.writerows([[word] for word in word\_dict])*

*Representation of semantically similar words(neighbourhood creation)*

*#sample representation of matrix operations on word vectors#CAUTION: computationally expensive. ~5mins for these two function calls)#print(find\_closest\_embeddings(embeddings\_dict,embeddings\_dict["king"])[1:6])*print(find\_closest\_embeddings(embeddings\_dict, embeddings\_dict['twig'] - embeddings\_dict["branch"] + embeddings\_dict["hand"])[:5])

['twig', 'hand', 'finger', 'hands', 'fingers']

*Sample representation of step 3- thematic BOW.*

*# A sample thematic BOW. Only find elements of BOW also present in GloVe Word list*integrity\_list=['True','uprightness','probity','rectitude','honor','honorableness','upstandingness','good character','principle(s)','ethics','morals','righteousness','morality','nobility','high-mindedness','right-mindedness','noble-mindedness','virtue','decency','fairness','scrupulousness','sincerity','truthfulness','unity','unification','wholeness','coherence','cohesion','undividedness','togetherness','solidarity','coalition','soundness','robustness','strength','sturdiness','solidity','solidness','durability','stability','stoutness','toughness']print(integrity\_list)**for** i **in** integrity\_list: **if** i **not** **in** embeddings\_dict.keys(): print(i)

Returning back to step 2. Step 2.3 Example content text matching against glove word list. Helps in elimiating some noise

*#Sample content text is broken down into word list. Finding words only present in GloVe.* sample\_corpus=['our', 'women', 's', 'conf', 'kicks', 'off', 'tomorrow', 'at', '12pm', 'women', 'and', 'work', 'let', 's', 'talk', 'about', 'equity', 'thanks', 'statestreet', 'massgenbrigham', 'libertymutual', 'pncbank', 'wellingtonmgmt', 'bostonchamber', 'easternbank', 'cambridgebank', 'culturedei', 'foxwoodsct', 'equity', 'dei', 'advancingwomen', 'join', 'us', 'tomorrow', 'httpstco4ep7utlgfn']print("Part of the tweet:: **\n**",sample\_corpus)sample\_corpusprint("**\n** Not present in Glove:: **\n**")**for** i **in** sample\_corpus: **if** i **not** **in** embeddings\_dict.keys(): print(i)

Part of the tweet:: ['our', 'women', 's', 'conf', 'kicks', 'off', 'tomorrow', 'at', '12pm', 'women', 'and', 'work', 'let', 's', 'talk', 'about', 'equity', 'thanks', 'statestreet', 'massgenbrigham', 'libertymutual', 'pncbank', 'wellingtonmgmt', 'bostonchamber', 'easternbank', 'cambridgebank', 'culturedei', 'foxwoodsct', 'equity', 'dei', 'advancingwomen', 'join', 'us', 'tomorrow', 'httpstco4ep7utlgfn'] Not present in Glove:: statestreetmassgenbrighamlibertymutualpncbankwellingtonmgmtbostonchambereasternbankcambridgebankculturedeifoxwoodsctadvancingwomenhttpstco4ep7utlgfn

*Population of Neighbourhood For the given content text sample\_corpus, we aggregate the word vectors and find the 200 words in the nearest neighbor. The logic defined in the function is to sum up all the words in content text and sort the final vector to obtain 200 closest neighbours*

*#function call for single row of compute\_content\_embedding*compute\_content\_embedding(embeddings\_dict,sample\_corpus,200)

['us', 'them', 'him', 'we', 'me', 'everyone', 'her', 'tomorrow', 'going', 'hope', 'take', 'they', 'she', 'women', 'go', 'let', "'ll", 'what', 'get', 'people', 'things', 'talk', 'so', 'something', 'feel', 'myself', 'our', 'you', 'their', 'day', 'time', 'will', 'start', 'keep', 'someone', 'come', 'doing', 'really', 'want', 'every', 'how', 'know', 'give', 'it', 'guys', 'else', 'he', 'money', 'help', 'before', 'ourselves', 'ask', 'hear', 'morning', 'tonight', 'there', 'opportunity', 'work', 'just', 'again', 'themselves', 'next', 'tell', 'if', 'trying', 'chance', 'meet', 'too', 'my', 'leave', 'taking', 'getting', 'happy', 'anything', 'pay', 'yourself', 'today', 'right', 'night', 'coming', 'way', 'do', 'make', 'sure', 'men', 'wait', 'those', 'happen', 'stay', 'begin', 'try', 'think', 'some', 'stop', 'always', 'his', 'but', 'up', 'until', 'continue', 'out', 'gonna', "'re", 'because', 'would', 'need', 'down', 'everything', 'enough', 'bring', 'spend', 'speak', 'off', 'i', 'your', 'friends', 'share', 'hopefully', 'able', 'willing', 'kids', 'when', 'much', 'around', 'back', 'whatever', 'good', 'ahead', 'together', 'why', 'weekend', 'feeling', 'thinking', 'helping', 'nothing', 'life', 'place', 'could', 'enjoy', 'girls', 'mind', 'then', 'job', 'past', 'open', 'thing', 'even', 'evening', 'join', 'few', 'away', 'meeting', 'where', 'never', 'everybody', 'believe', 'rest', 'now', 'giving', 'church', 'world', 'all', 'walk', 'once', 'should', 'see', "'m", 'fun', 'own', 'big', 'say', 'call', 'understand', 'live', 'opportunities', 'put', "n't", 'care', 'here', 'future', 'week', 'learn', 'be', 'love', 'soon', 'that', 'parents', 'school', 'find', 'might', 'maybe', 'talking', 'decide', 'afternoon', 'anyway', 'while', 'anyone', 'telling', 'public', 'expect']

2.5 Replicating the computation for a content text in the data frame.

In [40]:

compute\_content\_embedding(embeddings\_dict,doc\_df["Tweet\_nonstop"][2],200)

['financial', 'job', 'pay', 'income', 'money', 'economic', 'salary', 'health', 'positive', 'outlook', 'investment', 'employment', 'debt', 'plan', 'growth', 'balance', 'cash', 'life', 'business', 'spending', 'credit', 'employee', 'long-term', 'increase', 'profit', 'earnings', 'paying', 'future', 'change', 'benefits', 'interest', 'government', 'decisions', 'tax', 'leadership', 'work', 'bank', 'employees', 'education', 'insurance', 'good', 'loan', 'retirement', 'management', 'policy', 'risk', 'revenue', 'economy', 'paid', 'higher', 'willing', 'personal', 'likely', 'opportunity', 'opportunities', 'important', 'they', 'he', 'negative', 'amount', 'better', 'career', 'expectations', 'potential', 'if', 'deal', 'because', 'should', 'corporate', 'social', 'keep', 'improve', 'sense', 'profits', 'care', 'lose', 'would', 'much', 'person', 'responsibility', 'bad', 'expected', 'understanding', 'mortgage', 'sales', 'equity', 'affect', 'hope', 'how', 'certain', 'able', 'finance', 'getting', 'him', 'but', 'energy', 'get', 'what', 'really', 'take', 'could', 'plans', 'company', 'healthy', 'strategy', 'payment', 'budget', 'someone', 'might', 'situation', 'investments', 'feel', 'commitment', 'decision', 'so', 'ability', 'employer', 'time', 'fund', 'compensation', 'approval', 'going', 'changes', 'payments', 'significant', 'enough', 'jobs', 'rather', 'fiscal', 'necessary', 'process', 'healthcare', 'need', 'funds', 'things', 'lack', 'expect', 'account', 'finances', 'give', 'needs', 'rate', 'market', 'even', 'medical', 'will', 'costs', 'strong', 'it', 'loss', 'wages', 'matter', 'increasing', 'move', 'policies', 'spend', 'banking', 'investors', 'benefit', 'seek', 'marketing', 'gain', 'loans', 'position', 'achieve', 'experience', 'their', 'cost', 'make', 'short-term', 'share', 'taxes', 'not', 'stay', 'kind', 'confidence', 'due', 'hopefully', 'relationship', 'monetary', 'continue', 'way', 'basis', 'progress', 'trust', 'people', 'too', 'rates', 'taking', 'them', 'doing', 'may', 'invest', 'success', 'that', 'difficult', 'mind', 'maintain', 'ways', 'his']

The below code in block 112 is to be used to create 200-word neighborhoods for 21k rows. Extracting the whole output will take 15 days in its current form. Need to optimize it. Do not execute it. To be used after refinement.

*#CAUTION: Computationally expensive.#doc\_df["ct\_neighbour"]=doc\_df['Tweet\_nonstop'].map(lambda x: compute\_content\_embedding(embeddings\_dict,x,200))#start time: 3PM stop time:4:08 PM. rows processed=0.*

temp\_df=pd.Series()*#Testing for performance and computational expense#temp\_df=pd.Series(doc\_df['Tweet\_nonstop'])#temp\_df[1:5]=doc\_df['Tweet\_nonstop'][1:5].map(lambda x: compute\_content\_embedding(embeddings\_dict,x,200))#start time 2:52 #end time: 2:58 for 5 rows*

C:\Users\Dwarkesh\anaconda3\lib\site-packages\ipykernel\_launcher.py:1: DeprecationWarning: The default dtype for empty Series will be 'object' instead of 'float64' in a future version. Specify a dtype explicitly to silence this warning. """Entry point for launching an IPython kernel.

*We replicate the code of in the block 40 for populating the neighborhood words of the first 25(i) rows of main dataframe. This is a computationally expensive step(1min/row). The output is stored in a separate table since the operation is performed only for a select number of rows.*

*#Attempt 2#Testing for performance and computational expense*i=25**for** x **in** doc\_df['Tweet\_nonstop'][0:i]: temp\_val=pd.Series([compute\_content\_embedding(embeddings\_dict,x,200)]) temp\_df=temp\_df.append(temp\_val)*#temp\_df[1:25]=doc\_df['Tweet\_nonstop'][1:25].map(lambda x:* compute\_content\_embedding(embeddings\_dict,x,200))#start time 5:19 #end time: for 25 rows

*List version of the same output in block below. Each element itself is a list of 200 words representing the neighborhood of each row*

*#temp\_val.append(pd.Series([compute\_content\_embedding(embeddings\_dict,doc\_df['Tweet\_nonstop'][1],200)]))***for** x **in** doc\_df['Tweet\_nonstop'][0:i]: print([compute\_content\_embedding(embeddings\_dict,x,200)])

Neighbourhoods stored in a separate series

temp\_df

0 [job, jobs, opportunities, help, money, opport...0 [financial, institutions, investment, economic...0 [financial, debt, income, economic, investment...0 [economic, financial, initiatives, investment,...0 [financial, government, policies, agencies, fe...0 [earn, cash, pay, money, paying, dollars, bonu...0 [nigga, niggas, yall, gotta, gonna, tryin, nig...0 [satellite, pay, they, insurance, car, could, ...0 [outlook, environment, economic, management, h...0 [dont, u, cant, didnt, cuz, shit, i, doesnt, g...0 [money, dollars, bank, cash, pay, him, they, t...0 [police, him, he, jail, someone, cops, prison,...0 [dollars, money, dollar, billion, year, spendi...0 [u, i, dont, lol, bank, ur, american, dollar, ...0 [career, college, someone, job, student, he, h...0 [business, companies, financial, professionals...0 [companies, customers, consumers, market, user...0 [management, financial, decisions, health, dev...0 [job, business, team, management, experience, ...0 [metals, commodities, commodity, crude, oil, m...0 [companies, customers, consumers, market, user...0 [cash, money, pay, credit, paying, dollars, ea...0 [ny, york, united, MortgageMarvel.com, state, ...0 [card, credit, pay, cards, cash, payment, mone...0 [money, cash, earn, pay, paying, spend, dollar...dtype: object

Output saved into csv

temp\_df.to\_csv("SampleNeighbourhoodSeries25.csv")

val=np.zeros\_like(embeddings\_dict['king'])

*#sample matrix operations represented below:*embeddings\_dict['king']-embeddings\_dict['queen']

array([-0.09408 , -0.12374999, 0.17561 , -0.17770004, 0.18615001, 0.04099 , -0.21473002, 0.06866997, 0.08703999, 0.35759997, -0.41689798, -0.14531001, 0.196125 , 0.01831001, -0.13628101, -0.287278 , -0.01310998, 0.02252001, 0.123005 , 0.27718002, 0.19306 , -0.41014004, 0.10839999, 0.157084 , -0.18248999, 0.057178 , -0.07647002, 0.345651 , -0.219916 , 0.10541004, -0.291718 , 0.00305998, -0.36931998, -0.53912 , -0.24551001, -0.37141 , -0.33017 , -0.08576 , 0.12420002, 0.46991038, -0.2155 , -0.413629 , 0.030652 , -0.354111 , 0.19298099, 0.11645301, 0.155269 , 0.03384 , -0.06288001, 0.296528 , 0.396857 , 0.426261 , 0.220081 , 0.40298 , 0.17123501, 0.54055 , -0.11355601, -0.104578 , -0.01269001, -0.058513 , 0.018726 , 0.04906 , -0.28237998, 0.24926771, 0.46070004, 0.10649997, 0.484271 , -0.15225999, -0.278597 , -0.02588999, -0.49642998, -0.19471999, 0.27817 , -0.30317 , -0.04342 , -0.366697 , 0.44667 , 0.23372003, -0.28825998, 0.533203 , 0.09583002, 0.32611 , 0.221396 , 0.05684 , -0.0770785 , 0.137215 , 0.13080001, -0.761398 , -0.16238001, -0.068278 , -0.21451339, -0.62956 , 0.05763 , 0.03915 , 0.43725 , -0.05915001, -0.136398 , -0.83176005, 0.26149 , 0.32271 , 0.01355 , 0.32392102, -0.13203001, 0.251102 , -0.074752 , -0.07941997, 0.19748001, -0.02922 , 0.17733002, 0.23126 , -0.206564 , -0.48092002, -0.201221 , -0.07593001, -0.06981 , 0.08336499, 0.47656402, -0.075855 , -0.20168 , 0.33757 , 0.23384999, -0.040942 , 0.05715001, 0.359825 , -0.115785 , 0.021327 , -0.25541002, 0.16334 , 0.02236 , -0.195446 , 0.48813 , 0.31678998, 0.06573 , -0.65261 , -0.595672 , -0.08459 , 0.25744 , -0.10958001, -0.51846004, -0.142442 , 0.13389993, 0.038775 , -0.19788 , 0.27016 , 0.484097 , 0.011465 , 0.73745203, 0.02928001, -0.18780899, -0.26448005, -0.04336998, -0.28222 , 0.05045 , 0.08398899, -0.03925002, 0.08612999, -0.613462 , 0.55199003, -0.043593 , -0.11263999, 0.77089 , -0.00496998, 0.04464599, -0.2003223 , -0.13155997, -0.02552 , 0.42652598, -0.11762002, -0.23923 , -0.02312002, -0.50487995, 0.070542 , -0.279496 , -0.31031 , -0.48132098, -0.1977226 , 0.31511 , -0.195957 , 0.51311 , -0.64466 , 0.051654 , -0.55724 , -0.06236 , -0.18817002, -0.171075 , 0.19591999, 0.33167 , -0.260391 , 0.184225 , -0.53195 , -0.48722997, -0.27334002, -0.372963 , 0.0261811 , 0.021473 , 0.25916997, 0.1477536 , -0.19028401, -0.06145999, -0.56020004, -0.01161 , 0.26310998, -0.019664 , -0.027667 , 0.273624 , 0.09263998, -0.16176999, -0.272811 , -0.22291899, 0.37010002, 0.5811 , -0.09797001, -0.016622 , 0.18548 , 0.129638 , -0.28609002, 0.49303 , 0.11981001, 0.31896108, 0.42901 , 0.476502 , -0.105783 , -0.15798 , 0.68056 , -0.31356996, 0.35846 , -0.171093 , -0.085976 , -0.28327 , 0.24415004, 0.08297001, 0.016318 , -0.49722 , -0.41504 , -0.13042101, 0.02842003, -0.287583 , -0.18477401, -0.21802 , 0.36571997, 0.41406 , 0.01135001, -0.126876 , -0.01861 , -0.21993999, 0.38343003, 0.43270004, -0.58523 , -0.38494998, -0.22890002, 0.58457005, -0.23916 , -0.422687 , 0.57535 , 0.306636 , -0.13894999, 0.191552 , 0.318658 , 0.18807 , -0.2064195 , 0.375004 , -0.42850298, 0.03937 , 0.27321 , 0.04587999, -0.30842 , -0.27228 , -0.08222002, 0.61183 , 0.032794 , 0.834291 , 0.25109 , 0.13105 , -0.414862 , -0.10979015, -0.09489 , 0.06716999, 0.59828097, 0.13943201, -0.34061003, 0.06056994, -0.09411997, 0.19154501, 0.43928 , 0.52074003, 0.22537 , 0.181705 , 0.28191 , -0.18013999, -0.533239 , -0.34465098, -0.04374999, -0.358177 , -0.05912 , 0.03245001, 0.11202002, 0.421092 , 0.4599 , 0.46178 , 0.25935 ], dtype=float32)

*#visualization of distances across all words. Currently not required.*tsne = TSNE(n\_components=2, random\_state=0)words = list(embeddings\_dict.keys())vectors = [embeddings\_dict[word] **for** word **in** words]Y = tsne.fit\_transform(vectors[:1000])plt.scatter(Y[:, 0], Y[:, 1])**for** label, x, y **in** zip(words, Y[:, 0], Y[:, 1]): plt.annotate(label, xy=(x, y), xytext=(0, 0), textcoords="offset points")plt.show()



*B. Base Model Iterations*

It should be noted that all attempts were made on Simple Linear Regression for Positive competence

**Iteration 1:**

Take all words in theme\_bow where frequency>1

Problem: many similarity\_measure values=1 since theme\_bow is 1000+words

**Iteration 2:**

Take top 10%ile of theme\_bow

Problem: No strength in correlation

**Iteration 3:**

Remove stopwords from theme\_bow for more meaningful bag of words

Problem: Still no significant improvement in correlation

**Iteration 4:**

theme\_bow and similarity measure applied on Noun POS tagging

Problem: near-zero correlation when removing employee reviews

**Iteration 5:**

Adj + noun +verb pos tag. If adj\_pos is empt>use noun\_pos> if still empty, use verb\_pos. In progress

More possibilities that can be explored:

1. Change GloVe source to Twitter
2. Change dimensionality of GloVe
3. Increase/Decrease neighbourhood size
4. Look for other arbitrary hyperparameters in use.
5. Other approaches to identify key words in a text(tf-Idf)

*5.2 Approach 2 – Analysis*

Since without the previous approach we could not converge to anything related to dimensions of trust as we had identified, we approached the problem with a different data set.

From the data that we had obtained from Talkwalker, we realized that the data was very noisy, and even after much cleaning and processing we could not find much information in it. We also know that LDA works much better on a longer text format and posts from twitter and Instagram certainly lacked it.

Reddit data was hence selected to try our approach. We were given a tableau dashboard which has data from Reddit posts for US bank. Reddit data was much cleaner in comparison to data from talkwalker in terms of noise and had much more sensible content in terms of conversations related to bank. However, the data was over 700, 000 rows and was computationally heavy for python to process.

We used NLTK libraries for basic preprocessing (cleaning, tokenizing, stopwords removal and lemmatization) we further used spacy to find parts of speech and took verbs, nouns and adjectives into consideration.

We further used LDA topic modelling to find topics in the Reddit data. The output from LDA using Reddit data converged to certain topics. However, it is still quite difficult to find trust. We can converge to topics related to banking, credit cards, home loans, etc. Using sentiment analysis and sentiment score could be given to topics.



# Conclusion and Recommendations

We approached this problem as a research project to see whether something as imperceptible as trust could be measured using NLP-based machine learning models. We came across many challenges on our way. A more evidence-based approach to trust can help identify focus areas in alignment with the business understanding of trust.

We had to score our posts; in case we want to approach this problem as supervised learning.

The data we had significant levels of noise, mostly containing financial news and ads. We find very few people talking about banks on social media if they are not complaining. Even after cleaning the data, the meaningful posts about the chosen idea of trust were sparse in comparison to other content.

With Reddit data, we were able to solve certain challenges as the data was much cleaner in a way that we had data that made more sense. However, since It was over 700,000 rows and each had a whole Reddit post, it became computationally heavy.

With LDA output from Reddit, we can see that it was able to distinguish the data into certain directions, but it still doesn’t converge to something trust-related.

Recommendations:

* Data quality and source are key determinants in model accuracy and interpretability.
* Computational complexity of preprocessing unstructured data can be reduced by using advanced techniques (Big data/ parallelization of data pipeline).
* Action-based trust dimensions like competence need to be explored more to use social media data effectively.
* Targeted, topic-based identification and labeling of trust attributes using Approach B followed by the data model pipeline designed in Approach A can add more explainability to trust quantification and thereby facilitate trust scoring.