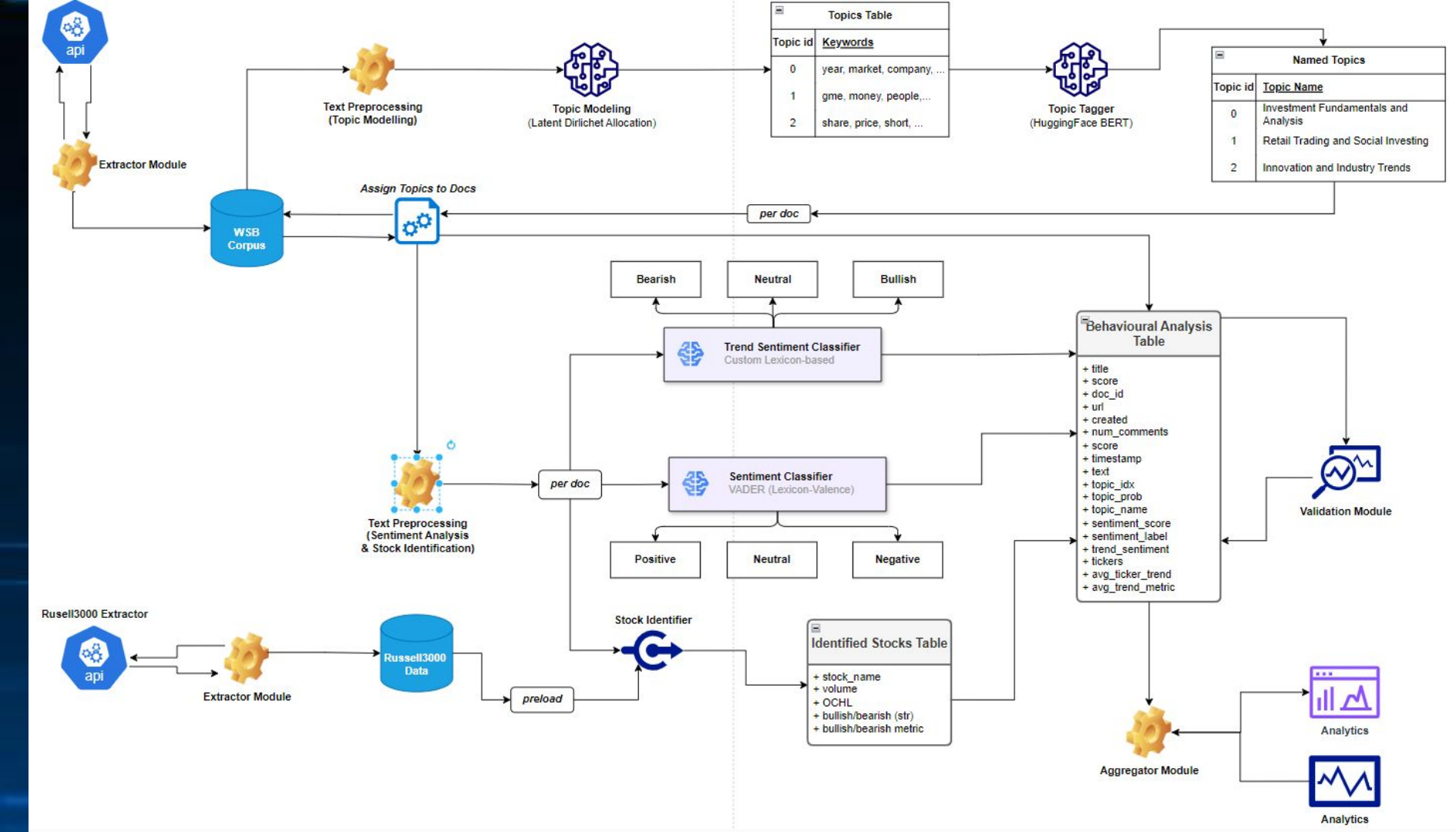
Behavioural Analysis on r/WallstreetBets

Jonathan Gonzalez
Cedric Lam
Hair Albeiro Parra Barrera


api
Extractor Module
WSB Corpus
Text Preprocessing (Topic Modelling)
Topic Modeling (Latent Dirlichet Allocation)
Topics Table
Topic id
Keywords
0
year, market, company, ...
1
gme, money, people,...
2
share, price, short, ...
Topic Tagger (HuggingFace BERT)
Named Topics
Topic id
Topic Name
0
Investment Fundamentals and Analysis
1
Retail Trading and Social Investing
2
Innovation and Industry Trends
Assign Topics to Docs
per doc
Bearish
Neutral
Bullish
Trend Sentiment Classifier
Custom Lexicon-based
Behavioural Analysis Table
+ title
+ score
+ doc_id
+ url
+ created
+ num_comments
+ score
+ timestamp
+ text
+ topic_idx
+ topic_prob
+ topic_name
+ sentiment_score
+ sentiment_label
+ trend_sentiment
+ tickers
+ avg_ticker_trend
+ avg_trend_metric
Validation Module
Text Preprocessing (Sentiment Analysis & Stock Identification)
per doc
Sentiment Classifier
VADER (Lexicon-Valence)
Positive
Neutral
Negative
Russell3000 Extractor
api
Extractor Module
Russell3000 Data
preload
Stock Identifier
Identified Stocks Table
+ stock_name
+ volume
+ OCHL
+ bullish/bearish (str)
+ bullish/bearish metric
Aggregator Module
Analytics
Analytics

Index

Introduction

Idea: Analyze the behavior and psychology of investors on r/wallstreetbets.

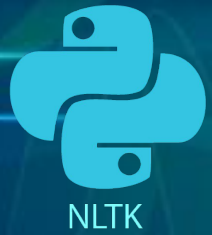
Introduction

Idea: Analyze the behavior and psychology of investors on r/wallstreetbets.

- **Methods:** NLP for behavioral analysis, topic clustering, sentiment analysis, trend analysis, text mining, data analytics, cloud computing.
- **Dataset:** Textual data from posts and comments + financial stock data.
- **Novelty:** In-depth analysis of investor psychology in a uniquely speculative and informal setting, including flagging of potentially new meme stocks.
- **Expected Output:**
  - Insights into the behavior of retail investors in hype-driven markets.
  - Model that flags potential meme stocks based on subreddit activity patterns.

Tech Stack

TensorFlow

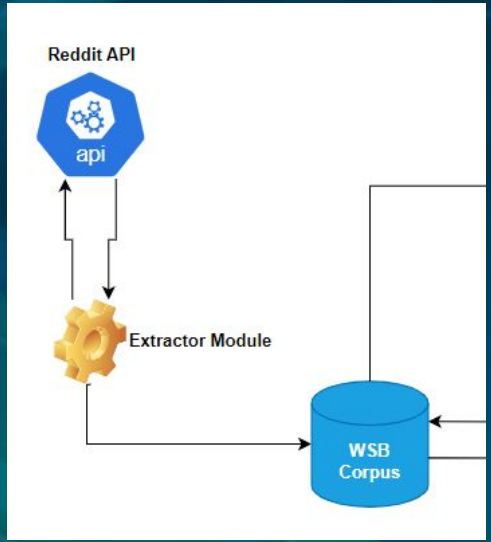


Google Cloud

Architecture

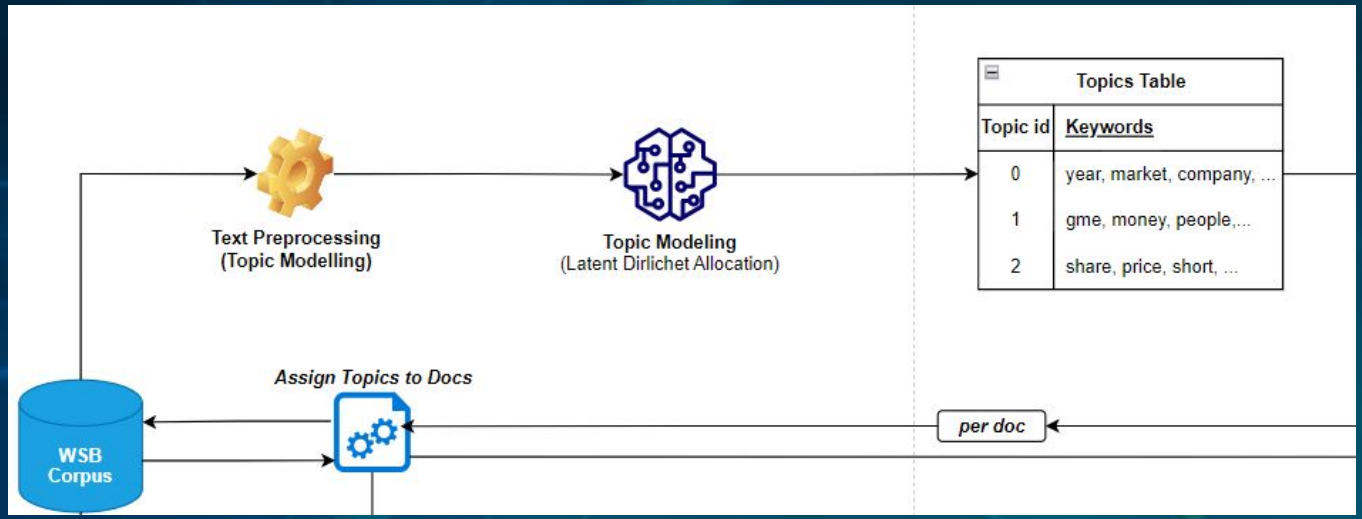
1. Reddit Extraction

- Extractor module queries the reddit API
- Data gets stored under a local table WSB Corpus
- **Improvements:**
  - Database indexing



2. Topic Modelling

- Corpus: The Wallstreetbets Subreddit recent* posts (title + body)
- Custom text preprocessing using NLTK + Spacy + Regex
- Topic modelling using **Latent Dirichlet Allocation**



Topic Modeling

Preprocessing

- Hyperlink Removal
- Emoji Handling
- Tokenization and Lemmatization
- Noise Reduction
- N-Gram Formation
- Special Char Removal

https://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf

2. Topic Modelling: Latent Dirichlet Allocation

3. Latent Dirichlet allocation

Latent Dirichlet allocation (LDA) is a generative probabilistic model of a corpus. The basic idea is that documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words.[1]

LDA assumes the following generative process for each document $\mathbf{w}$ in a corpus D:

1. Choose $N \sim \text{Poisson}(\xi)$.
2. Choose $\theta \sim \text{Dir}(\alpha)$.
3. For each of the N words w_n:
   (a) Choose a topic $z_n \sim \text{Multinomial}(\theta)$.
   (b) Choose a word w_n from $p(w_n \mid z_n, \beta)$, a multinomial probability conditioned on the topic z_n.

Src: https://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf

https://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf

2. Topic Modelling: Latent Dirichlet Allocation

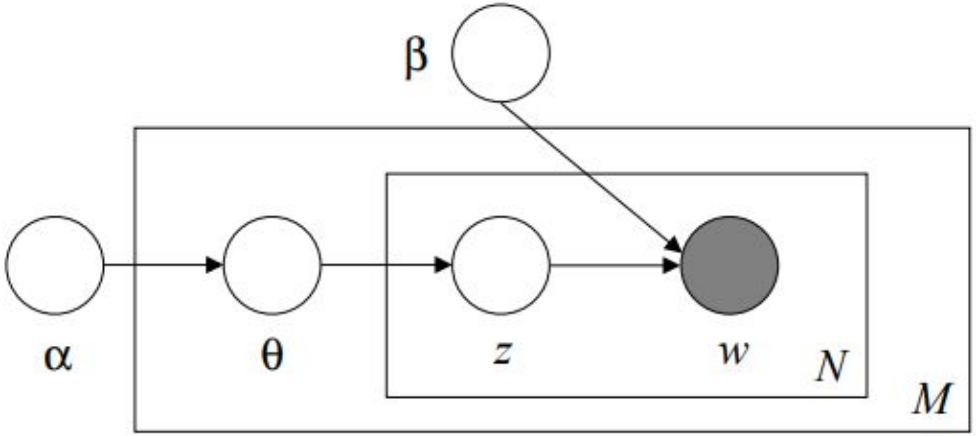


Figure 1: Graphical model representation of LDA. The boxes are "plates" representing replicates. The outer plate represents documents, while the inner plate represents the repeated choice of topics and words within a document.

Src: https://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf

https://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf

2. Topic Modelling: Latent Dirichlet Allocation

```python
# Import the necessary libraries
from src.topic_modelling import train_lda_model

# Create a dictionary with the parameters used in the LDA model
lda_params = {
    'num_topics': 4,                # The number of requested latent topics to be extracted from the training corpus
    'update_every': 1,              # Number of documents to be iteratively updated
    'chunksize': 100,                # Number of documents to be used in each training chunk
    'passes': 7,                    # Number of passes through the corpus during training
    'alpha': 'symmetric',           # Hyperparameter affecting sparsity/thickness of the topics
    'iterations': 100,              # Maximum number of iterations through the corpus when inferring the topic distribution of a corpus
}

# Train the LDA model
lda_model, corpus, id2word = train_lda_model(clean_texts, lda_params=lda_params)

# Display the topics
pprint(lda_model.print_topics())
```

Src: https://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf

https://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf

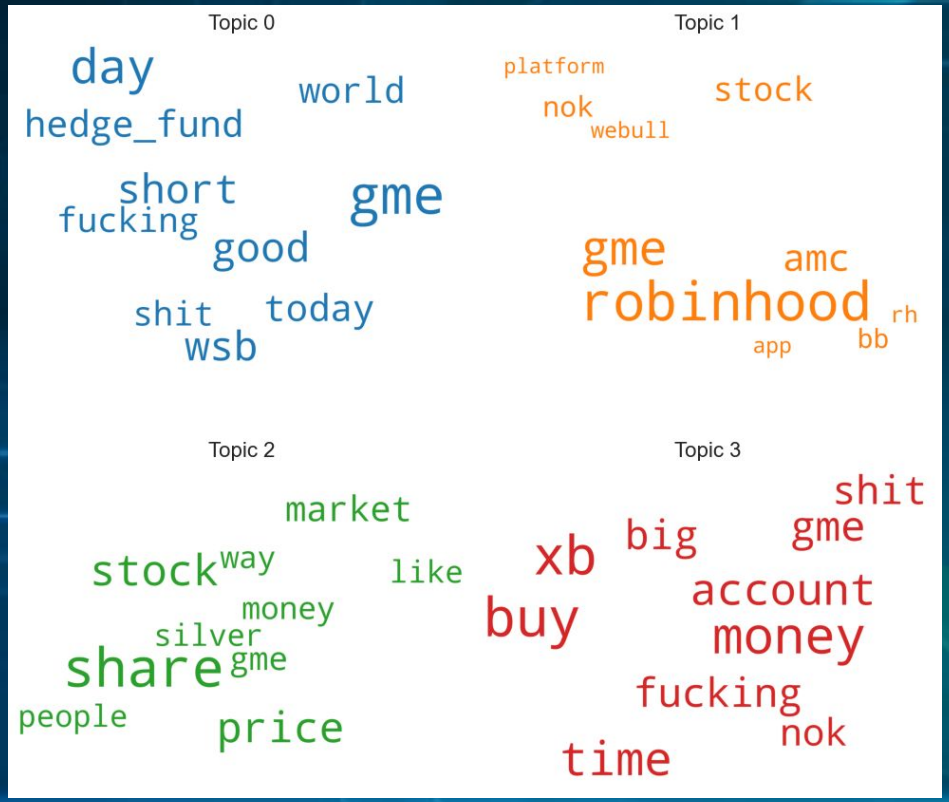
2. Topic Modelling: Latent Dirichlet Allocation

```
[   (   0,
        '0.018*"gme" + 0.014*"day" + 0.010*"good" + 0.010*"wsb" + '
        '0.010*"short" + 0.008*"today" + 0.008*"hedge_fund" + 0.007*"world" + '
        '0.007*"fucking" + 0.007*"shit"'),
    (   1,
        '0.083*"robinhood" + 0.066*"gme" + 0.037*"amc" + 0.029*"stock" + '
        '0.020*"nok" + 0.018*"bb" + 0.013*"rh" + 0.012*"webull" + 0.012*"app" '
        '+ 0.011*"platform"'),
    (   2,
        '0.037*"share" + 0.022*"stock" + 0.022*"price" + 0.014*"market" + '
        '0.012*"gme" + 0.011*"money" + 0.011*"way" + 0.011*"like" + '
        '0.011*"silver" + 0.011*"people"'),
    (   3,
        '0.017*"xb" + 0.017*"buy" + 0.016*"money" + 0.013*"time" + '
        '0.012*"account" + 0.010*"gme" + 0.010*"big" + 0.010*"fucking" + '
        '0.009*"shit" + 0.008*"nok"')]
```

Src: https://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf

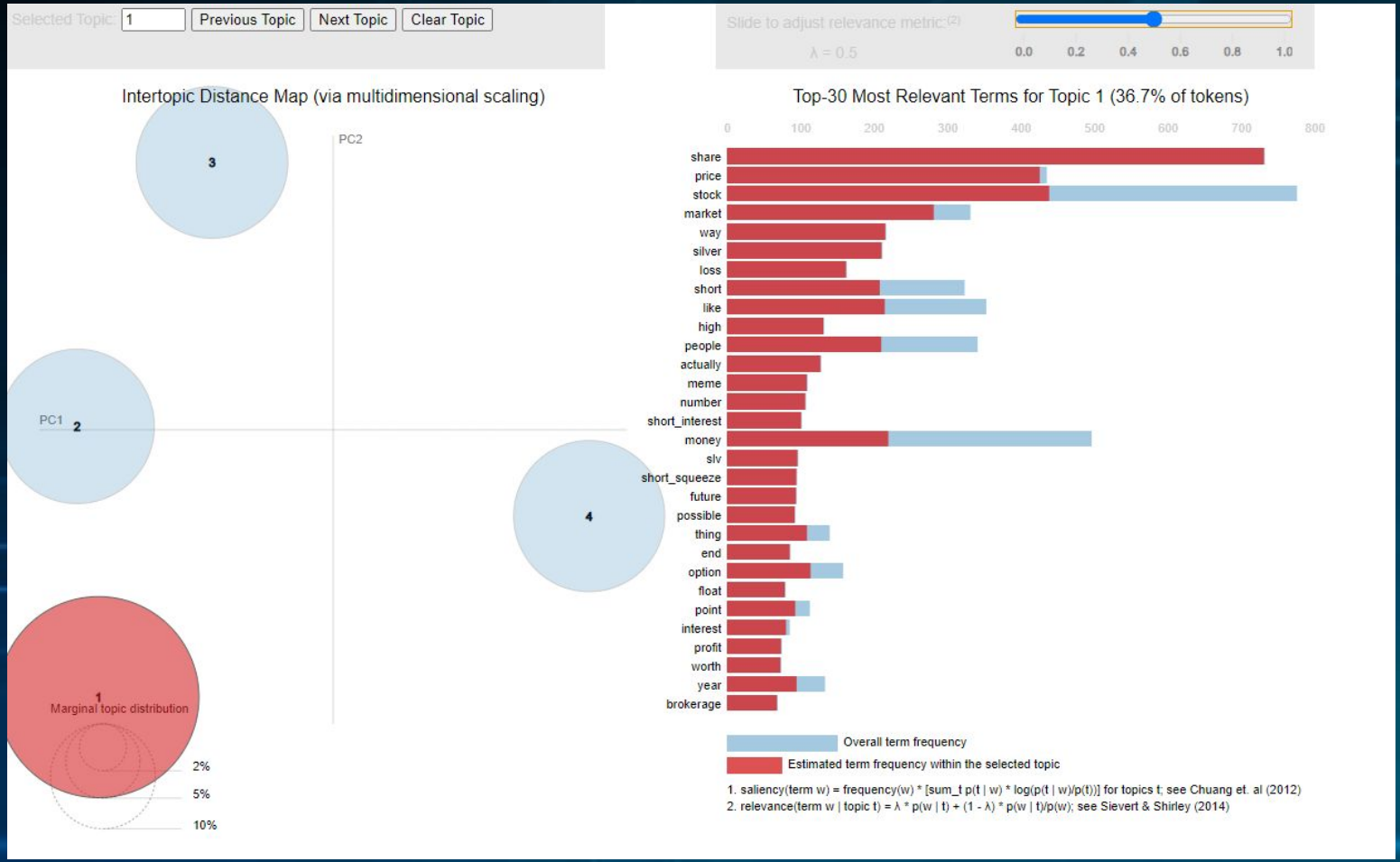
https://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf

2. Topic Modelling: Latent Dirichlet Allocation

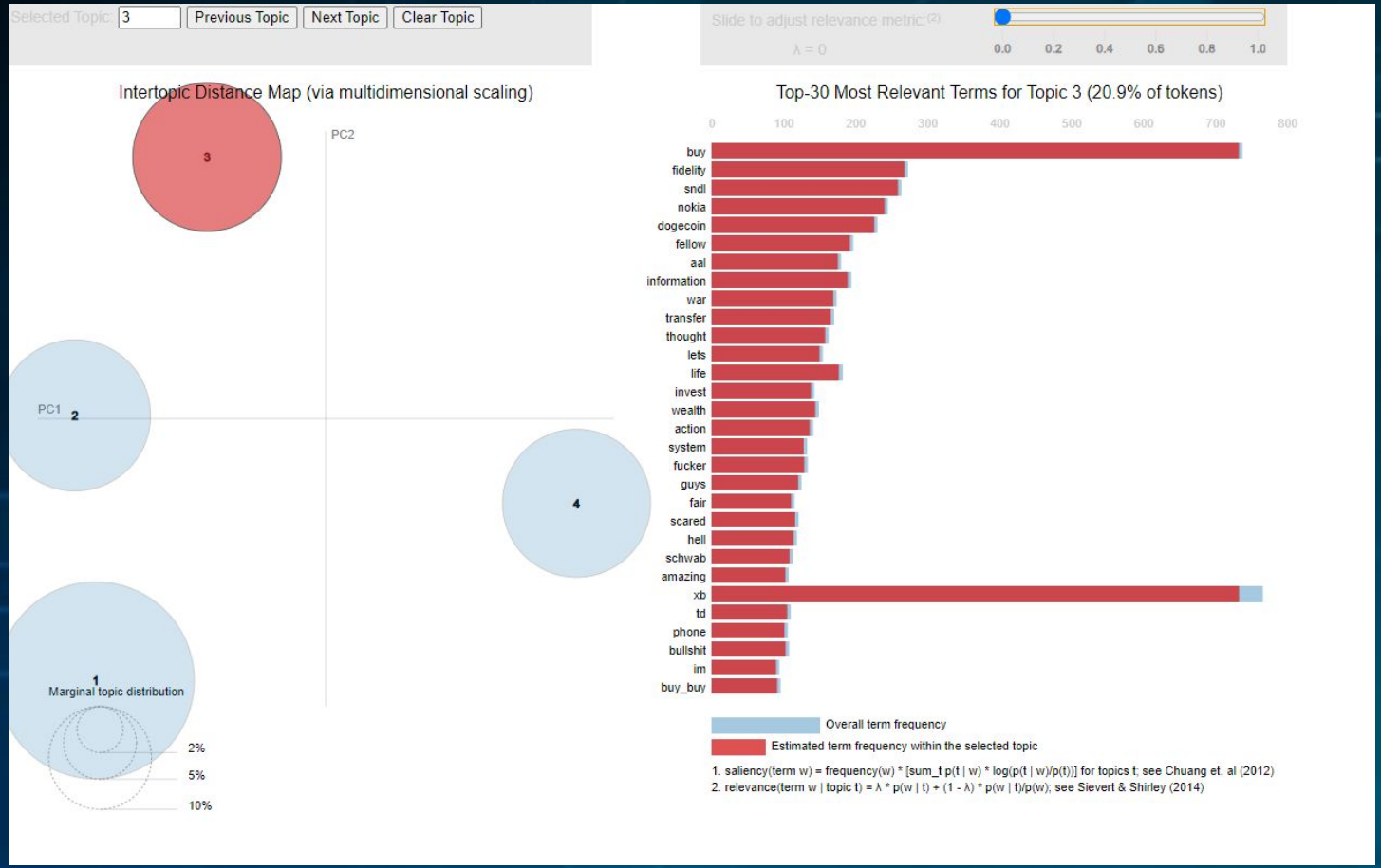


2. Topic Modelling: Latent Dirichlet Allocation

2. Topic Modelling: Latent Dirichlet Allocation

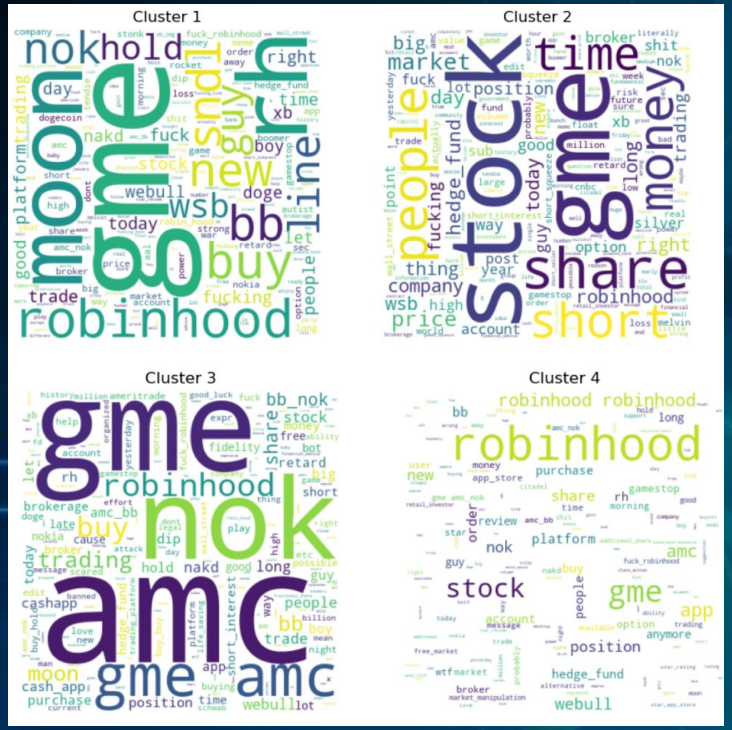


2. Topic Modelling: Latent Dirichlet Allocation

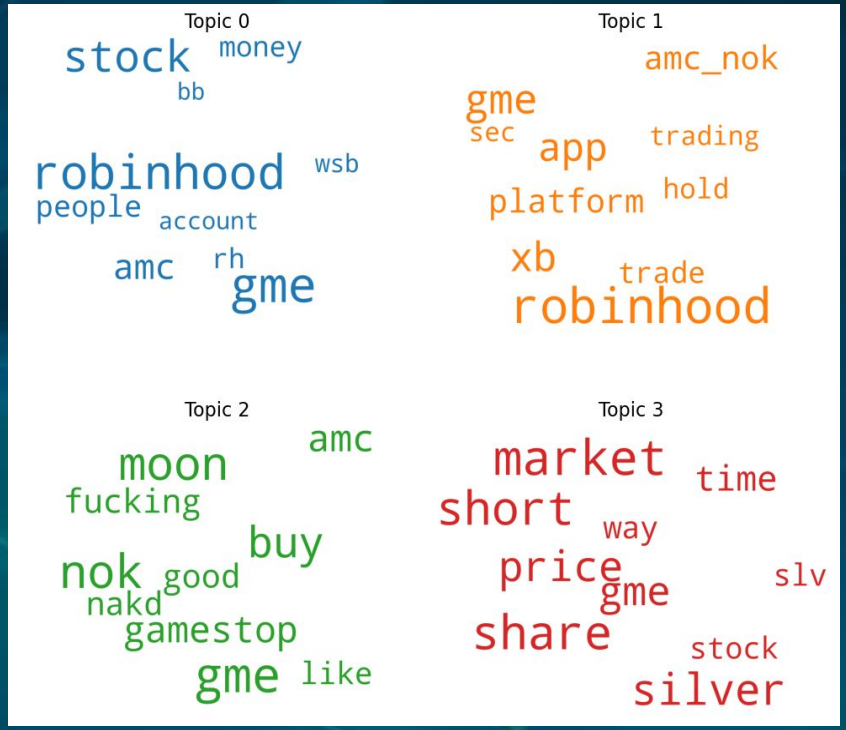


2. Topic Modelling: Baseline comparison

- **Baseline model:** TF-IDF + K-Means Clustering
- **Our Model:** LDA Topic modelling



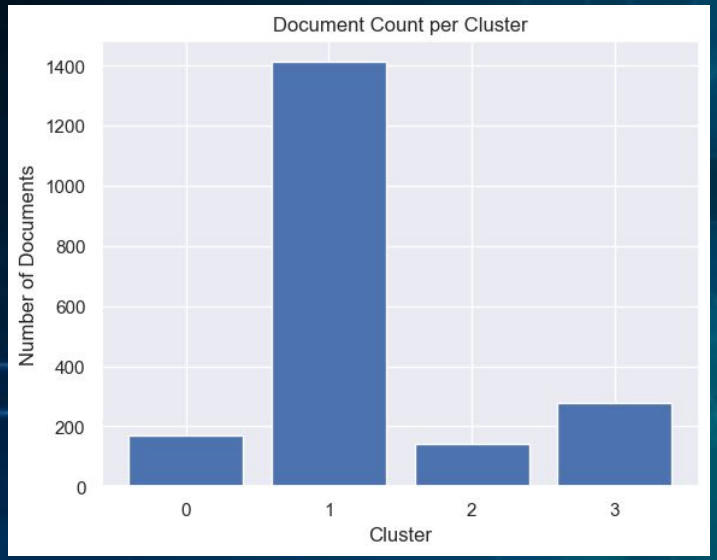
TFIDF + K-Means



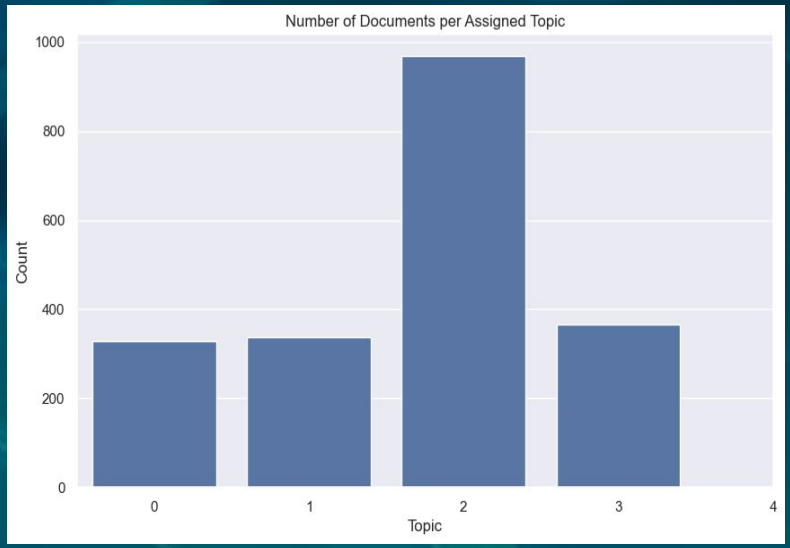
LDA Topic Modelling

2. Topic Modelling: Baseline comparison

- **Baseline model:** TF-IDF + K-Means Clustering
- **Our Model:** LDA Topic modelling



TFIDF + K-Means



LDA Topic Modelling

2. Topic Modelling: Baseline comparison

Silhouette Score: The silhouette score measures how similar an object is to its own cluster compared to other clusters. It ranges from -1 to +1, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters. The silhouette score for a single sample i is calculated as follows:

$$\text{silhouette\_sample}_i = \frac{b_i - a_i}{\max(a_i, b_i)}$$

where:

- a_i is the mean intra-cluster distance to all other points in the same cluster as sample i,
- b_i is the mean nearest-cluster distance to all points in the cluster not containing sample i.

The overall silhouette score is the average of the silhouette scores for all samples:

$$\text{Silhouette score} = \frac{1}{n} \sum_{i=1}^{n} \text{silhouette\_sample}_i$$

where n is the number of samples.

Src: Wikipedia

2. Topic Modelling: Baseline comparison

Coherence Score: The coherence score measures the interpretability of topics generated by a topic model. It calculates the pairwise cosine similarity between the top words in each topic. Higher coherence indicates that the words in the topic are more related. Given a set of top words T in a topic, the coherence score for that topic is calculated as follows:

$$\text{Coherence score} = \frac{2}{|T|(|T|-1)} \sum_{i \neq j} \text{sim}(w_i, w_j)$$

where:

- $|T|$ is the number of top words in the topic,
- w_i and w_j are the ith and jth top words in the topic,
- $\text{sim}(w_i, w_j)$ is the cosine similarity between words w_i and w_j.

The overall coherence score for a topic model is the average coherence score across all topics.

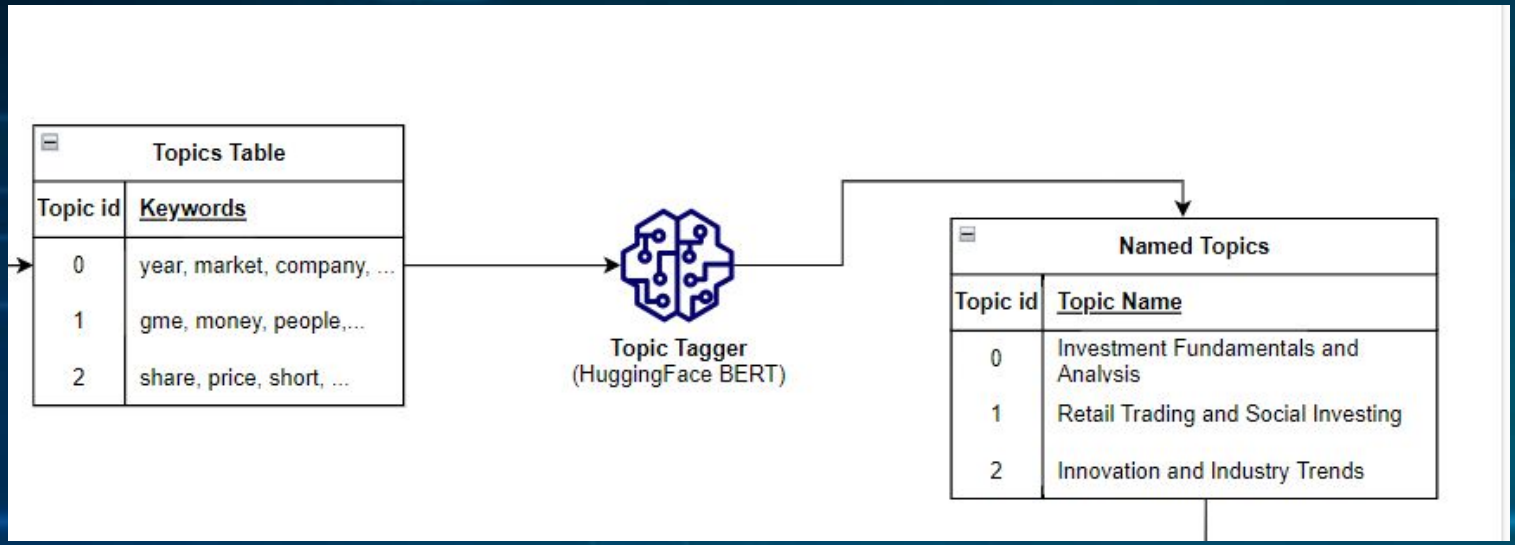
Src: Relativity One

2. Topic Modelling: Baseline comparison

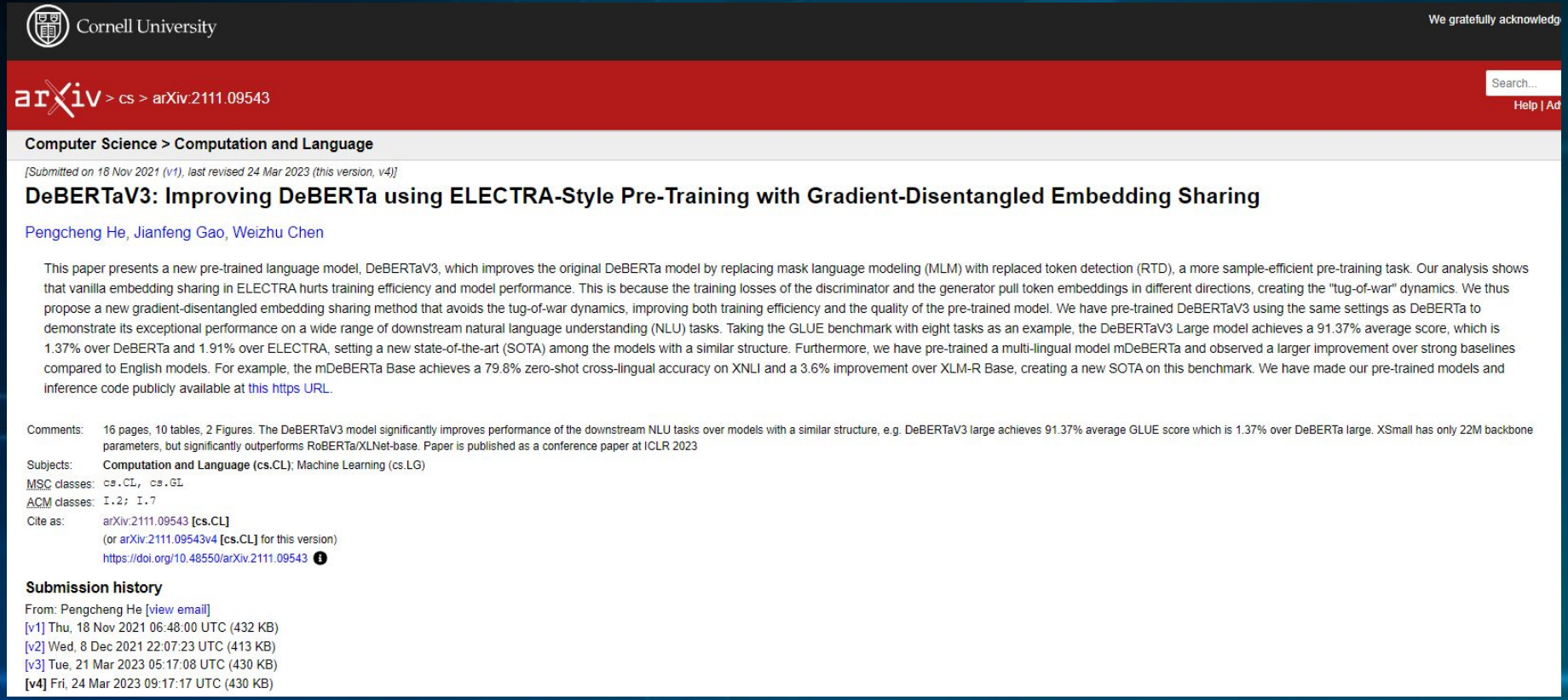
- **Sample:** 2000 texts from fixed dataset.
- **Metrics:**
  - **Silhouette Score:**
    - **TFIDF + K-Means:** ~0.01318
    - **LDA Topic Modelling:** ~0.56020 → More consistent clusters
  - **Coherence Metric:**
    - **TFIDF + K-Means:** ~0.001613
    - **LDA Topic Modelling:** ~0.36249 → More interpretable clusters

3. Topic Tagger

- **Input:** Topic mixture list of words for each topic.
- **Classification Model:** Pretrained zero-shot LLM classifier via HuggingFace interface:
  - [MoritzLaurer/DeBERTa-v3-base-mnli-fever-anli]
  - Preselected topic labels



3. Topic Tagger



Cornell University

We gratefully acknowledg

arXiv > cs > arXiv:2111.09543

Search...

Help | Ad

Computer Science > Computation and Language

[Submitted on 18 Nov 2021 (v1), last revised 24 Mar 2023 (this version, v4)]

DeBERTaV3: Improving DeBERTa using ELECTRA-Style Pre-Training with Gradient-Disentangled Embedding Sharing

Pengcheng He, Jianfeng Gao, Weizhu Chen

This paper presents a new pre-trained language model, DeBERTaV3, which improves the original DeBERTa model by replacing mask language modeling (MLM) with replaced token detection (RTD), a more sample-efficient pre-training task. Our analysis shows that vanilla embedding sharing in ELECTRA hurts training efficiency and model performance. This is because the training losses of the discriminator and the generator pull token embeddings in different directions, creating the "tug-of-war" dynamics. We thus propose a new gradient-disentangled embedding sharing method that avoids the tug-of-war dynamics, improving both training efficiency and the quality of the pre-trained model. We have pre-trained DeBERTaV3 using the same settings as DeBERTa to demonstrate its exceptional performance on a wide range of downstream natural language understanding (NLU) tasks. Taking the GLUE benchmark with eight tasks as an example, the DeBERTaV3 Large model achieves a 91.37% average score, which is 1.37% over DeBERTa and 1.91% over ELECTRA, setting a new state-of-the-art (SOTA) among the models with a similar structure. Furthermore, we have pre-trained a multi-lingual model mDeBERTa and observed a larger improvement over strong baselines compared to English models. For example, the mDeBERTa Base achieves a 79.8% zero-shot cross-lingual accuracy on XNLI and a 3.6% improvement over XLM-R Base, creating a new SOTA on this benchmark. We have made our pre-trained models and inference code publicly available at this https URL.

Comments: 16 pages, 10 tables, 2 Figures. The DeBERTaV3 model significantly improves performance of the downstream NLU tasks over models with a similar structure, e.g. DeBERTaV3 large achieves 91.37% average GLUE score which is 1.37% over DeBERTa large. XSmall has only 22M backbone parameters, but significantly outperforms RoBERTa/XLNet-base. Paper is published as a conference paper at ICLR 2023

Subjects: **Computation and Language (cs.CL)**; Machine Learning (cs.LG)

MSC classes: cs.CL, cs.GL

ACM classes: I.2; I.7

Cite as: arXiv:2111.09543 **[cs.CL]**

(or arXiv:2111.09543v4 **[cs.CL]** for this version)

https://doi.org/10.48550/arXiv.2111.09543

Submission history

From: Pengcheng He [view email]

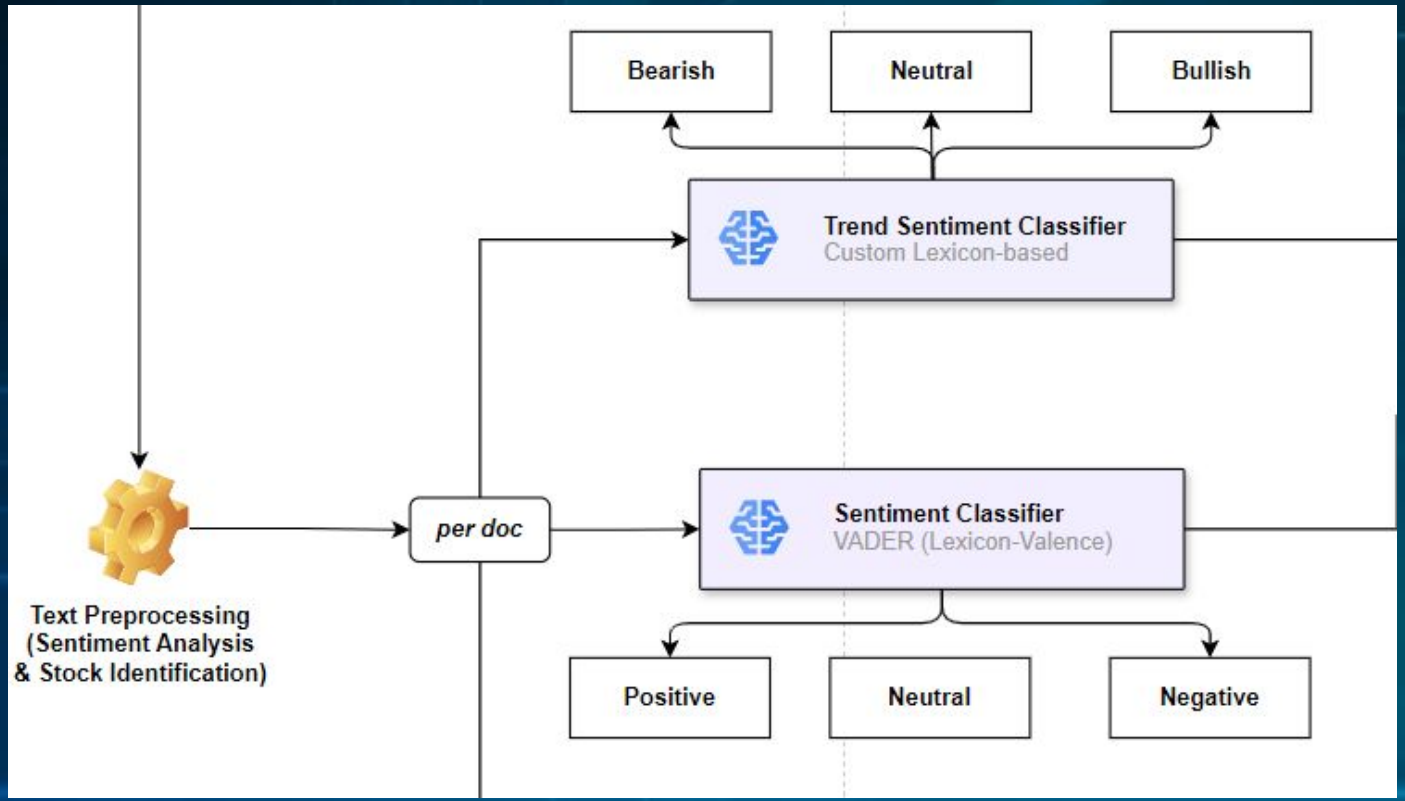
[v1] Thu, 18 Nov 2021 06:48:00 UTC (432 KB)

[v2] Wed, 8 Dec 2021 22:07:23 UTC (413 KB)

[v3] Tue, 21 Mar 2023 05:17:08 UTC (430 KB)

[v4] Fri, 24 Mar 2023 09:17:17 UTC (430 KB)

4 & 5. Sentiment & Trend Analysis



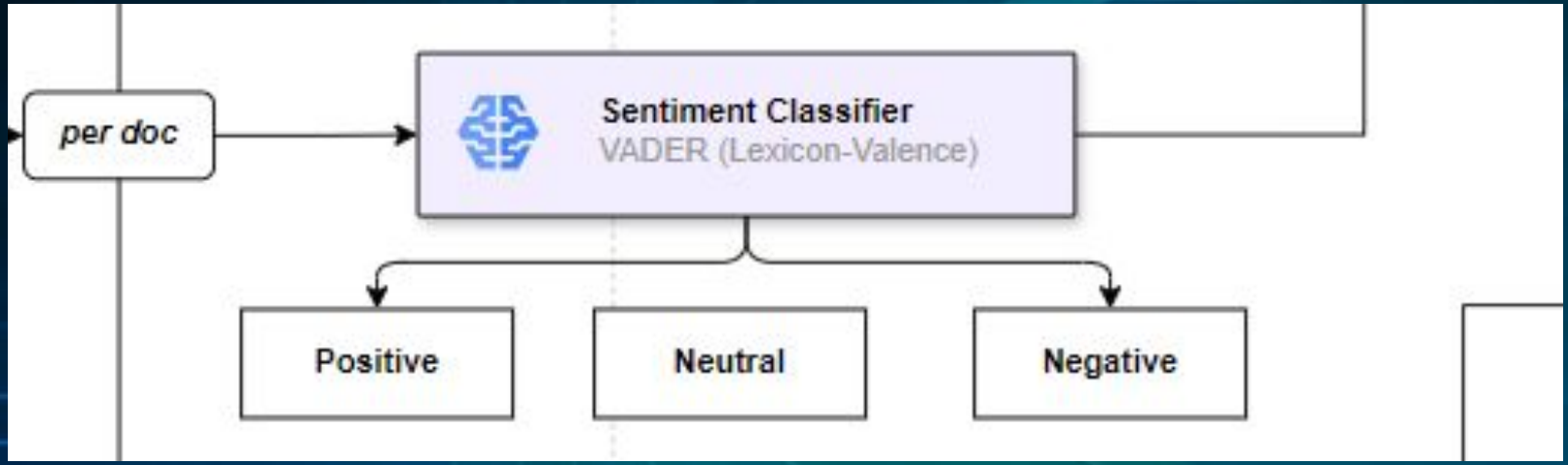
4. Sentiment and Trend

Preprocessing

- Hyperlink Removal
- Emoji Handling
- Tokenization
- Special Char Removal*

4. Sentiment Classifier

(VADER)



4. Sentiment Classifier

(VADER)

Valence Aware Dictionary for Sentiment Reasoning

Lexicon and rule-based sentiment analysis tool specifically tuned for social media.

Valence Score

A score assigned to the word under consideration by means of observation and experiences rather than pure logic.

LIWC Category	Examples	No. of Words
Positive Emotion	Love, nice, good, great	406
Negative Emotion	Hurt, ugly, sad, bad, worse	499

Table 1: Example words from two of LIWC's 76 categories. These two categories can be leveraged to construct a semantic orientation-based lexicon for sentiment analysis.

C.J. Hutto and E. Gilbert, "VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text," in *Proceedings of the International AAAI Conference on Weblogs and Social Media*, vol. 8, no. 1, 2014. [Online]. Available: https://ojs.aaai.org/index.php/ICWSM/article/view/14550/14399

Heuristics

Sentiment Lexicon:

"This movie is amazing!"

(Positive sentiment due to the word "amazing".)

Intensity Modifiers:

"This is really awesome!"

(The word "really" boosts the positive sentiment.)

Conjunctions:

"The plot was good, but the characters were unconvincing."

(The conjunction "but" shifts the sentiment.)

Punctuation Emphasis:

"I loved it!!!"

(Multiple exclamation marks increase the sentiment intensity.)

Capitalization:

“This was the WORST movie ever."

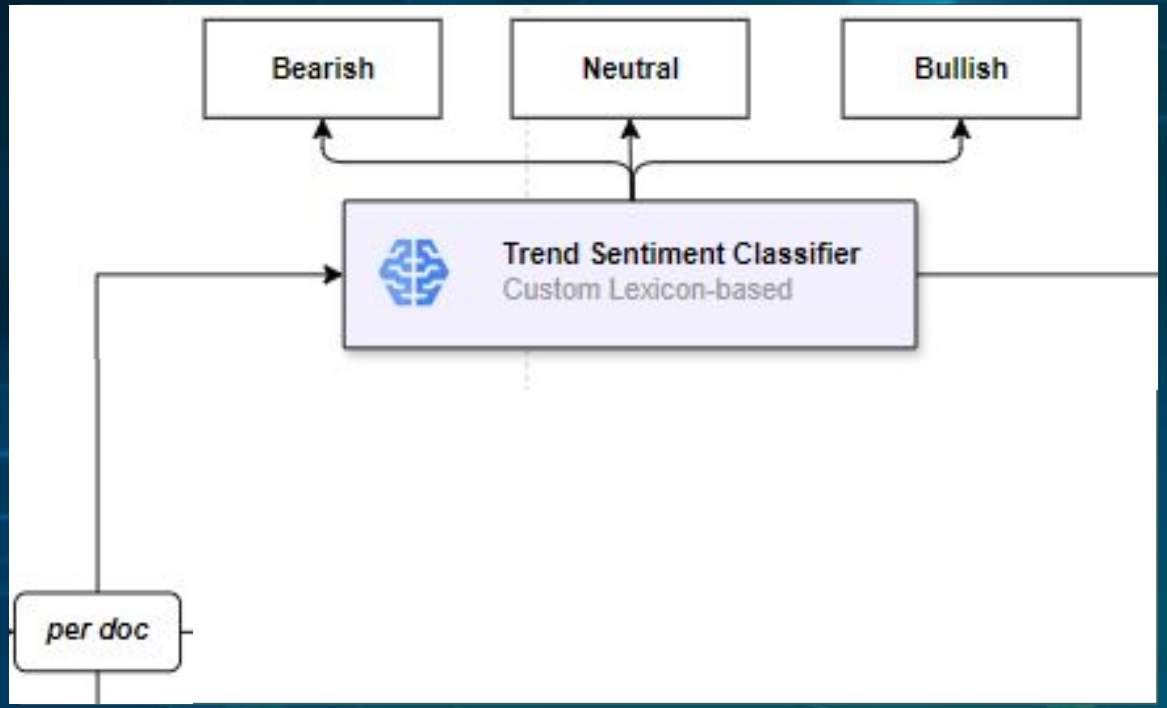
(Capitalization indicates strong negative sentiment.)

Negations:

"I didn't hate it."

(The negation "didn't" flips the sentiment to be less negative or neutral.)

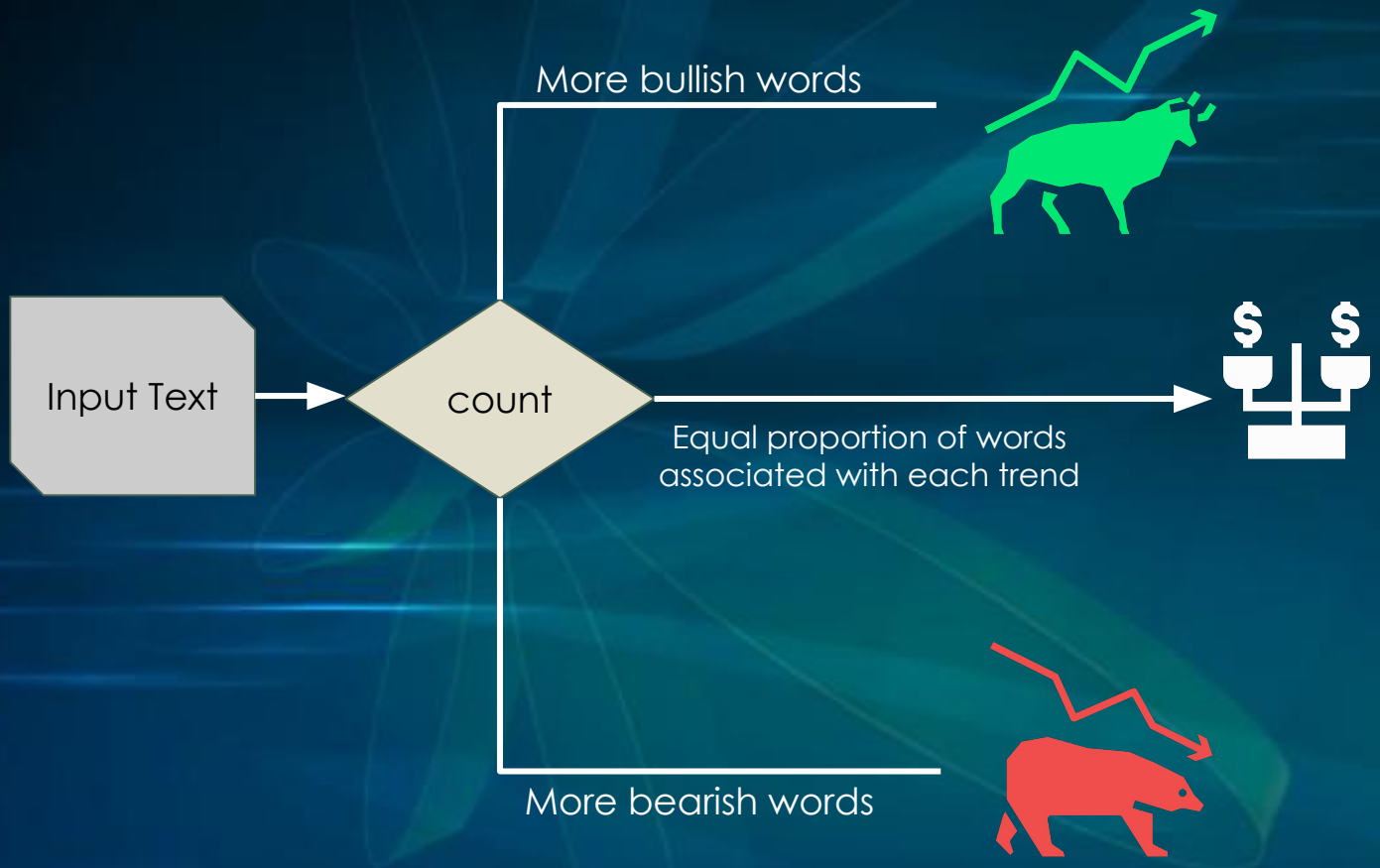
5. Market Trend Sentiment Classifier



5. Market Trend Sentiment

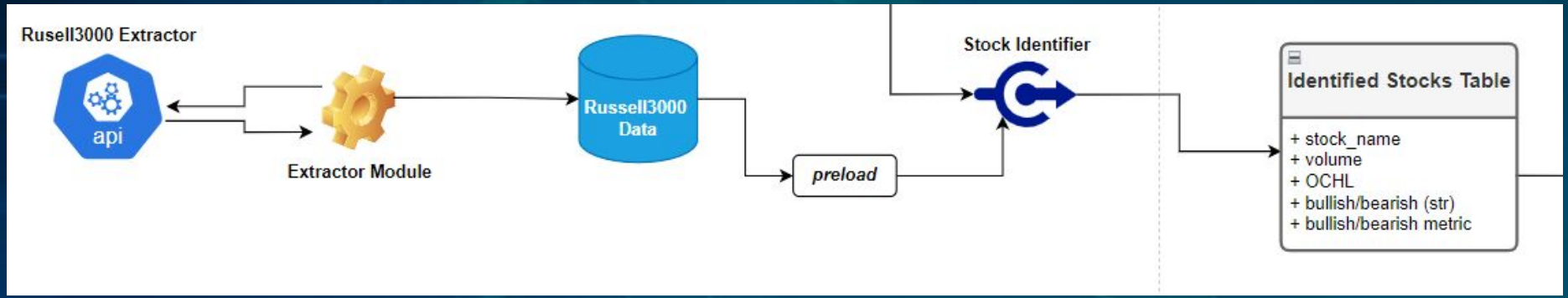
Simple yet effective way to gauge market sentiment from financial news articles, tweets, or analyst reports by analyzing the frequency of specific optimistic (bullish) or pessimistic (bearish) words.

5. Market Trend Sentiment



6. Stock Identification

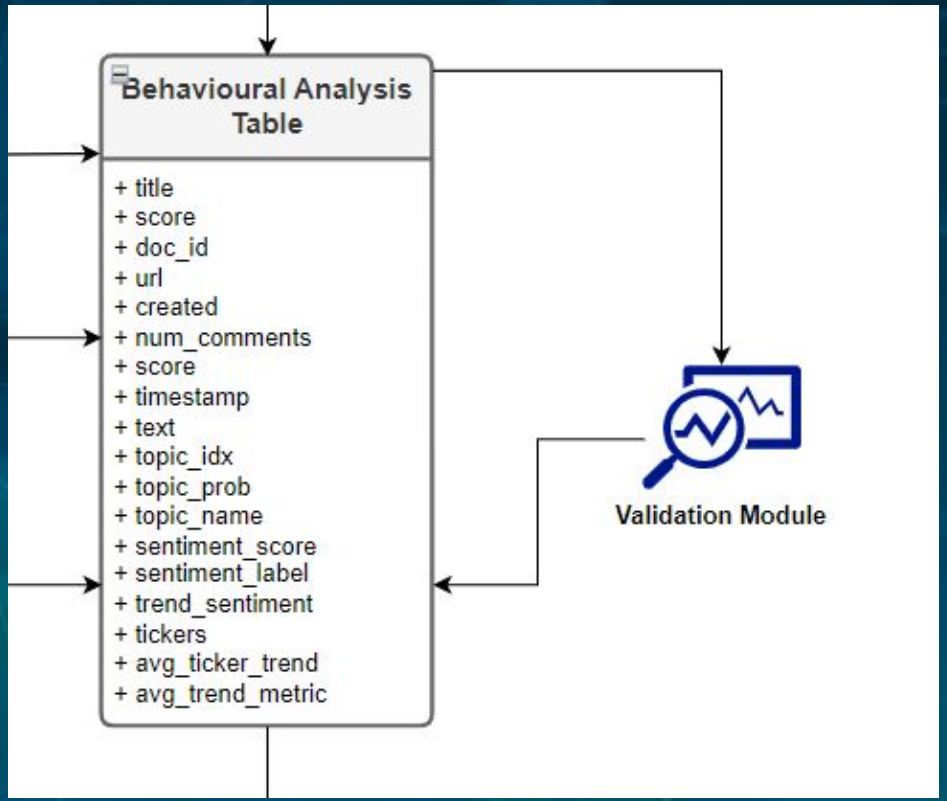
- **Input:** WSB Subreddit documents (with light to no preprocessing)
- **Output:** Identified stock tickers, weighted by market volume.
- **Methods:**
  - Russell3000 extractor script
  - Heuristic + regex-based methods
  - Yahoo finance API



7. Aggregation

- Wallstreetbets fetched API data
- Assigned topics per reddit (topic name, probability)
- Reddit sentiment score & label (negative, neutral, positive)
- Reddit trend analysis (bearish, neutral, bullish)
- Identified reddit tickers

Aside: *Validation module*

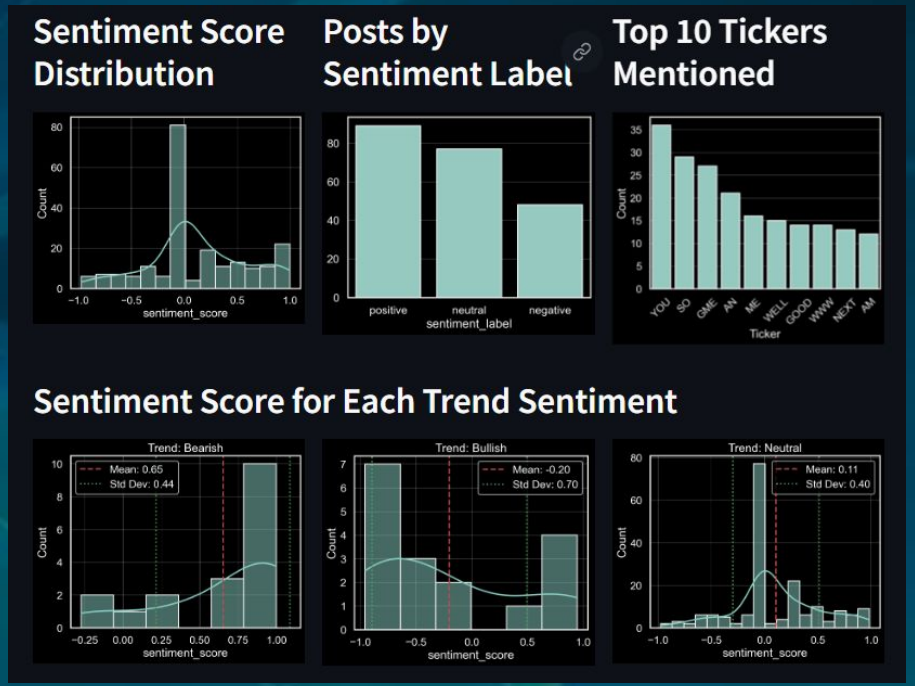


8. Dashboard & Analytics

WSB Data Analysis

WSB Analyzed Data

	title	score	url
126	I haven't clicked on r/wallstreetbets actual subreddit in years and lemme tell ya	403	https://www.reddit.com/r/w
127	Trading is like basketball	1	https://www.reddit.com/r/w
128	Further runway for oil stocks	43	https://www.reddit.com/r/w
129	CEO got $32.8 million last year to ruin Boeing. I would've done it for $10 million	9,760	https://www.barrons.com/a
130	Rate cuts or no?	199	https://i.redd.it/8ft0g527dw
131	Lilly calls?	20	https://www.reddit.com/r/w
132	Plays for The Holy Trinity Next Week	110	https://www.reddit.com/r/w
133	NVIDIA Chief Cloud Arch Speaks at UPENN	113	https://i.redd.it/bgg5w481o
134	How low can DJT (Donald Trump) stock go?	1,262	https://www.reddit.com/r/w
135	Uber is 100% going to miss earnings. Badly.	8,422	https://www.reddit.com/r/w

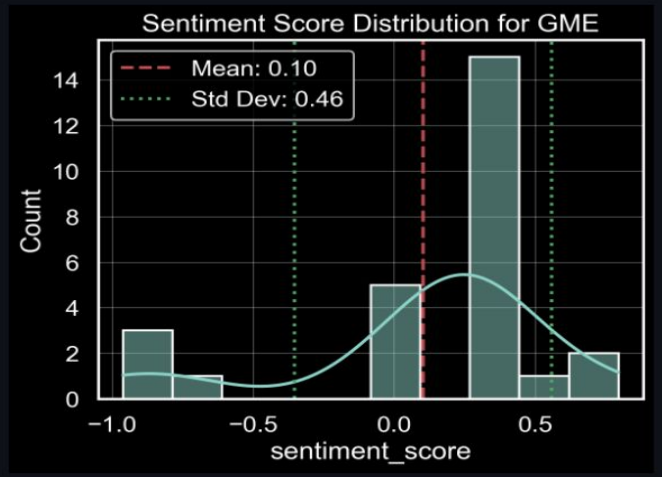


8. Dashboard & Analytics

Search Ticker for Sentiment Score Distribution

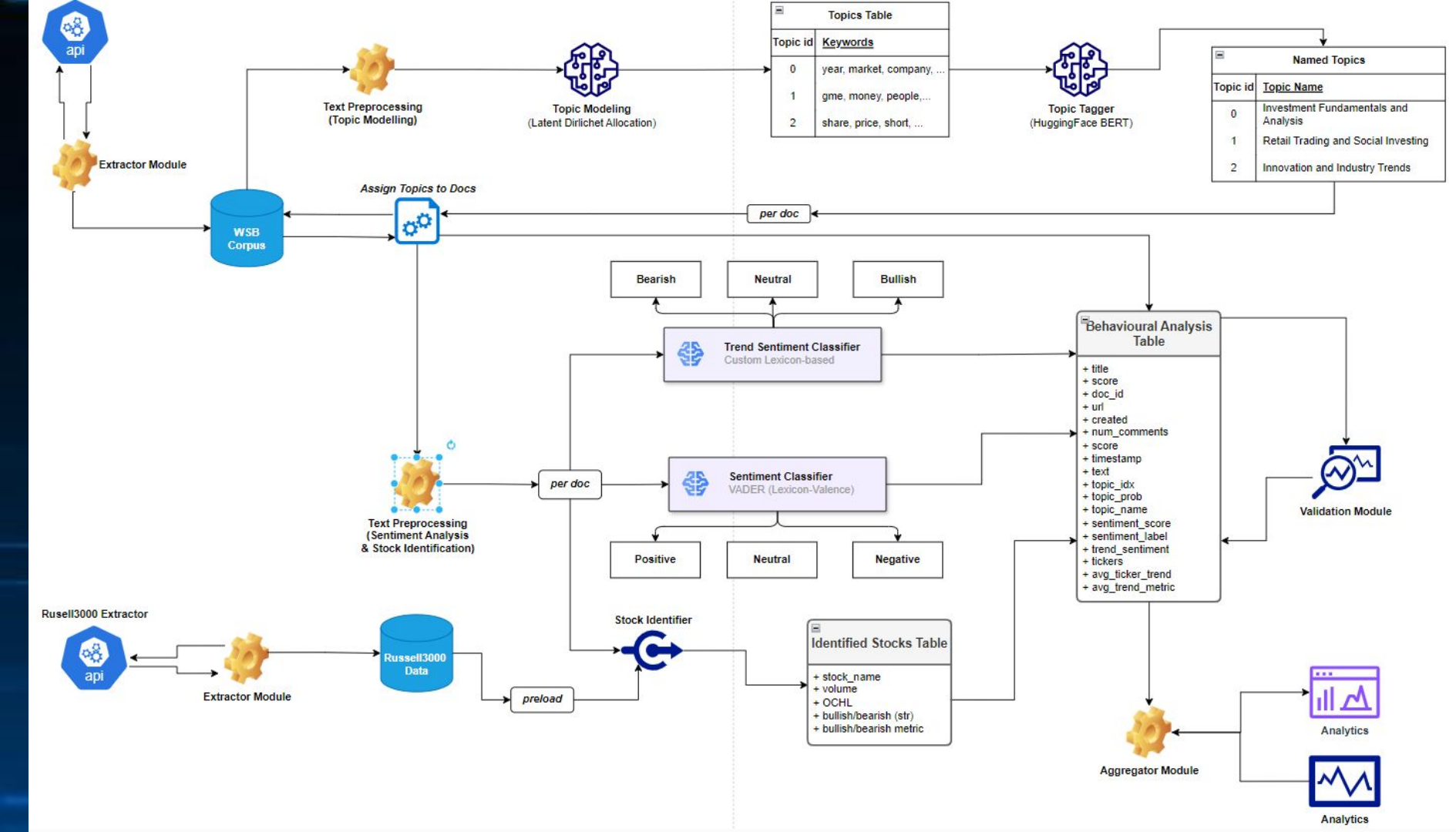
Enter a ticker to search:

GME



Sentiment Score by Topic Name

	topic_name	Avg Sentiment	std
0	Financial Growth and Valuation	0.316984	0.570779
1	Market Liquidity and Volume	0.026596	0.223454
2	Retail Investor Sentiment	0.137448	0.596498
3	Social Media Trading	0.142906	0.434057


api
Extractor Module
WSB Corpus
Text Preprocessing (Topic Modelling)
Topic Modeling (Latent Dirlichet Allocation)
Topics Table
Topic id
Keywords
0
year, market, company, ...
1
gme, money, people,...
2
share, price, short, ...
Topic Tagger (HuggingFace BERT)
Named Topics
Topic id
Topic Name
0
Investment Fundamentals and Analysis
1
Retail Trading and Social Investing
2
Innovation and Industry Trends
Assign Topics to Docs
per doc
Bearish
Neutral
Bullish
Trend Sentiment Classifier
Custom Lexicon-based
Behavioural Analysis Table
+ title
+ score
+ doc_id
+ url
+ created
+ num_comments
+ score
+ timestamp
+ text
+ topic_idx
+ topic_prob
+ topic_name
+ sentiment_score
+ sentiment_label
+ trend_sentiment
+ tickers
+ avg_ticker_trend
+ avg_trend_metric
Validation Module
Text Preprocessing (Sentiment Analysis & Stock Identification)
per doc
Sentiment Classifier
VADER (Lexicon-Valence)
Positive
Neutral
Negative
Russell3000 Extractor
api
Extractor Module
Russell3000 Data
preload
Stock Identifier
Identified Stocks Table
+ stock_name
+ volume
+ OCHL
+ bullish/bearish (str)
+ bullish/bearish metric
Aggregator Module
Analytics
Analytics

Demo

Questions?