Text Mining Project - Master's Degree in Data Science

Emotion-Focused Analysis of Stock Tweets: Challenges and Insights

Robin Smith[1] and Sergio Verga[1]

[1]Dipartimento di Informatica, Sistemistica e Comunicazione, Università degli Studi di Milano-Bicocca

January 5, 2025

Abstract

This project investigates emotion analysis in the financial domain, focusing on multi-class emotion classification and topic modeling of stock-related tweets.
For classification, various models, including Random Forest, XGBoost, and SVM, are tested on different text representations such as TF-IDF and contextualized embeddings from DistilRoBERTa and BERTweet. Hand-crafted features, involving sentiment lexicon scores and emoji encoding, are also integrated to assess their impact. In parallel, topic modeling via LDA and BERTopic is employed to uncover dominant themes in the data, exploring their relationship with emotional content.
Results highlight the challenges posed by the brevity and noise in the dataset. Traditional methods like TF-IDF perform competitively in classification, likely because they align better with the strengths of classical classifiers compared to embedding-based representations. Topic modeling, despite limited coherence, provides valuable insights into the connection between themes and emotions.
These findings underscore the complexity of analyzing stock-related tweets, offering a foundation for future refinement in both the explored tasks.

1 Introduction

The growing use of sentiment analysis in financial contexts has highlighted its value in understanding market sentiment and predicting stock movements. In particular, analyzing stock-related tweets provides a unique opportunity to capture nuanced investor emotions that can directly influence trading behavior. These tweets, often brief and diverse in expression, pose significant challenges due to the granular classification required to distinguish between a wide range of emotions.
Recent studies, such as [1] and [2], have demonstrated a strong correlation between public sentiment and stock market trends, emphasizing the practical importance of this analysis. Building on these findings, this project aims to address the complexities of multi-class emotion classification of stock-related tweets, tackling the challenges posed by linguistic variability, brevity, and the need for fine-grained categorization in a domain with substantial implications for financial decision-making.

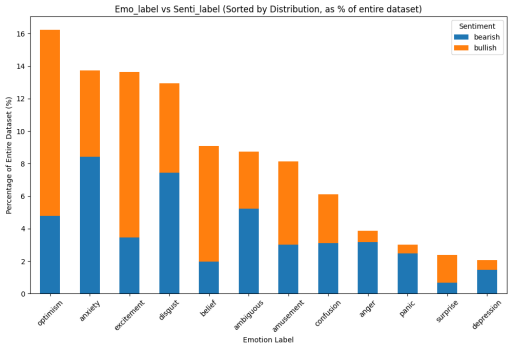
2 Dataset Exploration

The stock emotions dataset consists of 10000 total observations with 7 features, split in training (80%), validation (10%) and test (10%). It does not have missing values or duplicated rows.
The breakdown of the features is as follows.

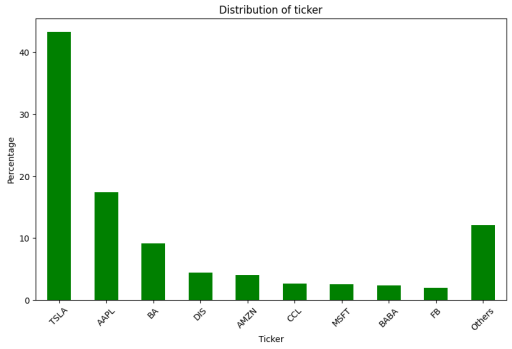
- `id`: the identifier of each row [integer].
- `date`: the day in which the post was shared [yyyy-mm-dd].
- `ticker`: the asset to which the post refers. The format is encoded, as for Tesla, which is represented by TSLA [string].
- `emo_label`: multi-class label describing which emotion fits the post [string].
- `senti_label`: binary label describing if the post is bullish or bearish. This means separating if the author believes the stock will increase or decrease in value, respectively [string].
- `original`: the original content of the post [string].

- `processed`: the processed content of the post. Even if this is available, in this project a different approach is proposed [string].

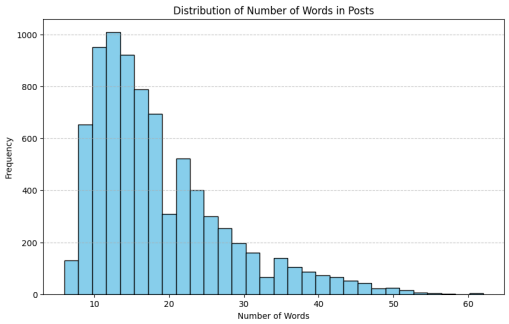
Considering the binary sentiment label (bullish/bearish) the distribution is slightly imbalanced with 54.8% bullish and 45.2% bearish.
As for emotion analysis, the multi-class label is the result of a hybrid generation, partially AI-made and partially human-made. The distribution of the labels are not balanced, with the three most common emotions being optimism (16.24%), anxiety (13.74%) and excitement (13.65%). The least common ones are panic (3.00%), surprise (2.39%) and depression (2.08%).



As shown in the distribution of the emotions, cross-visualized with the sentiment label, the emotions share a heterogeneous proportion of the two sentiment classes (bullish/bearish).
Among the 37 tickers in the training set, there is a severe imbalance in favor of Tesla, present in 43.29% of the observations.



Visualizations of temporal information do not reveal interesting insights for emotion classification or topic modeling. The majority of the posts are made on working days, the minority during Saturday and Sunday.
For each emotion, each post has an average length (defined as the space-based word count) ranging between 18.2 and 18.4. In general, the distribution of the posts length is as follows.



The distribution of space-based word counts is characterized by a minimum value of 6 words, a maximum equal to 62 words and an overall average value of 18.6 words per post.
With regard to emojis, EDA reveals that the most common is the laughing face (which occurs in approximately 10% of the training dataset). Triple and double smiling faces are, respectively, the sixth and seventh most common emoji usage.

3 Dataset cleansing

A few cleansing steps are performed before proceeding to text processing:

1. Unique ticker lists are extracted from training, test and validation sets.

2. Since the training set alone, as we would expect, has all the validation and test sets tickers mentioned in it, we consider the unique ticker list of the training set to be the complete one and sort it in alphabetical order.

3. We define, with the help of ChatGPT 4o, a dictionary mapping each ticker to the name of the company, to be further exploited.

4. We drop columns `id`, `date`, and `ticker`, redundant to our analysis. However, we keep the `processed` column as a comparison term for our pre-processing procedure.

4 Text pre-processing

All the pre-processing steps are performed on the `original` column of the dataset, which constitutes the *corpus* of this project. Each item in such a column will, in fact, be considered a document of our collection.
The pipeline we propose follows a modular design, basing on the text representations we intend to implement: it begins with the transformations necessary for the contextualized embeddings generation, subsequently incorporates

steps for (non-contextualized) embeddings computation and finally integrates the last steps for TF-IDF vectorization.
Most of these steps are performed by means of the `re` Python library, through the adoption of regular expressions and custom functions.
Tokenization is performed in different ways depending on the module (and thus the representation) in question, as described in 4.4.

4.1 Module 1

The first pre-processing module consists of:

1. Replacement of tickers with company names, according to the dictionary computed in Section 3.
2. Mapping of all the company names into the placeholder `company_name`. This is done to avoid bias arising from a company being linked to a set of emotions because of a positive (or negative) trend during the time window considered within the collection.
3. Removal of newline expressions [`\n`].
4. Mapping of different quote symbols into the standard one [`"`].
5. Removal of multiple spaces.

The output of these steps is fed to the contextualized embeddings generation algorithms.

4.2 Module 2

The second module is built on the results of the first one, by adding:

1. Meaningful multiple punctuation tokens: `multiple_exclamation`, `multiple_question`, `multiple_ellipsis` to replace, respectively, multiple exclamation points, question marks and ellipses.
2. Neutral punctuation removal, with a few cautions. [`|,"`] are always removed. [`.`] are removed when not part of decimal numbers. [`;:`] are removed when not near parentheses of any kind, possibly being in that case part of an emoticon.
3. Emoji encoding. By means of the `emoji` library, the emojis are translated into words. Here, in the case of translations consisting of multiple words, it was decided to keep underscores between them, so that the statistical or frequency-based algorithms would identify each emoji as a single token.

Note that the processing performed on the non-neutral punctuation marks stems from the intention to provide the algorithms (generating the textual representations) with meaningful tokens with respect to the intensity of emotions in the documents. Attempts were made to give a measure of intensity by indicating the number of consecutive signs (e.g., of exclamation points), but this resulted in a performance drop, probably related to the size of the vocabulary that does not allow such a granular distinction.
The result of this module is fed to the *Word2Vec* algorithm to produce non-contextualized embeddings.

4.3 Module 3

In addition to the two previous modules, text lemmatization is performed by means of the `spacy` Python library in order to prepare the documents to be fed to the TF-IDF vectorizer. The same is done for the BoW representation.
Note that:

- lemmatization is preferred to stemming because of the reduced size of the document collection, giving a more correct and complete result with negligible loss in terms of performances;
- `spacy` library is preferred to `nltk` since it automatically implements a POS tagging step which is focal to avoid ambiguities during the lemmatization procedure.

4.4 Lower casing and tokenization

Lower casing and tokenization are handled differently depending on the various text representations:

- Contextualized embedding models (4.1) are designed to process text while preserving case information. Therefore lower casing is not to be applied, while tokenization is automatically performed by the model itself.
- When implementing Word2Vec (4.2), both lower casing and tokenization are performed (in this order) via `nltk` Python library.
- When generating BoW and TF-IDF vectors (4.3), lower casing and tokenization are performed by means of the `spacy` module before lemmatizing the documents.

4.5 Stop words

Stopwords are removed in both classification task and topic modeling task using NLTK module `stopwords` or the `stop_words` argument in

the case of TF-IDF. This process is beneficial for decreasing computational cost and making the data more meaningful, especially in Bag of Words (BoW) representation (for the topic modeling task). TF-IDF is more robust to stopwords, since it tends to assign them low weights. Still, even in this case, removing them is more reasonable than simply lowering their weights.

5 Feature extraction

5.1 Textual features

Six different kinds of text representation are tested:

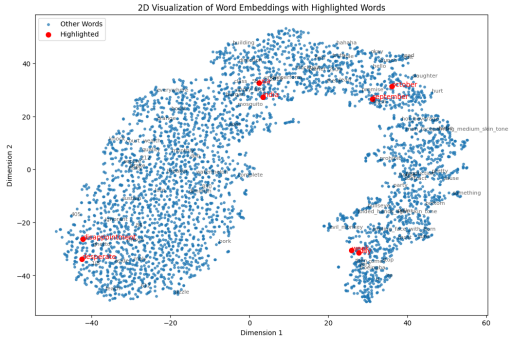
1. Bag of Words (BoW)
2. TF-IDF with unigrams (standard).
3. TF-IDF with bigrams.
4. Word2Vec embeddings.
5. Contextualized embeddings based on two pre-trained language models:
   (a) BERTweet [3].
   (b) Distil-RoBERTa fine-tuned on emotion text data (originally trained to predict 6 basic emotions plus a neutral class) [4].

Note that:

- for traditional vectorization techniques such as TF-IDF and Word2Vec, the vectorizer is fit exclusively on the training set. The fitted vectorizer is then applied to the validation and test sets to ensure no data leakage and consistency across datasets;
- as for contextualized embeddings derived from pre-trained language models, they are generated directly by applying the model independently to the training, validation, and test sets. Since these models are pre-trained, no additional fitting is required.

Both TF-IDF with unigrams and with bigrams are trained with 10000 `max_features`, limitating the dimension of the output for computational reasons. The standard TF-IDF only uses 7291 features.
As for Word2Vec, two hyperparameters are explored: `vector_size` and `window`. These values determine the dimension of the embedding representation and the context used in the model training. The choice is made by maximizing the accuracy of the Random Forest on the validation set.

t-distributed Stochastic Neighbor Embedding (t-SNE) is used to visulize in two dimensions the embedding, highlighting some particular words to display the fact that semantically close words are close in terms of distance in the embedding. This representation is selected for its property of preserving local structures.



As shown from the t-SNE plot, the following pairs of words (semantically close to each other) remain close in the representation:

- week and day
- usa and india
- october and september
- disappointment and desperate

5.2 Hand-crafted features

The following set of hand-crafted features is computed, with the aim of capturing meaningful sentiment and emotion related information.

Text length and uppercase ratio

Text length is computed as the total number of tokens in a document.
Uppercase ratio is calculated before lower casing and is defined as the number of capitalized terms relative to the text length. Words composed of a single letter are excluded from the count, in order to avoid particular cases, such as the *A* article at the beginning of sentences or the *I* pronoun.
Both for text length and uppercase ratio, tokenization is done basing on the `nltk` library on the text without emojis.

Bing Liu's Lexicon based features

Agreement Score and Polar Word Occurrence based on Bing Liu's Sentiment Lexicon [5] are computed as defined in [6].
The Agreement Score (AS) measures the difference between positive (T_p) and negative (T_n) terms, normalizing it with respect to the total

number of positive and negative words. Formally:

$$\text{AS} = \begin{cases} 0, & \text{if } T_\text{p} + T_\text{n} = 0 \\ 1 - \sqrt{1 - \left|\frac{T_\text{p} - T_\text{n}}{T_\text{p} + T_\text{n}}\right|}, & \text{otherwise} \end{cases}$$

Polar Word Occurrence (PWO) records the prevalence of positive, negative or neutral sentiment. It is defined as:

$$\text{PWO} = \begin{cases} 1, & \text{if } T_\text{p} > T_\text{n} \\ -1, & \text{if } T_\text{p} < T_\text{n} \\ 0, & \text{if } T_\text{p} = T_\text{n} \end{cases}$$

NRC Lexicon based features

The extraction of more specific emotions from texts is performed by means of the NRC Emotion Lexicon [7], a dictionary that associates words with ten discrete emotions: anger, anticipation, disgust, fear, joy, negative, positive, sadness, surprise, trust. Each word in the lexicon is associated with one or more emotions through a binary label.
The features are computed as follows: for each emotion e, the number of words associated with e in the text [*N(e)*] is normalized against the text length [n]. Formally:

$$\text{Normalized Emotion Count } (e) = \frac{N(e)}{n}.$$

Note that:

- the tokenization procedure is handled by the `nltk` library;
- normalization is performed to avoid bias resulting from assigning higher values to longer texts.

VADER features

VADER (Valence Aware Dictionary and sEntiment Reasoner) features quantify the sentiment of a text by analyzing the emotional polarity of words and their intensity.
They consist of four sentiment scores calculated using the VADER model [8], which assigns positive [`vader_pos`], neutral [`vader_neu`], negative [`vader_neg`], and compound [`vader_compound`] scores to the texts of the collection.
Said $\mathcal{P}$ the set of positive words, $\mathcal{N}$ the set of negative words, $\mathbf{N}$ the set of netural words, $\mathbf{I}$ the intensity of a word according to the VADER model, such scores are defined as follows:

$$\texttt{vader\_pos} = \frac{\sum_{i \in \mathcal{P}} \mathbf{I}_i}{\sum_{i \in \mathcal{P},\mathcal{N},\mathbf{N}} \mathbf{I}_i}$$

$$\texttt{vader\_neg} = \frac{\sum_{i \in \mathcal{N}} \mathbf{I}_i}{\sum_{i \in \mathcal{P},\mathcal{N},\mathbf{N}} \mathbf{I}_i}$$

$$\texttt{vader\_neu} = \frac{\sum_{i \in \mathbf{N}} \mathbf{I}_i}{\sum_{i \in \mathcal{P},\mathcal{N},\mathbf{N}} \mathbf{I}_i}$$

$$\texttt{vader\_compound} = \frac{\sum_{i \in \mathcal{P}} \mathbf{I}_i - \sum_{i \in \mathcal{N}} \mathbf{I}_i}{\sum_{i \in \mathcal{P},\mathcal{N},\mathbf{N}} (\mathbf{I}_i)^2}$$

Note that while the first three scores – representing the proportions of positive, neutral and negative intensities with respect to the whole text – range between 0 and 1, the compound score – providing an overall sentiment score of the text – falls within the range $[-1, 1]$.

All the hand-crafted features are aggregated into a single dataset. This allows these features to be used as input for machine learning models or neural networks. Note that each emotion resulting from the NRC Lexicon based extraction has a dedicated column in such a dataset.

6 Classification task

A single label multi-class classification of the documents – with target variable `emo_label` – is performed by means of different classifiers, which are trained using two configurations:

1. only the features derived from the text representations;
2. an extended feature set that combines these text-derived features with the hand-crafted ones.

This approach enables a comparative analysis of the impact of the hand-crafted features on the classification performance.
We experiment with three classifiers: Random Forest (RF), XGBoost (XGB) and Support Vector Machine (SVM).
Among these, RF is used as the baseline model: all feature sets are evaluated using the RF to identify the best-performing sparse and dense feature configurations. These optimal feature sets are subsequently tested across the remaining classifiers to evaluate their generalizability.
Furthermore, we employ a pre-trained version of Distil-RoBERTa fine-tuned on an emotion-specific dataset (same as in 5.1), as a state-of-the-art benchmark to provide a robust performance comparison.

For each classifier, brute force hyperparameters optimization is performed, by maximizing F_1 score on the validation set.
F_1 score is calculated as the harmonic mean of

precision and recall for each class individually. Formally:

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

To obtain a single performance metric for multi-class classification, the macro-averaged F_1 score is computed as the arithmetic mean of the F_1 scores across all classes. This metric ensures that all classes contribute equally to the evaluation, making it particularly useful for analyzing performance on imbalanced datasets.

The selection of the feature set is treated as an additional hyperparameter. For each feature set, a grid search is performed over the predefined parameter space to identify the configuration that maximizes the F_1 score.

6.1 Random forest

Random Forest, an ensemble learning method, constructs multiple decision trees during training and aggregates their outputs to improve generalization and mitigate overfitting.
As our baseline model, we evaluate it (on the validation set) across a predefined hyperparameter space, detailed in the following table.

Parameter	Values
`n_estimators`	{100, 200, 300}
`max_depth`	{10, 20, None}
`min_samples_split`	{2, 5, 10}
`min_samples_leaf`	{1, 2, 4}

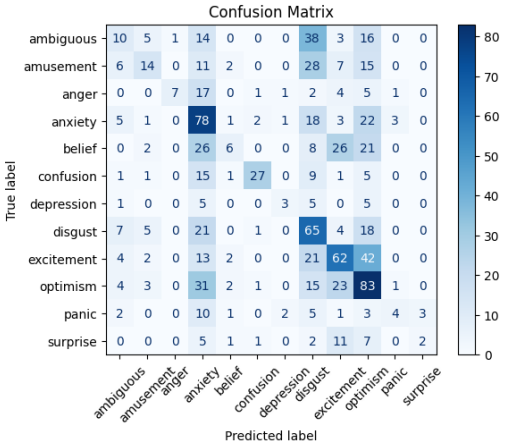
Table 1: Parameter grid for RF

A second table presents the accuracy and macro-averaged F_1 score of the model on the test set, evaluated using the best hyperparameter configuration for each feature set. Note that letter **B** indicates the basic text representation, while **E** refers to the enriched feature set.

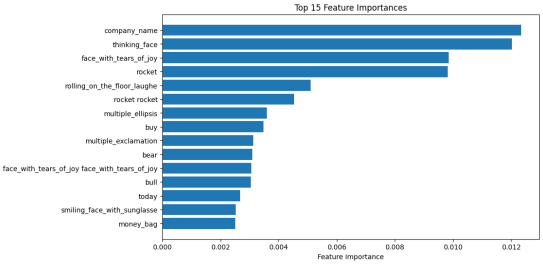
Model	Acc.	m-F1
TF-IDF B	0.34	0.28
TF-IDF E	0.34	0.28
Bigram TF-IDF B	0.36	0.30
Bigram TF-IDF E	0.34	0.28
Word2Vec B	0.23	0.13
Word2Vec E	0.23	0.13
RoBERTa B	0.26	0.15
RoBERTa E	0.26	0.15
BERTweet B	0.26	0.11
BERTweet E	0.26	0.12

Table 2: RF performances across feature sets

The performance scores suggest that handcrafted features do not improve results. The best sparse feature configuration is basic TF-IDF with bigrams, and the best embedding-based representation is the one derived from Distil-RoBERTa. These two representations will be used to train the subsequent models.
The confusion matrix of the best overall representation follows.



Finally, the graph of the most important features for such a feature configuration is presented, to be cited for further considerations.



6.2 XGBoost

XGBoost is a powerful gradient boosting framework that builds additive models by iteratively optimizing an objective function.
The model is configured with the objective function `multi:softmax` for multi-class classification and the evaluation metric `mlogloss`. Learning rate and maximum tree depth are tuned according to the table below. Early stopping with 10 rounds is applied based on validation set performance, and the best configuration is selected by maximizing the macro-averaged F_1 score.

Parameter	Values
`learning_rate`	{0.01, 0.05, 0.1}
`max_depth`	{4, 6, 8}

Table 3: Parameter grid for XGB

Model accuracy and macro-averaged F_1 score for the two best performing feature sets follow.

Model	Acc.	m-F1
Bigram TF-IDF B	0.35	0.32
RoBERTa B	0.27	0.21

Table 4: XGB performances

6.3 Support vector machine

Support Vector Machine is a supervised learning algorithm designed to find the optimal hyperplane separating data points into distinct classes. We use a linear kernel, well-suited for textual representations, and optimize the regularization parameter `C`, which controls the trade-off between maximizing margin and minimizing classification errors.

Parameter	Values
`C`	{0.01, 0.1, 1, 10, 100}

Table 5: Parameter grid for SVM

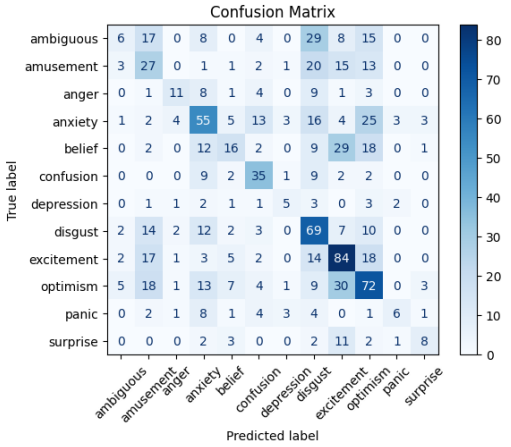
Model accuracy and macro-averaged F_1 score on the two best performing feature sets follows.

Model	Acc.	m-F1
Bigram TF-IDF B	0.35	0.29
RoBERTa B	0.31	0.26

Table 6: SVM performances

6.4 Distil-RoBERTa

We now show the results obtained by directly using the fine-tuned Distil-RoBERTa model for emotion classification. While this model was previously utilized to generate contextual embeddings, it is now employed as a full classifier, serving as a benchmark to compare the performance of our tested models against a state-of-the-art solution. First, the confusion matrix is represented for further comparison with the baseline one.



Last, performance scores are reported.

Model	Acc.	m-F1
RoBERTa B	0.39	0.35

Table 7: Distil-RoBERTa performances

7 Topic modeling task

To perform topic modeling on the dataset, two different models are implemented: Latent Dirichlet Allocation (LDA) and BERTopic. Respectively, LDA is trained on the TD-IDF representation, while BERTopic exploits Word2Vec as the embedding model.
To evaluate the performance of the models, coherence is selected as the appropriate metric. It assesses the semantic similarity between the top words within a topic. Coherence combines a sliding window approach with normalized pointwise mutual information (NPMI) and cosine similarity. This result reflects the interpretability of the topics, hence it is to be maximized.

$$C_V = \frac{1}{|P|} \sum_{(w_i, w_j) \in P} \text{NPMI}(w_i, w_j) \cdot \cos(w_i, w_j)$$

7.1 Latent Dirichlet Allocation

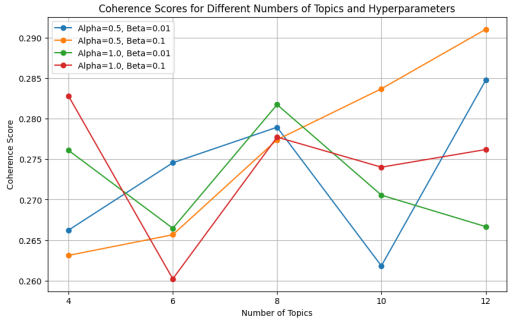
Latent Dirichlet Allocation (LDA) is a Bayesian probabilistic model, which assumes that each document in a corpus is a mixture of latent topics and that each topic is characterized by a distribution over words. LDA models documents as generated through the following process:

1. **Input**: A corpus of documents and a predefined number of topics (k).
2. **Generative process**:
   - Each document is associated with a distribution over topics (document-topic distribution).
   - Each topic is associated with a distribution over words (topic-word distribution).
   - Words in a document are generated by sampling a topic from the document's topic distribution, then sampling a word from the chosen topic's word distribution.
3. **Output**: A set of topics and the representation of each document as a combination of those topics.

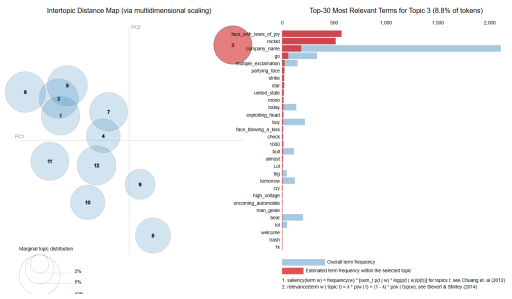
LDA exploits Dirichlet distributions to ensure that documents are composed of few dominant topics and that the topics consist of a limited subset of words. The sparsity of these distributions is controlled by two hyperparameters: α and β. To optimize the model, the following combinations of hyperparameters are evaluated in terms of C_V with brute force.

Parameter	Values
k	{4, 6, 8, 10, 12 }
α	{0.5, 1}
β	{0.01, 0.1}

Table 8: Parameter Grid for LDA



In general, the C_V results of the topic modeling are low (close to 0.3), which shows the difficulty of the algorithm in modeling the tweet posts. The low number of words per document, the high intrinsic noise and the variability in the date of the posts (coming between 2022 and 2023) represent a difficulty for this task.
Using the pyLDAvis module, it is possible to have a deeper inspection of the result (optimized for 12 topics, alpha equal to 0.5 and beta equal to 0.1). On the left of the following figure, in the dimensionality reduction plot, it is possible to see that some topics (like 1,2,6,8 and 4,7,12) overlap. Topic 3 appears to be more distant in this lower-dimensional representation.

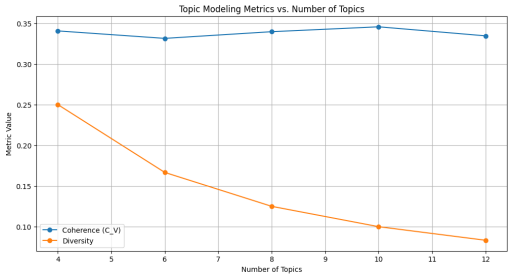


Inspecting the result, the following topics are discernible.

- **Topic 1**: company_name, price, stock, earning, grinning_squinting_face, know, people, would, winking_face, see
- **Topic 2**: company_name, thinking_face, multiple_ellipsis, chart_decrease, drop_of_blood, fall, future, year, car, new
- **Topic 3**: face_with_tears_of_joy, rocket, company_name, partying_face, strike, multiple_exclamation, go, star, moon, united_state
- **Topic 4**: company_name, get, light_skin_tone, share, break, bullish, 100, go, right, ready
- **Topic 5**: money, mouth_face, company_name, put, see, tomorrow, buy, multiple_exclamation, today
- **Topic 6**: company_name, short, week, sell, news, next, one, well, go, lol
- **Topic 7**: rolling_on_the_floor_laughing, company_name, clown_face, bull, buy, grinning_face_with_sweat, dip, Elon, guy
- **Topic 8**: multiple_ellipsis, company_name, multiple_exclamation, time, back, let, still, wait, go, soon
- **Topic 9**: money_bag, company_name, dollar_banknote, red_heart, battery, zany_face, baby, face_screaming_in_fear, cash, Monday
- **Topic 10**: company_name, hold, smiling_face_with_sunglasses, chart_increase, market, green_apple, split, call, thumbs_up, keep
- **Topic 11**: company_name, high, buy, long, take, sell, folded_hand, low, 500, close
- **Topic 12**: company_name, bear, fire, eye, like, day, loudly_crying_face, look, see, today

7.2 BERTopic

BERTopic is a topic modeling framework that combines clustering algorithms with dense vector representations of documents. In particular, word2vec is the selected embedding. BERTopic uses HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) to group reduced embeddings into clusters (reduced because dimensionality reduction techniques like UMAP are applied).
Evaluating BERTopic on the same number of topics tested with LDA (4,6,8,10,12), leads to the results shown in the following image.



The C_V values have very low variability (all approximately equal) and low magnitude (slightly higher than 0.3). This indicates that the model struggles to identify a clear pattern in the topics. The noise in the data complicates the task of modeling the topics in a definite way, as seen for LDA too.
Inspecting the results, similarly to LDA, the following lists of relevant topics are obtained - note that topic indexed with -1 is the outlier topic (hence is not included):

- **Topic 0**: company_name, money_bag, multiple_ellipsis, thinking_face, sell, buy, get, go, see, today
- **Topic 1**: face_with_tears_of_joy, rolling_on_the_floor_laughing, company_name, multiple_ellipsis, clown_face, bull, bear, lol, go, buy
- **Topic 2**: mouth_face, money, company_name, light_skin_tone, dark_skin_tone, smiling_face_with_heart, eye, evil_monkey, see, star
- **Topic 3**: rocket, multiple_exclamation, company_name, money_bag, multiple_ellipsis, go, moon, day, buy, let
- **Topic 4**: facebook, meta, multiple_ellipsis, company_name, chart_decrease, call, bear, buy, go, backhand_index_pointing_right
- **Topic 5**: bear, company_name, multiple_ellipsis, bull, bears, like, eat, short, go, get
- **Topic 6**: fire, company_name, multiple_ellipsis, multiple_exclamation, burn, dizzy, put, short, go, tornado
- **Topic 7**: dollar_banknote, drop_of_blood, smiling_face_with_sunglasses, money_with_wing, chart_increase, folded_hand, yen_banknote, company_name, flexed_biceps, check_mark_button
- **Topic 8**: money_bag, multiple_exclamation, fire, company_name, bottle_with_popping_cork, wine_glass, money_with_wing, clinking_glasses, make, smiling_face_with_open_hands

When evaluating both the presented algorithms, it is important to keep in mind that the dataset consists only of 8000 observations with an average length of approximately 18 words. Additionally, the intrinsic noise (given the extreme variety of the possible topics) complicates the topic modeling task.

7.3 Comparison with classification

As a comparison, the word clouds corresponding to the different emotions proposed for the single-label multi-class classifications are computed.
One observation is to be made before: the different emotions do not present definite "boundaries" between each other, meaning that if they are assumed to be topics, they also display overlap between each other. This property can be observed, for instance, inspecting the word clouds for "belief" and "surprise" (respectively the fifth and eight most popular emotion).

Also other emotions display ambiguity, or are not well defined as a topic. This property reflects reality, since a tween can be characterized from a mixture of emotions (or labels, or topics).
Due to the higher performance, the topics created from BERTopic are compared with the word clouds of the different emotions. The following similarities are observed.

Topic	Observed similarity
0	excitement, surprise
1	amusement
2	ambiguous
3	surprise, excitement, belief
4	amusement
5	ambiguous
6	unknown
7	belief
8	surprise

Table 9: Observed similarities between the topics and the emotions

Finally, it is important to note that, as seen in the feature importance with the random forest classification (the best performing classifier), emojis play a pivotal role in determining the content of the documents also in topic modeling.

8 Conclusions

In conclusion, the classification task displays the best performance in terms of macro F1-score with XGB on the bigram TF-IDF (0.32), followed from the RF in the same representation (0.31). The engineered features do not imply an increase in the performances and thus at the end they are not included. In average, the classifiers trained on the embedding models underperformed with respect to the simpler representations with TF-IDF in the current dataset. The relatively poor performances in general are confirmed by the usage of state-of-the-art models, looking in particular at the confusion matrixes (all displaying poor performance on the rare classes, better performance in the common ones).
As for the topic modeling, both LDA and BERTopic display poor performances in terms of C_v coherence, reaching approximately 0.3 even with several experiments on different hyperparameters values.
Both the classification and topic modeling tasks revealed the importance of the emojis, respectively in the feature importance and in the most relevant topics. This confirms the choice of encoding them, instead of filtering, in order to mantain the semantic meaning of the document.

9 Future perspectives

The current project can be extended exploring different models for classification and topic modeling, as well as different evaluation metrics for topic modeling.
Text clustering could be performed on the dataset to gain more insights on the possible topics in the documents (optimizing the number of clusters).
The results of topic modeling can be inspected further, for example with the definition of an automated approach to inspect the association with the emotion labels. An idea can be to measure the numbers of shared top relevant terms (assigning a weight depending on the relevance) and measuring the association.
Furthermore, the output of the topic modeling can be added in input to a new classification pipeline, by definition of a document-topic matrix. Training the models with these additional features can potentially increase the performance.

References

[1] Milikich and Johnson, "Taureau: A Stock Market Movement Inference Framework Based on Twitter Sentiment Analysis", 2023

[2] Mokhtari et al., "The Impact of Twitter Sentiments on Stock Market Trends"

[3] Nguyen et al., "BERTweet: A pre-trained language model for English Tweets", 2020

[4] https://huggingface.co/, 2022

[5] Hu and Liu, https://www.cs.uic.edu/ liub/, 2004

[6] Kumar et al., "IITPB at SemEval-2017 Task 5: Sentiment Prediction in Financial Text", 2017

[7] Mohammad and Turney, https://saifmohammad.com/, 2011

[8] Hutto and Gilbert, "VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text", 2014