Sentiment Analysis

Idea: Use NLP to quantify subjective information

Typically: classify polarity [ex: positive/neutral/negative]

More complex polarities also possible

↳ ex: hawkish vs. dovish statements by the Federal Reserve

There is rarely an objective score

Performance is traditionally measured by how well it agrees with human judgement

General Approaches

1) Manual Tagging

a) Define a list of polarized words (ex. bad, worst, good, best)

b) Count the number of positive/negative words in a given text

c) If positive > negative then positive sentiment

< negative

= neutral

↳ Can be made more sophisticated, but this is the basic idea

Benefits: Simple + explainable

Issues:
- Constructing the dictionary is very time consuming
- Each application requires a different dictionary
  - ↳ ex. Are high oil prices good? Depends on the industry

2) Automatic Tagging / Machine Learning

Fundamentally a classification problem

Take previously scored documents (ex: product reviews)

↳ preprocess and use to train a classifier

Use this classifier to score new documents

Benefits: Learns the language relationships with minimal upfront work

Issues: Requires a large corpus of scored documents for training

↳ Lack of large labeled financial datasets makes using ML for financial sentiment analysis challenging

Approaches a) Meta-learning using pre-trained language models

b) Automatically score documents based on the response of interest

This permits either regression or classification

ex: Score Federal Reserve reports / speeches based on the impact on the yield curve of Treasury bonds over the following day

↳ Idea: "objective" score does not matter only care how the market reacts

BERT and FinBERT

FinBert is a text classification scheme based on the pre-trained BERT language model for financial text

「Other language models exist which can be used instead of BERT」

Bidirectional Encoder Representations from Transformers (BERT)

Idea: 1) Attempts to predict randomly masked tokens in a sequence instead of the "next" token

2) Classify if 2 sentences follow each other or not

- Developed by Google & trained on a huge corpus of online documents
  - ↳ contains over 100M parameters

Masked Language Model (MLM)

MLM enforces the bidirectional learning from text by masking / hiding a word in a sentence & forcing BERT to use the rest of the sentence to predict that missing word

ex: I am eating ice cream

I am ___ ice cream

↑ eating, licking, sharing, scooping, ...

NOT chair, fruit, nice, ...

This is in comparison to Causal Language Modeling (CLM) which is unidirectional

↳ "I am ___" without looking at the rest of the sentence

Idea: Train the neural network to "understand" language as humans would

(ie. context clues)

Next Sentence Prediction (NSP)

NSP is used to help BERT learn the relationship between sentences
by predicting if 2 sentences follow each other or not

ex: Paul went shopping. He bought a new shirt. [Correct pair]

Ramona made coffee. Ice cream is on sale [Incorrect pair]

In training, 50% of correct pairs are combined with 50% of incorrect pairs
to balance the dataset/classifier

Transformers (Details in "Attention is All You Need" (2017))

Similar to recurrent neural networks in that transformers are designed to process sequential data

Unlike RNNs, transformers process the entire input all at once

↳ As discussed with MLM, this allows BERT to "read" sentences in both directions

Fundamentally, a transformer is an encoder-decoder with an attention mechanism

Encoders + Decoders

The encoder takes an input sequence + maps it into a vector

The decoder takes this vector and turns it into an output sequence

ex: In cryptography, we encode documents which can then be decoded back to an exact copy of the original

[In NLP, we are not interested in exact copies]

ex: Imagine 2 translators: 1 Knows English + German
1 Knows English + Mandarin

↳ To translate from German to Mandarin, the translators "encode" the information in English

Attention Mechanism

Idea: Mimic cognition by enhancing some parts of the input + diminishing other parts

Can think of this as the neural network highlighting keywords in the text

Typically accomplished with multiplication of elements
(especially with a softmax which maps between 0 & 1)

FinBERT

Begin with the trained BERT model on generic text

Recall: We want sentiment, not just an NLP model

And we want it to specifically understand financial text

2 Steps

1) Further train BERT on financial data

「Authors of this method used Reuters News Articles」

2) Create a sentiment analyzer

↳ Add a dense neural network to the last hidden layer / hidden state of BERT (for regression or classification)

↑ This approach can be used with newer NLP tools as well
ex: ChatGPT, GPT-4, etc.

But the typical issues for training data still exist

↑ Authors of this method used "Financial Phrase Bank" which has annotated "financial sentences"
+ "FiQA Sentiment" which has sentiments between -1 + 1 ↓

In practice, FinBERT is shown to perform well compared to simpler methods

You can install FinBERT yourself (or BERT alone)!