Harry Hillsdownley
Hsh47
Case Western Reserve University

Sentiment Analysis for Movie Reviews using a Hidden Markov Model

3

Abstract

Sentiment analysis is at the core of natural language processing, and it has become increasingly popular because of the rising popularity of large language models (LLMs). This project aims to probabilistically examine movie reviews to have a probabilistic estimation of the sentiment (positive, negative, neutral). To accomplish probabilistic sentiment estimation, this paper will explore the application of Hidden Markov Models (HMMs) to examine the sequential nature of language and analyze the sentiment for movie reviews.

Because of the structure of Hidden Markov Models and their ability to link causal factors to hidden states, I will be using this model for sentiment analysis. This paper is going to work on the idea that every word has a possible sentiment, and by observing a sequence of words (states) in a review, we could use HMMs to determine the hidden sentiment for these observations and do a majority or weighted voting to determine the sentiment of the whole review. Using an HMM is suitable because this probabilistic model can estimate the probability of a word having a certain sentiment given the context of the previous word and its sentiment, similar to generating sentiments based on the tokenized input.

Some challenges this paper expects to have are the probability computation for the probabilities of each of the words inside the vocabulary, as for this model to work, an emission state matrix must be filled for the probabilities of the word being generated from a sentiment. Additionally, another problem this paper expects to have is the process of cleaning up the dataset and building the probabilities that HMMs require, and generating valid probability distributions for words given their sentiment. Additionally, the sparse representation of sentiment-possessing words for our model will pose a challenge to the distribution of sentiment-possessing words.

Harry Hillsdownley
Hsh47
Case Western Reserve University

Background

Markov Chain:

A hidden Markov Model is based on augmenting the Markov chain. A Markov Chain is a model where the state transitions through probabilistic states, and the only thing that matters to make the transition is the current state. This is known as the **Markov Assumption**. More formally,

$$P(X_{n+1}|X_{n,}, X_{n-1},..., X_1) = P(X_{n+1}|X_n)$$

We could denote a Markov chain as the probabilities of transitioning between discrete states, such as the weather being hot, cold, or warm, with the transition probabilities between each state being specified. This means that a Markov chain can be specified by three components:

- Set of N states $Q = \{q_1, q_2,..., q_n\}$
- A transition probability matrix $A = a_{11}a_{12}.... a_{N1}...... a_{NN}$
- An initial probability distribution $\pi = \{\pi_1, \pi_2,... \pi_N\}$

Hidden Markov Model using the Markov Chain:

The Hidden Markov Model augments the idea of the Markov Chain being useful to compute a sequence of observable events given their transition probabilities. However, not all events are observable, some events can be hidden, so we cannot observe them directly. For example, when we look at text, we do not see the part-of-speech of particular words, we rather infer them from the sequence of observable words, in this case, the part of speech is the hidden states, while the observable states are the words inside the text, and we can infer probabilistically the part of speech certain words belong to given their sequence and the probability of transitioning from one part of speech to another, the transition matrix represents this. The hidden Markov model allows us to relate hidden events and observable events through causal factors in the probabilistic model.

This relationship between hidden and observable events is formalized through two key probability distributions: the transition probabilities and the emission probabilities. The transition probabilities, as mentioned, define the likelihood of moving from one hidden state to another, capturing the temporal dependencies between these unobserved events.

An HMM, therefore, needs the following components:

- A set of N states $Q = \{q_1, q_2,..., q_n\}$
- A transition probability matrix $A = a_{11}a_{12}.... a_{N1}...... a_{NN}$

Harry Hillsdownley
Hsh47
Case Western Reserve University

- A sequence of observation likelihoods: probability of an observation $B = b_i(o_T)$
- An initial probability distribution $\pi = \{\pi_1, \pi_2, ... \pi_N\}$

The simplest form of an HMM is the first-order hidden Markov Model, which instantiates the Markov Assumption and Output Independence simplifying assumptions:

Output Independence:

$$P(o_i | q_1 ... q_i .... q_n, o_1, ... o_2 ... o_t) = P(o_i q_i)$$

Explains that the probability of an observation o_i is **only** dependent on the state that produced that observation and not any other states or observations.

The output independence assumption in Hidden Markov Models (HMMs) states that each observed word depends only on the current hidden state and not on previous observations can be problematic for sentiment analysis of sentences because natural language is inherently context-dependent. In sentiment-rich text, the meaning and sentiment of a word often rely heavily on its surrounding words. For example, the word "great" conveys a positive sentiment, but in the phrase "not great," the overall sentiment is negative. HMMs that treat observations as conditionally independent fail to capture such nuances, leading to potential misclassifications. This limitation affects the model's ability to accurately detect the sentiment expressed across multi-word expressions or complex syntactic constructions, where the sentiment is not encoded in isolated words but in their interdependent usage.

Inference Process for Hidden Markov Models: Decoding using the Viterbi Algorithm

For the process of decoding hidden states by analysing a sequence of observations, the Viterbi Algorithm is used. This algorithm finds the most likely sequence of hidden states the result of the sequence of observations. The Viterbi algorithm does this by finding the best path of states in the Hidden Markov Model that emits the observation sequence with maximum probability. In other words, Viterbi estimates the Maximum Likelihood of hidden states to produce the observations.

The Viterbi algorithm computes the best path of hidden states through dynamic programming. It starts by performing a Forward pass through the observations and their likely states by keeping track of:

$$\delta_1(z_i) = \pi(z_i) * B(z_{i,} x_1)$$

$$\psi_1(z_i) = 0 \text{ If no predecessor}$$

Harry Hillsdownley
Hsh47
Case Western Reserve University

Where:

$\delta_t(z_i)$: the maximum probability of reaching state z_i at time t, given the previous observations

$\psi_t(z_i)$: The backtracking pointer that stores the best previous state

By this formulation, hence, the termination for Viterbi is when t =T, meaning the end of the observations, where the final state can be defined as:

$$z^*_T = argmax_{z_i} \delta_T(z_i)$$

Following the forward pass, the Viterbi algorithm executes a backward pass to reconstruct the optimal sequence of hidden states. This traceback utilizes the backtracking pointers, ψt(zi), accumulated during the forward step. Starting from the final state, zT∗, the algorithm iteratively retrieves the preceding state, zt−1∗, using the pointer ψt(zt∗). This process continues until the initial state, z1∗, is reached, effectively tracing the path with the highest probability through the Hidden Markov Model. The resulting sequence, z1∗,z2∗,...,zT∗, represents the most likely sequence of hidden states that generated the observed sequence, providing a solution to the decoding problem.

Sentiment analysis using hidden Markov:

Sentiment analysis using a Markov model lies in tying the underlying language to concepts inside the model. When communicating through language, humans use sequences of words to express a message, which can be associated with a sentiment on the matter. In the case of movie reviews, the text is a sequence of words (observed states) that convey a message about a movie, often expressing the writer’s sentiment towards the movie (hidden states).

By analysing the sentiment of the sequence of words, and how likely the sentiment of the words in a sequence is to change as it progresses, we can probabilistically define sentiment for every word in the review, and generate a probability of overall sentiment from a review.

This project will be using a Hidden Markov model to find the hidden state (sentiment) with the highest probability of generating the current observation (review word) via the Viterbi algorithm. In other words, we will be performing a scan word by word to produce the sequence of most likely hidden states (sentiments) that generated the observed word in the review, and will generalize the majority sentiment to the sentiment of the review.

Harry Hillsdownley
Hsh47
Case Western Reserve University

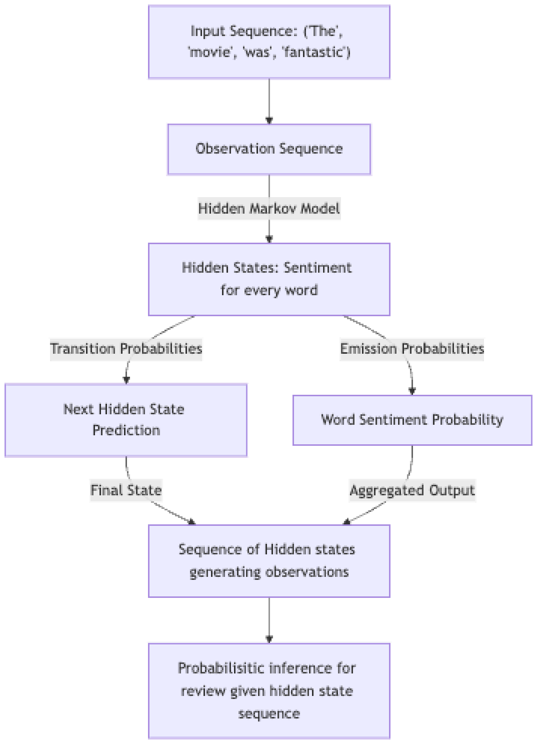
Input Sequence: ('The', 'movie', 'was', 'fantastic')
Observation Sequence
Hidden Markov Model
Hidden States: Sentiment for every word
Transition Probabilities
Emission Probabilities
Next Hidden State Prediction
Word Sentiment Probability
Final State
Aggregated Output
Sequence of Hidden states generating observations
Probabilisitic inference for review given hidden state sequence

Figure 1: Illustration of how HMM will process a review

Generating Probability Matrices for Hidden Markov Model

1. **Emission Probabilities**

An emission probability for sentiment analysis using HMM is formally defined as:

$$B_{ij} = P(word_i \mid sentiment_j)$$

This means that we could calculate the probability of a word given that it was generated by a positive or negative observation, but there is a catch: we cannot generalize the sentiment of a review to all the words contained inside the review. For this review generalization problem, I found what is called the AFINN Lexicon. The AFINN lexicon assigns words a score between -5 and 5, with negative scores indicating negative sentiment and positive scores indicating positive sentiment.

The AFINN lexicon only holds words with a relative sentiment. Therefore, we will assume that words from the processed dataset that are not recognized as sentimental by AFINN will hold no meaningful sentiment (an AFINN score of 0) to make this calculation easier. This implies that words not included in AFINN will be of neutral sentiment. This idea is also intuitive because for the exploration of the data, words not included in AFINN are largely nouns, names

Harry Hillsdownley
Hsh47
Case Western Reserve University

of characters, or misspelled words, which are hard and computationally intensive to process in the dataset are sentiment-neutral.

Because of AFINN, we can find the statistical sentiment for specific words in the English language. Now we can use the softmax function and create feature vectors to feed them to compute the emission probabilities for words.

Softmax

The softmax function is a generalization of the logistic function that converts a vector of real numbers into a probability distribution.

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Where:
Z is an input vector with K real numbers
σ_i is the probability distribution for the input vector Z

To create a feature vector to produce the probability distribution for a word, the following assumptions are made: The strength of one sentiment has the same magnitude, but the opposite sign as that of its contrary sentiment. Therefore, a strong sentiment towards one end will have a 'neutral sentiment' of 0

Emission probabilities matrix

To calculate the emission probabilities matrix, I first recognized the properties of the scores matrix from which we could obtain the feature vectors matrix, F.

$$\begin{bmatrix} w_1 * s_1 & w_1 * s_2 & \dots & w_n * s_n \\ w_2 * s_1 & w_2 * s_2 & \dots & w_2 * s_n \end{bmatrix}$$

Where W is the feature vector of the weights for the softmax distribution, and S is the AFINN score for the words on the column index. Keep in mind that a row in matrix F represents

Harry Hillsdownley
Hsh47
Case Western Reserve University

a sentiment; for notation purposes, index 0 represents a positive sentiment, while index 1 represents a negative sentiment.

To formulate this matrix, I used the outer product of the vectors W and matrix S to end with the resulting matrix.

$$\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} [s_1, s_2, \ldots, s_n] = \begin{bmatrix} w_1 * s_1 & w_1 * s_2 & \ldots & w_n * s_n \\ w_2 * s_1 & w_2 * s_2 & \ldots & w_2 * s_n \end{bmatrix}$$

For weights, I followed the assumptions stated above that a word will have the same magnitude but opposite sign for contrary sentiments. Therefore:

$$\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Once I took the softmax of each of the rows inside matrix F, I was left with a valid probability distribution for each row, representing the probability that word i was generated from sentiment j.

Transition probabilities

For calculating the transition probabilities, I translated every word inside its specific review to its AFINN equivalent, such that a sentence would be transformed into an array of hidden states represented by numbers. 0 for positive, 1 for negative. Therefore, the sentence "That was a good movie" would be encoded into something like this [0]. It is important to mention that because of sparsity and the density of words without any sentiment, our review representations will only count the "sentimental" words of a review. Because of this representation, we will have a flow of sentiment throughout the review. Take the following as an example:

Example:

Review

I liked this movie from the beginning. The characters are well-developed and work together. However, by the end of the movie, it got long and incomprehensible.

Harry Hillsdownley
Hsh47
Case Western Reserve University

Output of tokenization:

[liked, well, developed, long, incomprehensible]

Output of sentiment tokenization:

[0,0,0,1,1]

To elaborate, the transformation of reviews into numerical sequences using AFINN sentiment scores allows for a simplified, yet informative, representation of the emotional flow within a text. By mapping words to their corresponding sentiment values and then categorizing them as positive (0) or negative (1), we effectively create a discrete sequence of hidden states. This sequence, while sparse due to the exclusion of neutral words, captures the essential shifts in sentiment throughout the review. For example, a review like "The acting was great, but the plot was disappointing" could be represented as [0, 1], indicating an initial positive sentiment followed by a negative one. This representation allows us to model the transition probabilities between positive and negative sentiments, revealing patterns in how sentiment evolves within reviews. These transition probabilities are crucial for understanding the overall sentiment trajectory and can be used to analyze the dynamics of opinion expression in textual data.

Once we have tokenized using the hidden states of training data, what is left is a nested array of 1s and 0s. These represent hidden states for the data.

$$T = \begin{bmatrix} 0.62 & 0.38 \\ 0.48 & 0.52 \end{bmatrix}$$

Applying Hidden Markov Models for Sentiment Analysis.

The Dataset:

For this project, I will be using the IMDB Movie Reviews dataset from Kaggle. This dataset can be found in the bibliography section of this document. This is a dataset with 50,000 movie reviews in a CSV Format, with two columns: the movie review and the binary sentiment labels.

Harry Hillsdownley
Hsh47
Case Western Reserve University

Dependencies:

For this project, I used Python 3.9 as the programming language, thanks to its fast data processing and a wide variety of pre-implemented libraries for Hidden Markov Models. For the main library, I used hmm learn, which is a library designed for working with HMMs. I used this

Process by milestones:

The first challenge I faced with milestone 1 was having to redefine the HMM structure multiple times, as I thought that reviews could be inferred as a whole. That is, the HMM model can predict the sentiment of a review solely based on the review as a whole; no tokenization or other information is needed. I realized that the representation of the problem mentioned above was flawed and had to redefine the framework.

When redefining the framework, I came across the tokenization approach for words. However, this meant that the emission probability representation would be very large, depending on the effectiveness of my tokenization algorithm.

While working on these challenges, I also learned more about the Viterbi algorithm and its dynamic programming approach for decoding sequences of observations using HMMs. I also wanted to better understand how to approach tokenization effectively, as the initial vocabulary from the movie reviews was about 428,000 tokens (not necessarily words).

To solve the tokenization problem, I used Regex to parse through each movie review and extract the vocabulary, discarding numbers and processing HTML tags that were embedded in the data. This proved that data pre-processing was an essential part of this project. Simultaneously, another problem arises: how to represent sentiment across the vocabulary for the model to train on, or at least, generate a valid probability distribution for the emission probabilities. For this challenge, I used AFINN, translated the vocabulary into sentimental words from AFINN, and converted them into probability distributions using the softmax function. With the softmax function, the biggest challenge was over: calculating the emission probabilities, as this matrix is the backbone of the HMM.

Once done with the emission probabilities. I delved into the transmission probabilities. With the approach mentioned earlier in the paper. This process was more straightforward than the emission probabilities, as I just had to convert the words in reviews to indices and sentiments.

Discussion against state-of-the-art models.

While the model used for this project is purely probabilistic, state-of-the-art models use transformer-based approaches to handle sentiment analysis. This is done by using models such a Bi-directional Encoder Representation from Transformers (BERT) models. BERT models use embeddings and large word vocabularies to serve complex language patterns from words, essentially deriving meaning from single words and their context in the processed pattern. This makes sentiment analysis much more effective by taking into account word patterns such as “not good,” whereas this model would just predict an indecisive sentiment from the review. Additionally state-of-the-art models use embeddings to work in a feature space which denotes meaning from the distances within specific vectors, giving them the ability to understand comprehensive context from reviews and generalize that to the sentiment of the review while HMMs can only infer probabilistically the underlying sentiment of a word. Therefore the generalization strategies of state-of-the-art models are objectively better.

Limitations of Hidden Markov Models for Sentiment Analysis

Hidden Markov Models have limited contextual understanding, as they use the Markov assumption, generalizing the probability of a word only being dependent on the previous word. This poses a problem because sentiment often depends on a longer-range context and stopwords such as ‘but’, which may change the whole sentiment of a review. Additionally, because this paper implements a first-order HMM, the probability of a word only depends on its previous word; therefore, HMMs will also struggle with things like bag-of-words, such as ‘I don’t like this’ against ‘I like this’. These limitations make HMMs ill-suited for tasks that demand nuanced interpretation of language, especially when compared to more advanced models like LSTMs or Transformer-based architectures, which can account for longer-term dependencies and the semantic relationships between words.

Comparison of My HMM vs Bidirectional Encoder Representation from Transformer (BERT) NLP approach.

When looking at the dataset, I was able to find an approach that used a Bidirectional Encoder Representation Transformer (BERT) on the same dataset. The following insights from this discussion will be made in reference to this approach found in the sources section of this paper.

When using a test set to evaluate the performance of the BERT approach, the BERT approach achieved the following results:

```
Test Set Classification Report:
   precision    recall  f1-score   support
```

```
    negative       0.91      0.88      0.89      6250
    positive       0.88      0.91      0.90      6250

    accuracy                           0.90     12500
   macro avg       0.90      0.90      0.90     12500
weighted avg       0.90      0.90      0.90     12500
```

All of the following metrics are more effective than this paper's HMM approach by a range of 20-30% from our results, which are available in the conclusion section of this report.

This performance boost is due to a wide range of reasons. First, BERT considers the entire sentence using left and right context at the current word, using a transformer architecture, while HMM only considers the previous word for context. Additionally, BERT can also handle language constructs such as 'not good' vs 'good' and 'bad', while HMMs struggle because of their context window. This gives BERT the ability to have contextual understanding, understand word meanings, and handle complex language structures, while HMM can only generalize sentiment using the probability of the previous observations.

Conclusion:

Through the conclusion of this project, I used the Viterbi algorithm implemented into the CategoricalHMM class of hmmlearn by fitting my transmission and emission matrices into the model. The goal was to evaluate my approach to computing emission and transmission matrices against the approach that the library implemented for HMM movie reviews.

While the results of using HMMs to perform sentiment analysis were not as effective compared to other solutions for natural language processing, such as neural networks with an attention window, the approach used in this paper proved to be better metrics when compared to standard approaches, such as using the standard categorical HMM class.

The approach used in this paper boosted the accuracy of individual movie reviews by a most probable sequence of hidden states by 7% to an accuracy of 49%. Upon further research, this improvement may be due to the utilization of the softmax function for the calculation of the emission probabilities. While the results are essentially arbitrary, it is promising to use HMM in addition to another layer for sentiment analysis to refine accuracy.

The approach used in this paper also proved significantly better than the standard HMMLearn implementation in terms of confusion matrices, F1 score, precision, and recall.

Harry Hillsdownley
Hsh47
Case Western Reserve University

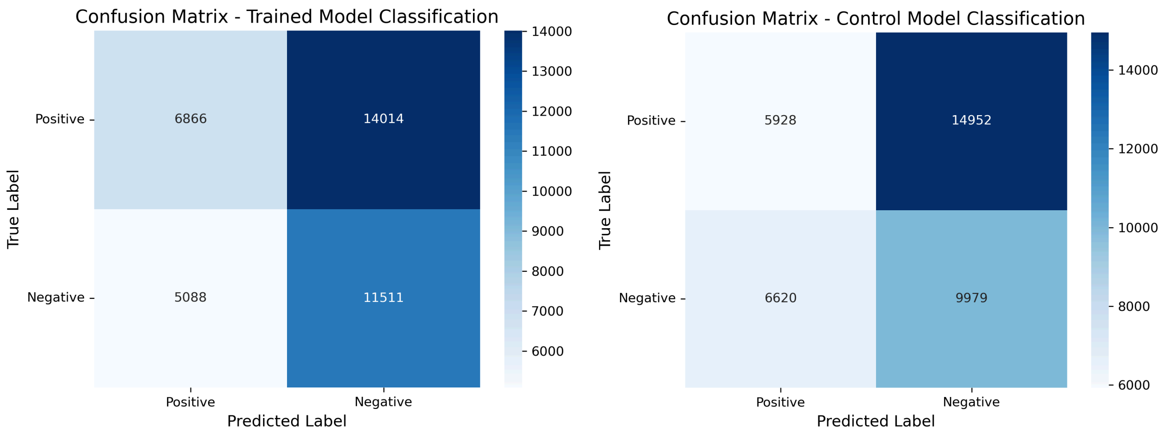

Figure 2: Confusion matrices for trained and control models

As seen by the confusion matrix for both models, the trained model performs better in terms of its true positive rate as well as its true negative rate when compared to the standard HMMLearn implementation. This evidence can also be backed up by the ROC Curves for both models:

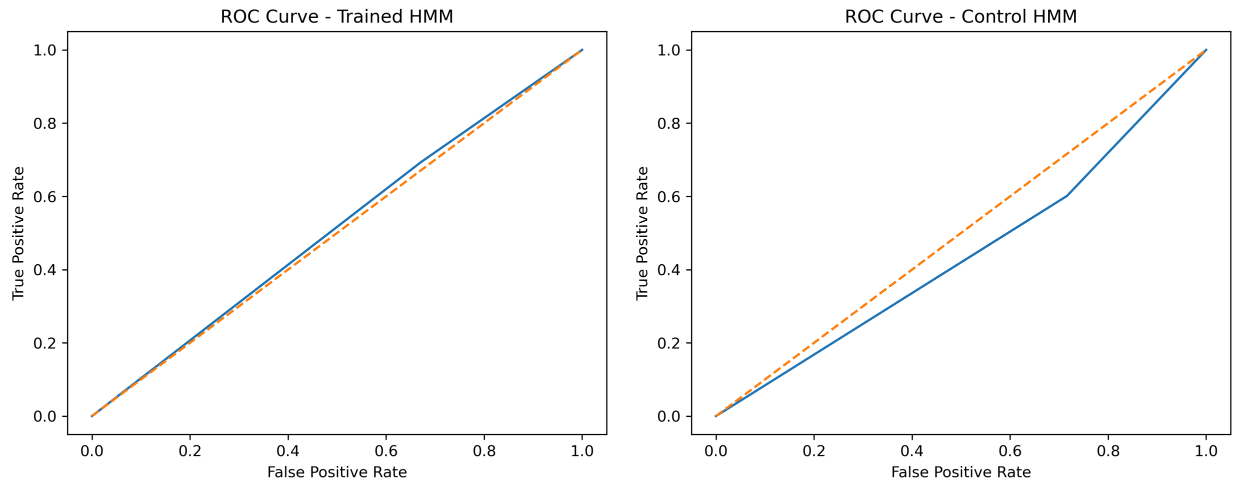

Figure 3: ROC Curves for control and trained model

While the performance of both models is almost arbitrary, the trained HMM is able to balance a better true positive rate and a better false positive rate when compared to the standard HMMLearn approach. Additionally, the precision and recall for the trained HMM backed these results:

Control f1 score: 0.48

Control precision: 0.40

Control recall: 0.60

Harry Hillsdownley
Hsh47
Case Western Reserve University

Trained f1 score: 0.55
Trained precision: 0.45
Trained recall: 0.69

In conclusion, this project demonstrated the feasibility of implementing a custom approach for computing emission and transmission matrices within the Viterbi algorithm using the hmmlearn library's CategoricalHMM class. While the overall sentiment analysis performance using HMMs remained limited compared to more advanced techniques like neural networks, our custom matrix computation significantly improved the accuracy of individual movie review sentiment predictions by 7%, achieving an average accuracy of 49%. This improvement, though modest, suggests the potential for HMMs to contribute as a supplementary layer in more complex sentiment analysis pipelines. Furthermore, the single-token inference results highlighted a substantial 7% accuracy increase using our emission probability calculation compared to hmmlearn's default methods. This statistically significant improvement likely stems from our unique word processing and softmax-based feature vector generation. These findings indicate that tailored matrix computation within HMMs can yield measurable gains, warranting further exploration of hybrid models that leverage HMMs alongside other NLP techniques for enhanced sentiment analysis.

Harry Hillsdownley
Hsh47
Case Western Reserve University

Sources

Daniel, Jurafsky, and James Martin. *Speech and Language Processing*. 7 Jan. 2023, web.stanford.edu/~jurafsky/slp3/A.pdf.

Degirmenci, Alperen. *Introduction to Hidden Markov Models*. 2014, scholar.harvard.edu/files/adegirmenci/files/hmm_adegirmenci_2014.pdf.

Rustamov, Samir, et al. "Sentiment Analysis Using Neuro-Fuzzy and Hidden Markov Models of Text." *2013 Proceedings of IEEE Southeastcon*, IEEE, Apr. 2013, pp. 1–6, https://doi.org/10.1109/secon.2013.6567382. Accessed 21 Apr. 2025.

Soni, Swati, and Aakanksha Sharaff. "Sentiment Analysis of Customer Reviews Based on Hidden Markov Model." *Proceedings of the 2015 International Conference on Advanced Research in Computer Science Engineering & Technology (ICARCSET 2015) - ICARCSET '15*, 2015, https://doi.org/10.1145/2743065.2743077.

Peralta, Billy, et al. "A Simple Proposal for Sentiment Analysis on Movies Reviews with Hidden Markov Models." *Lecture Notes in Computer Science*, Springer International Publishing, 2019, pp. 152–62, https://doi.org/10.1007/978-3-030-33904-3_14. Accessed 21 Apr. 2025.

L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," in Proceedings of the IEEE, vol. 77, no. 2, pp. 257-286, Feb. 1989, doi: 10.1109/5.18626.