Inducing Word Tags Using HMM and K-means with BERT Embeddings

Amro Salman

December 5, 2024

Overleaf(latex) word count : 1999

1 Introduction

This report explores the effectiveness of two approaches for inducing word tags. The first approach involves the use of a Hidden Markov Model (HMM) trained with an unsupervised algorithm to generate word tags. The second approach applies the K-means clustering algorithm on Bidirectional Encoder Representations from Transformers (BERT) embeddings to achieve the same objective. The performance of these two models is evaluated and compared to assess their ability to induce accurate word tags.

2 Methodology

2.1 Dataset and Preprocessing

The dataset used for this project is the Penn Treebank (PTB). From its ten columns, seven were removed, and only the three columns—*Words*, *UPOS*, and *XPOS*—were used in the code.

2.2 HMM Implementation and Training

For this project, I trained four HMM models: two on the Universal Part-of-Speech (UPOS) corpus (one supervised and one unsupervised) and two on the Languge-Specific Part-of-Speech (XPOS) corpus (one supervised and one unsupervised).

To implement these models, I created an HMM class where each object is initialized with a list of unique part-of-speech tags to determine the number of hidden states. The class defines three main functions as its interface:

- **unsupervisedTraining(numEpochs, batchSize, sentences)**: Implements the Forward-Backward algorithm to iteratively estimate the transition and emission probabilities using expected counts. The function takes the number of epochs and batch size as parameters. The batch size parameter informs the function how many sentences should we process before estimating the transition and emission matrices (an online EM is equivalent to batch size = 1, and a Batch EM is equivalent to batch size = len(sentences)). The sentences parameter contains the unlabeled sentences that are used for training.

- **supervisedTraining(corpus)**: Estimates the transition and emission probabilities directly from the labelled corpus (passed as a parameter) using maximum likelihood estimation.
- **viterbiAlgorithm(sentence)**: Uses dynamic programming to compute the most likely sequence of hidden states (tags) for a given sentence.

2.2.1 Challenges and Solutions

Challenge 1: Efficiency The initial naive implementations of the Forward-Backward algorithm and the Viterbi algorithm were too slow.

Solution: I optimized the implementations by using NumPy for efficient matrix operations and parallelization. For example, in the Viterbi algorithm, I replaced explicit loops with matrix multiplications, reducing runtime from over 5 min to under 30 seconds.

Challenge 2: Underflow Problem As sentences grow longer, intermediate probabilities in the HMM calculations become extremely small, leading to underflow errors.

Solution: Log-Space Operations I converted all probability calculations to log space. In log space, multiplication becomes addition, and the log-sum-exp trick is used for summation:

$$\log\left(\sum_i P_i\right) = \max_i \log(P_i) + \log\left(\sum_i e^{\log(P_i) - \max_i \log(P_i)}\right)$$

Challenge 3: Zero Probabilities Zero probabilities caused errors when taking their logarithm.

Solution: I added the smallest representable float to zero probabilities before taking the logarithm.

2.2.2 Unsupervised HMM Training

I used a development corpus of 10,000 sentences to tune hyperparameters. For the final chosen training strategy, each model was trained for 60 epochs with varying batch sizes:

- 20 epochs with a batch size of 512.
- 20 epochs with a batch size of 1024.
- 20 epochs with a batch size of 2048.

Overall, this strategy ensures a balance between convergence speed and model stability.

2.3 K-means on BERT Embeddings

For this approach, we first use BERT to extract embeddings for each word in each sentence and then apply the K-means algorithm to group these embeddings into predefined clusters. The clusters should represent word categories. Two models were trained by applying k-means twice on the same embeddings (once with 17 clusters for uposCorups and once with 45 clusters for xposCorups).

2.3.1 BERT Embeddings

Challenge 1: Choosing the Right BERT Layer BERT generates embeddings from 12 layers (transformer blocks), each potentially capturing different types of information:

- Earlier layers: Represent structural and low-level linguistic features.
- Later layers: Capture high-level semantic information.

Since this task focuses on word-level tagging, I hypothesized that embeddings from earlier layers would better capture the syntactic dependencies between words and their corresponding tags. The idea is, as we go through the layers, the embeddings focus more on semantics and high-level details and lose some syntactic and low-level details.

Solution: To test this, I conducted experiments using embeddings from individual BERT layers and then applying K-means clustering with the same parameters on them. **The results agree with my hypothesis**. I used the uposCorups to test this. Using the last layer embedding I got an accuracy of around 26% and as I used earlier layers the accuracy increased to reach 56% at the first layer. Based on these results, I used the **first layer embeddings of BERT** for this task.

Challenge 2: Handling Wordpiece Tokenization BERT's tokenizer splits some words into multiple subword tokens, generating embeddings for each token separately. This creates a mismatch between the number of generated embeddings and the number of actual words.

Solution: For each word, I averaged the embeddings of its subword tokens to obtain a single embedding per word.

Challenge 3: Computational Efficiency Extracting embeddings for all words in the dataset initially required over 4 hours.

Solution: I utilized GPU acceleration with batching to speed up the process. This optimization reduced the time to approximately 5 minutes.

2.3.2 K-means Clustering

Since the K-means objective is non-convex, different hyperparameters can lead to different convergence results. I conducted several experiments to identify the best hyperparameters for my setup.

I used a smaller dataset to test the parameters init, n_init, max_iter, and random_state. For more details on K-means and its parameters, refer to the Scikit-learn documentation here.

2.4 Evaluation Metrics

2.4.1 Accuracy

Since the output of the unsupervised models is in the form of clusters, it is necessary to map these clusters to the actual tags to calculate accuracy. For this, I used the Hungarian Algorithm, which uses dynamic programming to construct the optimal matching that maximizes the accuracy—more details about the Hungarian Algorithm here.

$$\text{Accuracy} = \frac{\text{Number of correctly predicted tags in all sentences}}{\text{Total number of tags in all sentences}}$$

2.4.2 V-Measure

The V-measure is a clustering evaluation metric that assesses the quality of the predicted clusters by balancing two components:

- **Homogeneity**: Measures whether each cluster contains only data points that are members of a single class.

- **Completeness**: Measures whether all data points of a given class are assigned to the same cluster.

The V-measure is the harmonic mean of homogeneity and completeness:

$$V = \frac{2 \cdot \text{Homogeneity} \cdot \text{Completeness}}{\text{Homogeneity} + \text{Completeness}}$$

2.4.3 Variation of Information

The Variation of Information (VI) is a metric that quantifies the distance between two clusterings. It measures how much information is lost or gained when moving from one clustering to another. It is defined as:

$$\text{VI}(U, V) = H(U) + H(V) - 2 \cdot I(U, V)$$

Where:

- $H(U)$: Entropy of the first clustering U.
- $H(V)$: Entropy of the second clustering V.
- $I(U, V)$: Mutual information between the two clusterings.

Additionally, the normalized version of the variation of information metric is:

$$\text{Normalized VI} = \frac{\text{VI}(U, V)}{H(U) + H(V)}$$

3 Results

3.1 Quantitiative Analysis

Table 1: Evaluation of Models on UPOS

Model	Accuracy (%)	Homogeneity	Completeness	V Score	VI	Normalized VI
Supervised HMM	96.03	0.9225	0.9210	0.9217	0.5536	0.0783
Unsupervised HMM	47.65	0.4577	0.4197	0.4379	4.1530	0.5621
BERT Embeddings + K-means	56.74	0.5619	0.7358	0.6372	2.2615	0.3628

Table 2: Evaluation of Models on XPOS

Model	Accuracy (%)	Homogeneity	Completeness	V Score	VI	Normalized VI
Supervised HMM	97.03	0.9493	0.9476	0.9484	0.4482	0.0516
Unsupervised HMM	36.57	0.5501	0.4734	0.5089	4.6088	0.4911
BERT Embeddings + K-means	47.03	0.6438	0.6078	0.6253	3.3495	0.3747

3.1.1 Supervised Models

From the results presented in the tables above, it is evident that the supervised HMM models outperform the unsupervised models. This is expected, as the supervised models have access to the true tags during training. These models are included here to illustrate the advantage of using labelled data for part-of-speech tagging tasks.

Additionally, the supervised HMM trained on XPOS tags performs slightly better than the one trained on UPOS tags. This can be attributed to the supervised training process, where probabilities are estimated based on tag occurrences in the data; Increasing the number of tags reduces the ambiguity in transitions, as the same corpus transitions are distributed across a larger set of tag pairs, resulting in more predictable patterns.

3.1.2 Unsupevised Models

For the unsupervised models, the results show that the BERT Embeddings + K-means model outperforms the HMM model. This outcome is expected, as BERT embeddings capture rich contextual and semantic information, which enhances the quality of clustering. In contrast, the HMM relies on statistical estimations derived from the data, which may not be as effective in identifying meaningful patterns for part-of-speech tagging tasks.

We also observe that the BERT Embeddings + K-means model performs better on the UPOS corpus compared to the XPOS corpus, with the exception of homogeneity. This highlights the challenges faced by the K-means algorithm as the number of clusters increases, requiring the detection of more subtle details to achieve effective clustering. For instance, distinguishing between broad categories such as Nouns and Verbs is simpler compared to differentiating multiple subtypes of Verbs and Nouns.

Finally, it is evident that the HMM, whether supervised or unsupervised, performs better with more tags than with fewer tags (because of what we mentioned for the supervised HMM), in contrast to the behaviour observed with the BERT Embeddings + K-means approach (due to the challenges this

imposes on K-means). However, there is an exception where the accuracy and VI of the HMM are better on the UPOS corpus compared to the XPOS corpus. For the Variation of Information (VI), this discrepancy is attributed to the entropy bias inherent in the VI metric. This is confirmed by examining the unbiased normalized VI, which eliminates this bias. Regarding accuracy, the measure is more sensitive to the number of tags than other metrics. Accuracy relies on constructing an optimal matching between clusters and tags, a process that becomes easier with fewer clusters. As a result, accuracy is generally higher for UPOS than XPOS, except in the supervised case. This exception arises because, in the supervised approach, accuracy does not depend on optimal matching, as the correct labels are directly available.

3.2 Qualtitative Analysis

I use the unsupervised HMM model and the BERT Embeddings + K-means model trained on the UPOS to illustrate some qualitative differences.

Note the indexing of the sentences is based on my sorted sentences and it differs from the original dataset order.

3.2.1 Long-range Dependencies

HMM models inherently struggle to capture long-range dependencies, whereas the BERT Embeddings + K-means approach excels in this aspect due to the contextualized representations provided by BERT embeddings.

Sentence Example 1: Journalists and their families are constantly threatened **as are the newspaper distribution outlets**. (Index 16577)

In the given sentence, there is a connection between "Journalists and their families" and "newspaper distribution outlets" that aids in determining the correct tags. The analysis table below demonstrates that the K-means tags, powered by BERT embeddings, successfully classify these related parts correctly. For instance, recognizing "Journalists" as a noun helps the model correctly infer that "newspaper" is also a noun. In contrast, the HMM not only misclassifies both "newspaper" and "Journalists," but it also assigns them different tags, highlighting its inability to capture such long-range dependencies.

Table 3: Part of sentence 16577 analysis

Word	Journalists	and	their	families	as	are	the	newspaper	distribution	outlets
Gold Tag	NOUN	CONJ	PRON	NOUN	SCONJ	VERB	DET	NOUN	NOUN	NOUN
HMM Tag	AUX	CONJ	DET	ADP	SCONJ	VERB	DET	SYM	CONJ	NOUN
Kmeans Tag	NOUN	CONJ	PRON	NOUN	SCONJ	VERB	DET	NOUN	NOUN	NOUN

Sentence Example 2: Friday 's Market Activity. (Index 10606)

In this sentence, "'s" is treated as a separate word, which causes the HMM to struggle to establish a connection between "Friday" and "Market." In contrast, the K-means tags effectively captured this dependency, correctly classifying both words.

Table 4: Sentence 10606 analysis

Word	**Friday**	**'s**	**Market**	**Activity**
Gold Tag	PROPN	PART	PROPN	PROPN
HMM Tag	PROPN	PART	SYM	X
K-means Tag	PROPN	PART	PROPN	PROPN

Finally, given that HMMs clearly struggle with capturing long-range dependencies, it is reasonable to anticipate poorer performance on long sentences compared to K-means, while performing relatively better on shorter sentences.

Table 5: Accuracy Comparison for Short and Long Sentences

Sentence Length	**HMM Accuracy (%)**	**K-means Accuracy (%)**
Short Sentences (len $<$ 10)	78.60	74.86
Long Sentences (len $\geq$ 10)	61.94	64.15

The results align with expectations, further confirming that HMMs struggle with long-range dependencies, whereas BERT + K-means handles them more effectively (because it is likely to have more of these dependencies in longer sentences).

3.2.2 Performance on sentences with missing context

Because BERT provides contextualized embeddings, it is expected to struggle with sentences that lack complete meaning or context. In contrast, HMMs, being purely statistical models, do not rely on sentence meaning and are likely

to perform equally well on such sentences as they would on fully meaningful ones.

Sentence Example 1: Nobody has any credible estimate . (Index 20598)

Table 6: Sentence 20598 analysis

Word	**Nobody**	**has**	**any**	**credible**	**estimate**	**.**
Gold Tag	PRON	VERB	DET	ADJ	NOUN	PUNCT
HMM Tag	PRON	VERB	DET	ADJ	NOUN	PUNCT
K-means Tag	DET	VERB	DET	DET	DET	PUNCT

Sentence Example 2: Turnover remained relatively small . (Index 34503)

Table 7: Sentence 34503 analysis

Word	**Turnover**	**remained**	**relatively**	**small**	**.**
Gold Tag	NOUN	VERB	ADV	ADJ	PUNCT
HMM Tag	NOUN	VERB	ADV	ADJ	PUNCT
K-means Tag	NOUN	NOUN	NOUN	NOUN	PUNCT

In both examples, the lack of context significantly impacted the performance of BERT Embeddings + K-means, achieving only 50% accuracy on the first sentence and 40% accuracy on the second. In contrast, the HMM achieved 100% accuracy on both sentences, demonstrating its insensitivity to contextual dependencies.

3.2.3 How models deal with the same word in different contexts

Based on the observations, it is reasonable to expect that HMM will struggle to classify the same word with different tags depending on the context. However, the contextualized embeddings provided by BERT make this task more manageable for the K-means approach. We will look at how the word "open" was tagged by the models in two different contexts.

Sentence Example 1: The hotel is **scheduled to open** in 1992 . (Index 30045)

Sentence Example 2: When the hacker moved , Mr. Stoll moved too , calling up other systems managers to alert them but keeping **his own system open** to avoid arousing suspicion . (Index 35893)

Table 8: Part of Sentence 30045 Analysis

Word	**scheduled**	**to**	**open**
Gold Tag	VERB	PART	VERB
HMM Tag	PROPN	PART	VERB
K-means Tag	VERB	PART	VERB

Table 9: Part of Sentence 35893 Analysis

Word	**his**	**own**	**system**	**open**
Gold Tag	PRON	ADJ	NOUN	ADJ
HMM Tag	DET	ADJ	NOUN	VERB
K-means Tag	PRON	VERB	VERB	ADJ

In the first sentence, the word "open" is correctly classified as a VERB by both models. However, in the second example, "open" functions as an adjective modifying "system" and should be tagged as ADJ. As anticipated, the BERT + K-means model successfully recognized this contextual difference and classified it correctly, whereas the HMM model misclassified "open" as a VERB.