# Unsupervised Learning for PoS Tagging

zz458

## 1 Introduction

Part-of-speech (PoS) tagging [1] is a fundamental task in natural language processing (NLP), crucial for many downstream applications [2]. The complexity of language rules necessitates machine learning (ML) methods. [3]. To address the challenges of costly manual data annotation [4], unsupervised learning (UL) techniques have emerged [5], leveraging unlabeled data effectively.

This project explores unsupervised PoS tagging on the Penn Treebank dataset using two approaches, HMMs and K-means. The HMM is trained with Expectation-Maximisation (EM) algorithms, while K-means employs BERT-generated word embeddings for clustering. The results indicate that both approaches effectively distinguish coarse-grained PoS categories, with improved performance on the XPOS task due to reduced ambiguity. However, both methods face challenges with class imbalance in the UPOS task, as well as issues like irrelevant features affecting K-means clustering and local minima in the HMM optimisation.

## 2 Methods

### 2.1 Task definition

Part-of-speech (PoS) tagging is a sequence labeling task where the input sentence $\mathbf{x} = [x_1, \cdots, x_l]$ is assigned PoS labels, resulting in predicted labels $\mathbf{c} = [c_1, \cdots, c_l]$ that correspond to the true labels $\mathbf{z} = [z_1, \cdots, z_l]$. The model is trained on a labeled dataset and tested on a separate one.

In unsupervised learning, the algorithm generates label indices without predefined mappings to true PoS labels, ensuring consistent labeling for the same PoS categories and distinct labels for different categories.

In this project, the models are provided with the total number of possible labels in advance. To manage computational complexity and simplify the problem, individual sentences are used as the unit of input for both training and testing phases.

### 2.2 Hidden Markov Model

This project employs Hidden Markov Models (HMM) with the Expectation-Maximisation (EM) algorithm for unsupervised training.

In supervised learning, part-of-speech (PoS) tags are provided for each sentence, enabling Maximum Likelihood Estimation (MLE) using tag counts. In unsupervised learning, only token sequences and the number of possible tags are available. The EM algorithm is then used to maximise the likelihood of the observed data iteratively:

1. E-step: Compute posterior probabilities based on current model parameters.
2. M-step: Update model parameters to maximise likelihood using these probabilities.

The EM algorithm is guaranteed to converge to a local maximum of the likelihood. Details of the EM algorithm on HMM are described in Appendix A.

The **stochastic EM algorithm** (or stepwise EM algorithm) [5] is a variant that updates parameters incrementally for each sentence using the rule $\theta := (1 - \eta) \cdot \theta + \eta \cdot \theta_{\text{new}}$, with a learning rate defined as $\eta_k = (k + 2)^{-\alpha}$ with update count $k$ and $0.5 < \alpha \leq 1.0$. A smaller $\alpha$ accelerates the forgetting of older statistics. This approach can improve convergence speed and reduce local optima risk but may introduce optimisation instability.

Table 1: Example data from `ptb-train.conllu`

| ID | FORM | LEMMA | UPOS | XPOS | FEATS | HEAD | DEPREL | DEPS | MISC |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Ms. | _ | PROPN | NNP | _ | 2 | nn | _ | _ |
| 2 | Haag | _ | PROPN | NNP | _ | 3 | nsubj | _ | _ |
| 3 | plays | _ | VERB | VBZ | _ | 0 | root | _ | _ |
| 4 | Elianti | _ | PROPN | NNP | _ | 3 | dobj | _ | _ |
| 5 | . | _ | PUNCT | . | _ | 3 | punct | _ | _ |

The HMM parameters are randomly initialised, and values are converted to a logarithmic scale to prevent numerical underflow. The **Viterbi algorithm** is used to make PoS predictions by finding the most likely sequence of hidden states based on observed data.

### 2.3 K-means algorithm

The Lloyd K-means algorithm clusters data points in an $M$-dimensional space ($x \in \mathbb{R}^M$) into $K$ groups using a distance metric, typically Euclidean distance. The process involves:

1. Initialisation: Randomly selecting $K$ points as initial centroids.
2. Assignment: Assigning each data point to the nearest centroid's cluster.
3. Update: Recalculating each centroid's position as the average of its assigned points.

This is repeated until the centroids stabilise. For predictions, a new data point is assigned to the nearest centroid's cluster.

To classify tokens into PoS categories using the K-means algorithm, we need an effective M-dimensional latent representation of the tokens. This project uses the pre-trained BERT-base-uncased model [6] to generate contextualised embeddings that reflect the token's characteristics and context. We employ the last layer outputs of BERT as the word embeddings for K-means clustering.

## 3 Experimental Setup

### 3.1 Dataset

The Penn Treebank [1] is a large corpus with over 4.5 million words, annotated with part-of-speech (PoS) information, and some sentences include skeletal syntactic structure annotations.

This project utilises a subset of the Penn Treebank formatted in the CoNLL-U standard [7], containing 39832 sentences and 989860 words, with each word represented by ten attributes outlined in Appendix B. Table 1 shows an example sentence. We use the **FORM** attribute as input to predict the **UPOS** and **XPOS** tags. The dataset is for both training and testing, focusing on words from single sentences while ignoring sentence-level dependencies due to computational constraints.

Tokenisation is essential for the BERT model. We use the BERT tokeniser to split words into tokens and align PoS tags by assigning the tag of the first token to all resulting tokens.

### 3.2 Measurement

This project uses two independent metrics to evaluate predictions from unsupervised learning models, without relying on a direct mapping to the true PoS labels. Measurements are taken for the entire dataset rather than averaged across sentences, as sentence lengths can vary and affect the results.

#### 3.2.1 Variation of Information

Variation of information (VI) measures the distance between two clusterings, defined as:

$$\mathrm{VI}(X, Y) = H(X) + H(Y) - 2I(X; Y) = H(X|Y) + H(Y|X)$$

where $H(\cdot)$ and $I(\cdot;\cdot)$ denote the information entropy and the mutual information respectively. The VI distance is bounded as $0 \leq \mathrm{VI}(X, Y) \leq \log n$, where $n$ is the number of elements in the dataset. A smaller VI distance indicates greater similarity between the two clusterings.

Table 2: Experiment results on UPOS tags

| Method | Epochs | VI | NVI | Homogeneity | Completeness | V-measure |
|---|---|---|---|---|---|---|
| HMM+MLE | - | 0.547 | 0.0774 | 0.924 | 0.922 | 0.923 |
| HMM+bEM | 20 | 5.67 | 0.764 | 0.248 | 0.226 | 0.236 |
| HMM+sEM ($\alpha = 0.6$) | 50 | 5.71 | 0.761 | 0.253 | 0.226 | 0.239 |
| HMM+sEM ($\alpha = 0.8$) | 45 | 5.11 | 0.708 | 0.299 | 0.286 | 0.292 |
| HMM+sEM ($\alpha = 1.0$) | 45 | 4.68 | 0.654 | 0.350 | 0.342 | 0.346 |
| BERT+K-means | 20 | 4.29 | 0.624 | 0.367 | 0.387 | 0.376 |

Table 3: Experiment results on XPOS tags

| Method | Epochs | VI | NVI | Homogeneity | Completeness | V-measure |
|---|---|---|---|---|---|---|
| HMM+MLE | - | 0.442 | 0.0509 | 0.950 | 0.948 | 0.949 |
| HMM+bEM | 20 | 6.06 | 0.674 | 0.337 | 0.315 | 0.326 |
| HMM+sEM ($\alpha = 0.6$) | 40 | 5.99 | 0.645 | 0.380 | 0.333 | 0.355 |
| HMM+sEM ($\alpha = 0.8$) | 25 | 6.14 | 0.665 | 0.356 | 0.316 | 0.335 |
| HMM+sEM ($\alpha = 1.0$) | 5 | 6.22 | 0.670 | 0.353 | 0.310 | 0.330 |
| BERT+K-means | 145 | 4.40 | 0.469 | 0.574 | 0.494 | 0.531 |

To address the unbounded nature of the VI distance, this project also employs a normalised version:

$$\mathrm{NVI}(X, Y) = \frac{\mathrm{VI}(X, Y)}{H(X) + H(Y)}$$

which scales the VI distance to the range $[0, 1]$. However, when both $X$ and $Y$ contain only one element (e.g. when the sentence has a single token), the denominator becomes zero, leading to a divide-by-zero error. In such cases, the normalised VI is defined as zero since $X$ and $Y$ are trivially matched.

#### 3.2.2 V-measure

The V-measure [8] evaluates clustering quality by considering two criteria:

- Homogeneity: A cluster contains only elements from a single true class;
- Completeness: All elements of a true class are assigned to the same cluster.

The V-measure is the weighted harmonic mean of homogeneity and completeness, detailed in Appendix C. It is bounded within $[0, 1]$, with higher values indicating greater similarity between the two clusterings.

The V-measure is mathematically equivalent to the normalised mutual information, as shown in Appendix C. This equivalence highlights its suitability for unsupervised evaluation tasks.

# 4 Results

## 4.1 HMM results

This project experiments with the HMM using both batch and stochastic EM algorithms. We allocate 5% of the dataset as a validation set for hyperparameter tuning. The V-measure is utilised to identify the optimal number of EM iterations and the appropriate value of $\alpha$ for the exponential update factor of the stochastic EM algorithm.

As illustrated in Figure 1, the V-measure for batch EM method stops increasing after the 20th iteration. Based on these validation results, we will use the model checkpoint from the 20th EM iteration for the final evaluation of the batch EM.

Figure 2 presents the validation results for the stochastic EM algorithm with $\alpha = 0.6$, 0.8, and 1.0, respectively. We assess each value of $\alpha$ using the model checkpoint that achieved the highest V-measure.

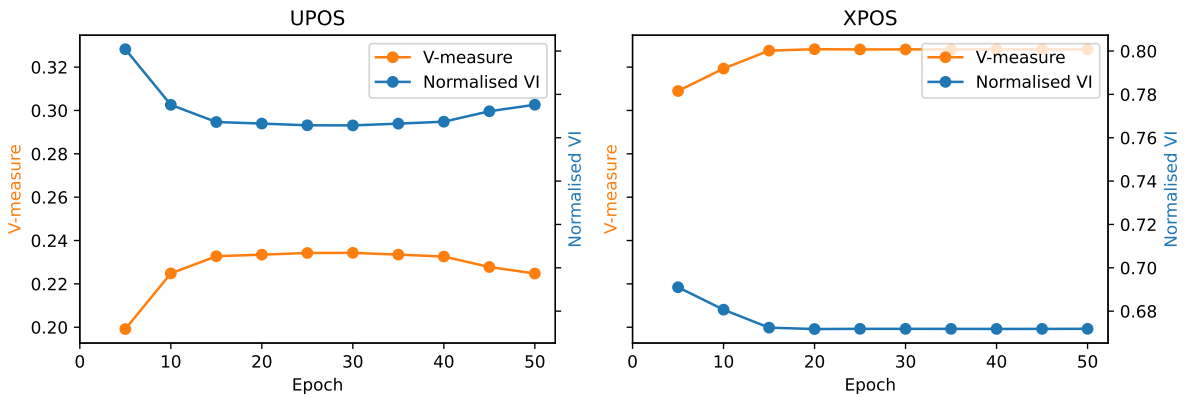



Figure 1: Validation results with HMM and batch EM algorithm on 5% of the dataset

The final results of HMMs are presented in Tables 2 and 3. Consistent with the validation findings, stochastic EM with a larger $\alpha$ performs better on UPOS tags, with $\alpha = 1.0$ surpassing $\alpha = 0.6$ by 0.107 in V-measure. This improvement is attributed to the faster forgetting of old statistics, which accelerates the optimisation of easier tasks. Conversely, stochastic EM with a smaller $\alpha$ performs better on XPOS tags, with $\alpha = 0.6$ topping $\alpha = 1.0$ by 0.025 in V-measure. In this case, the more complex XPOS task benefits from retaining previous statistics, which helps in future updates. Meanwhile, models with larger $\alpha$ values struggle to improve because they fail to accumulate sufficient information over iterations.

When compared to batch EM, stochastic EM demonstrates improved performance, with V-measure enhancements ranging from 0.003 to 0.110. This advantage is due to the stochastic nature of the method, which benefits from increased training iterations by providing more opportunities to escape local minima and achieve further improvements.

When comparing results on UPOS and XPOS, we find that HMMs perform better on XPOS. This indicates that a larger set of more fine-grained tags (XPOS) provides clearer distinctions, which helps reduce ambiguity in hidden state transitions compared to the coarser UPOS tags. HMMs effectively capture these detailed dependencies, leading to improved outcomes. However, an exception is the stochastic EM with $\alpha = 1.0$ on XPOS, as its rapid forgetting of previous updates hinders it from sophisticated learning.

## 4.2 K-means results

Hyperparameter tuning for the K-means algorithm is performed using the entire dataset, due to its unsupervised nature and low computational cost. We run the algorithm for a fixed number of iterations as well as until it converges, defined by a Frobenius norm of the centroid difference smaller than $1 \times 10^{-4}$.

Figure 3 displays the validation outcomes of the K-means algorithm. The algorithm begins to overfit the UPOS tagging task after the 20th iteration. In contrast, the K-means optimisation converges effectively on the XPOS tagging task without overfitting. We will use these two model checkpoints for the final evaluation.

The results presented in Tables 2 and 3 demonstrate the clear advantages of the K-means method, which outperforms other unsupervised approaches with HMM and EM algorithms. The superiority of K-means can be attributed to the enhanced capability of the BERT model, which effectively captures long-term dependencies. In addition, the larger embedding vector space allows for more complex knowledge to be explicitly encoded within a single vector.

Nevertheless, the K-means method performs noticeably worse than the supervised MLE method. This performance gap arises because the embeddings generated by BERT include irrelevant semantic information that interferes with clustering for PoS tags. Possible solutions to this issue include employing unsupervised continual training or supervised fine-tuning BERT in a supervised manner for PoS-specific tasks. Despite this limitation, K-means still achieves a reasonable level of accuracy due to the correlations between a word's PoS and its semantic meaning.

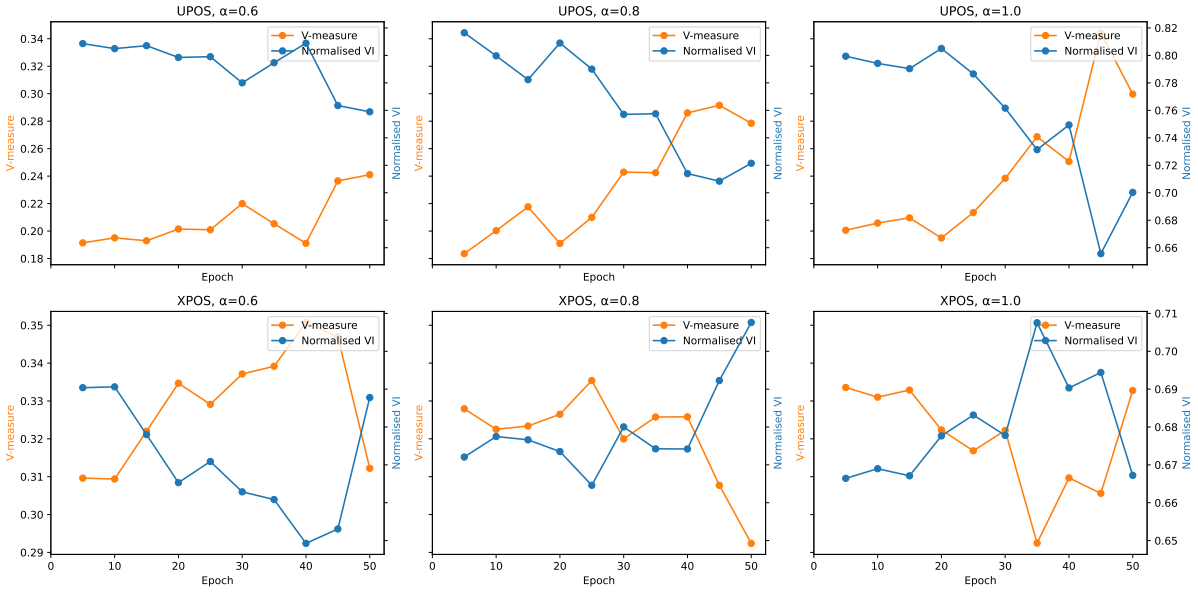



Figure 2: Validation results with HMM and stochastic EM algorithm on 5% of the dataset

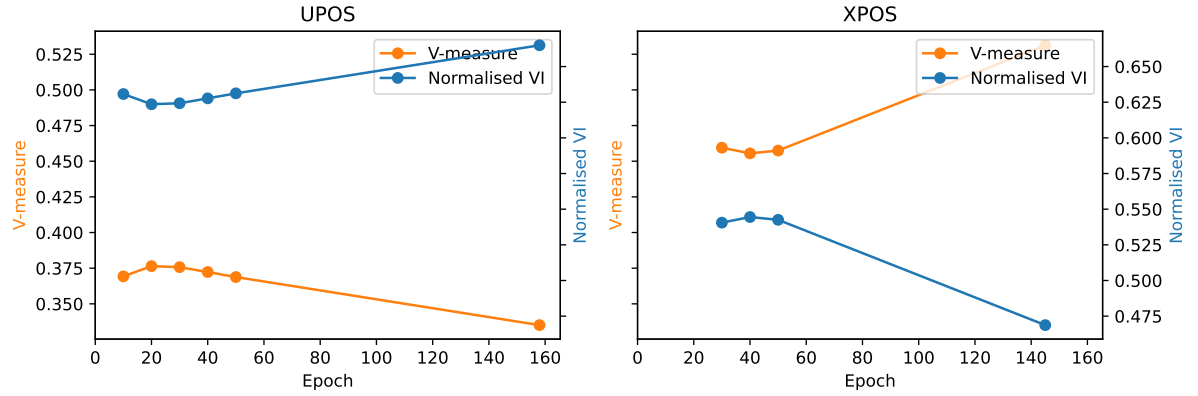
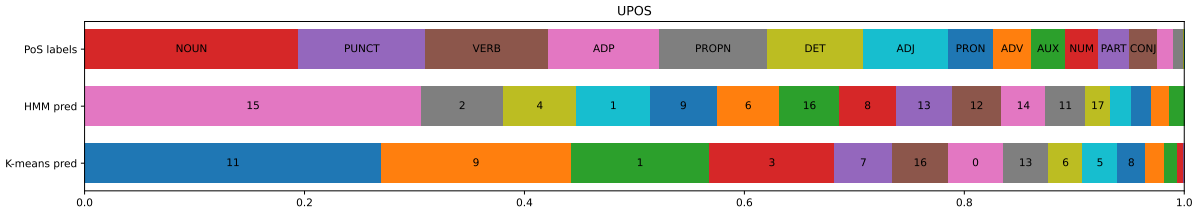
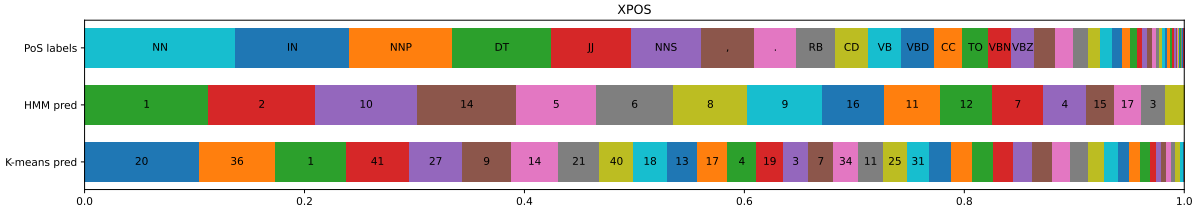



Figure 3: Results with BERT model and K-means algorithm




(a) Frequencies on UPOS




(b) Frequencies on XPOS

Figure 4: The frequencies of PoSes and predicted classes for UPOS and XPOS

Table 4: An example sentence showcasing K-means predicted classes alongside UPOS tags. Words are colour-coded according to their predicted classes, with content words emphasized in **bold**. The red, orange, and yellow classes align well with the `NOUN`, `VERB`, `PROPN`, and `ADJ` tags, illustrating the clustering's effectiveness in capturing key PoS categories.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Pred | A | **SEC** | **proposal** | to | **ease** | **reporting** | **requirements** |
| UPOS | DET | **PROPN** | **NOUN** | PART | **VERB** | **NOUN** | **NOUN** |
| Pred | for | some | **company** | **executives** | would | **undermine** | the |
| UPOS | ADP | DET | **NOUN** | **NOUN** | AUX | **VERB** | DET |
| Pred | **usefulness** | of | **information** | on | **insider** | **trades** | as |
| UPOS | **NOUN** | ADP | **NOUN** | ADP | **NOUN** | **NOUN** | ADP |
| Pred | a | **stock-picking** | tool | , | **individual** | **investors** | and |
| UPOS | DET | **ADJ** | **NOUN** | PUNCT | **ADJ** | **NOUN** | CONJ |
| Pred | **professional** | **money** | **managers** | **contend** | . | | |
| UPOS | **ADJ** | **NOUN** | **NOUN** | **VERB** | PUNCT | | |

# 5 Analysis and Case Studies

## 5.1 Number of training iterations

The V-measure from the **batch EM algorithm** demonstrates stable convergence, as illustrated in Figure 1. This stability arises because the batch EM updates the model parameters by aggregating posterior probabilities over the entire dataset. However, after the 20th iteration, the V-measure no longer increases due to the limited capacity of the HMM model. In the UPOS task, we observe a decline in the V-measure starting from the 40th iteration, likely due to overfitting given the small size of the training dataset. As a result, the model may be trapped in a suboptimal local minimum.

Figure 2 displays the result of the **stochastic EM algorithm** with $\alpha = 0.6$, 0.8, and 1.0, respectively. Compared to batch EM, the V-measure optimisations tend to be more unstable because updates occur with each sentence. However, the stochastic EM algorithm typically yields better validation results. This improvement likely stems from its stochastic nature, which promotes exploration of the parameter space. As a result, the optimisation process can escape local minima and achieve further enhancements when run for additional iterations.

The training curves of the **K-means algorithm** are shown in Figure 3. K-means begins to overfit the UPOS tagging task after the 20th iteration, whereas for XPOS, the K-means optimisation converges effectively without overfitting, achieving a better V-measure. One possible explanation for this difference is that the more complex nature of the XPOS task helps prevent overfitting. Moreover, using a larger value of $K$ allows K-means to attain a more suitable clustering of the word embeddings. This observation further suggests that the embeddings generated by BERT contain more detailed information than just the general PoS categories, thereby assisting K-means on the XPOS task. Additionally, increasing the number of iterations gives K-means more time to align with the fine-grained XPOS tags.

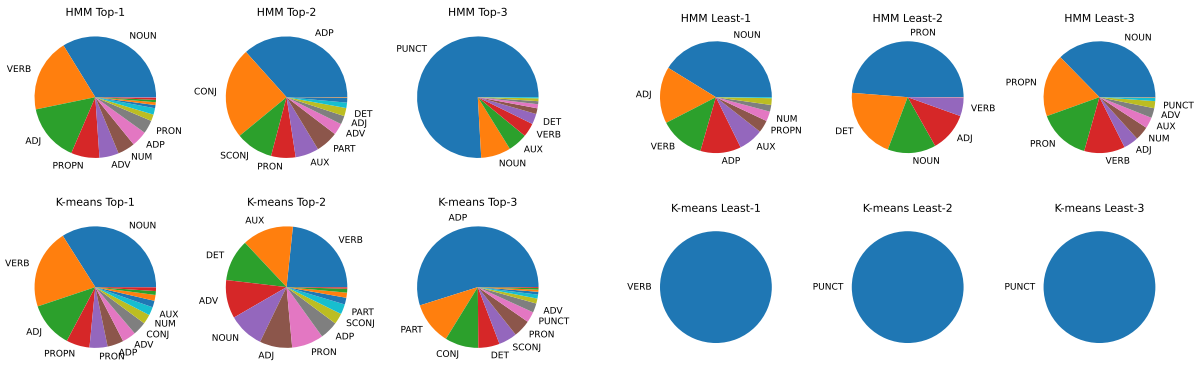
## 5.2 Influence of stochastic EM's $\alpha$ value

Due to the limited number of training iterations, only the stochastic EM with $\alpha = 1.0$ converged after 50 iterations on UPOS, as illustrated in Figure 2. A larger $\alpha$ allows for quicker advancement in this relatively easy task. However, $\alpha = 1.0$ tends to overfit the XPOS task quickly. This occurs because it rapidly disregards older statistics, which prevents the accumulation of sufficient information over iterations and inhibits further improvements.

## 5.3 Case studies

In specific sentences, K-means effectively separates content words from function words, as shown in Table 4, though it struggles to differentiate content words into finer categories like nouns and verbs. This suggests that embeddings capture major distinctions, but irrelevant features blur finer details.

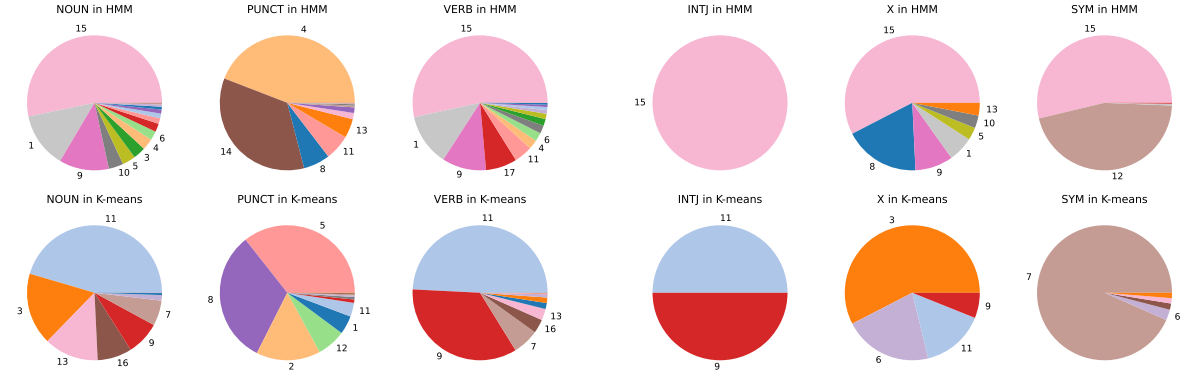
HMM exhibits a consistent issue: it assigns the same hidden state to the first word of each sentence in both UPOS and XPOS predictions. This error indicates that HMM's training often falls into local minima, revealing its sensitivity to initialisation and optimisation techniques.




(a) UPOS in top three predicted classes

(b) UPOS in last three predicted classes

Figure 5: Homogeneity, or the ratios of PoSes assigned to each of the top three most or least frequent predicted classes for UPOS




(a) Predicted classes in top three UPOS

(b) Predicted classes in last three UPOS

Figure 6: Completeness, or the ratios of predicted classes on words with each of the top five most or least frequent PoSes for UPOS. The same colour represents the same predicted class for each model.

Specifically, punctuations sharing the same `PUNCT` tag in UPOS are often divided into multiple classes, despite being clearly separated from words. This indicates that contextual information encoded by the HMM and BERT model, particularly at sentence endings, introduces variability that complicates accurate PoS predictions for punctuation.

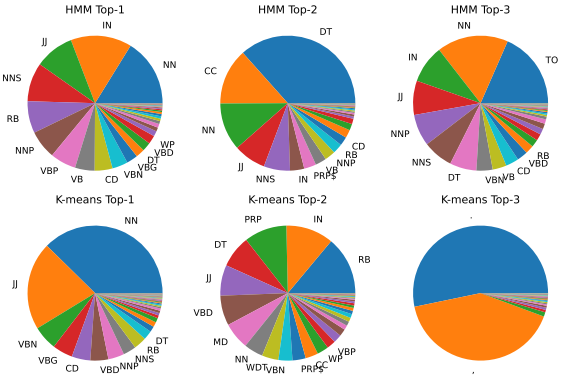
## 5.4 Statistical insights

Prediction statistics provide further insights. For UPOS, as shown in Figure 4a, both HMM and K-means disproportionately assign many words to a single class. The most frequent class nearly matches the combined counts of the two most common PoS tags, `NOUN` and `PUNCT`. In contrast, XPOS predictions, shown in Figure 4b, are more balanced, correlating with the models' better performance on XPOS.

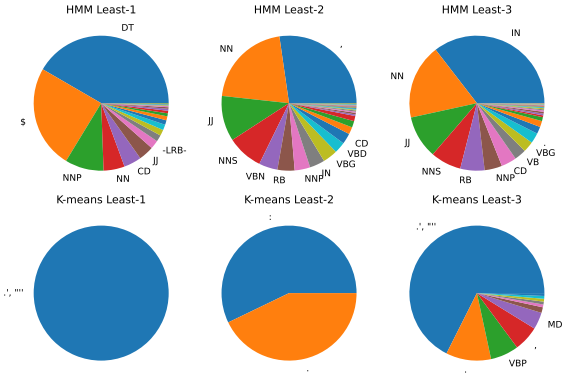
Visualising homogeneity and completeness, Figure 5a shows that frequent classes primarily capture either content or function words. Similarly, Figure 6a reveals that frequent content PoS tags are often grouped into the same predicted classes, confirming that the models distinguish broad PoS categories well.

In Figure 5b, K-means dedicates an entire class to a single PoS, while HMM does not achieve this consistently. Notably, K-means assigns `VERB` tags across both the most and least frequent classes, suggesting that the diverse semantic properties of verbs, unrelated to their PoS, have disrupted its classification. Additionally, Figure 6b reveals that both models group some function words with content words, illustrating the drawback highlighted in Figure 4a, where too many words are clustered into a single class.

Examining the homogeneity and completeness of the XPOS task, the increased number of classes allows both models to allocate distinct groups for various punctuation marks, as shown in Figures 7b and 8b.

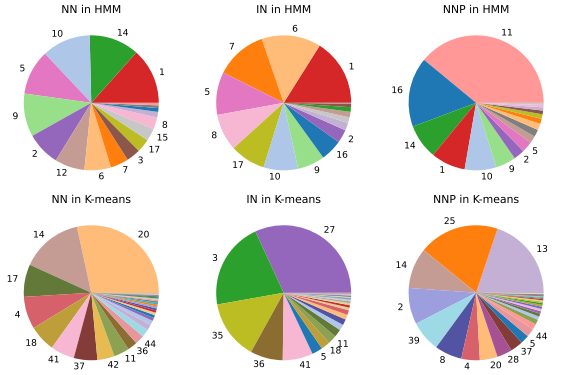



(a) XPOS in top three predicted classes

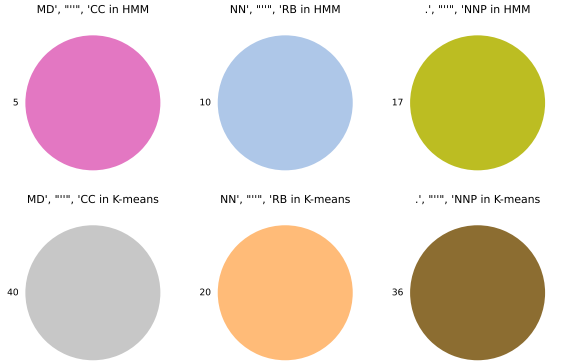



(b) XPOS in last three predicted classes

Figure 7: Homogeneity, or the ratios of PoSes assigned to each of the top three most or least frequent predicted classes for XPOS




(a) Predicted classes in top three XPOS




(b) Predicted classes in last three XPOS

Figure 8: Completeness, or the ratios of predicted classes on words with each of the top five most or least frequent PoSes for XPOS. The same colour represents the same predicted class for each model.

However, as content words are divided into more fine-grained PoS tags, both models face challenges in aligning PoS tags with predicted classes on a one-to-one basis. Notably, K-means assigns its top three predicted classes to different groups of PoS tags and distributes the top three PoS tags across separate classes, which helps explain its superior performance among the unsupervised learning approaches.

In general, the analysis reveals that both HMM and K-means perform better on the detailed XPOS task, as finer distinctions reduce ambiguity. K-means benefits from semantic correlations in BERT embeddings, while HMM captures nuanced dependencies in XPOS. However, challenges persist: HMM is sensitive to initialisation and struggles with local minima, while K-means is affected by irrelevant embedding features. Statistical evaluations reveal class imbalance issues, particularly in UPOS, where frequent PoS tags dominate predictions. Both methods face difficulties with punctuation and fine-grained PoS distinctions, highlighting areas for refinement.

# 6 Conclusion

This project evaluates two unsupervised learning methods–HMM with EM algorithms and K-means–on the PoS tagging task. While both approaches demonstrate the ability to train models without supervision effectively, they face challenges in avoiding suboptimal solutions and mitigating the influence of irrelevant information inherent to the unsupervised setup. These limitations highlight the need for further exploration of optimisation strategies and feature representations to improve performance in unsupervised PoS tagging.

# References


[1] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993. URL https://aclanthology.org/J93-2004.

[2] Hongwei Li, Hongyan Mao, and Jingzi Wang. Part-of-speech tagging with rule-based data preprocessing and transformer. *Electronics*, 11(1), 2022. ISSN 2079-9292. doi: 10.3390/electronics11010056. URL https://www.mdpi.com/2079-9292/11/1/56.

[3] Tatwadarshi P. Nagarhalli, Vinod Vaze, and N. K. Rana. Impact of machine learning in natural language processing: A review. In *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*, pages 1529–1534, 2021. doi: 10.1109/ICICV50876.2021.9388380.

[4] Minzhi Li, Taiwei Shi, Caleb Ziems, Min-Yen Kan, Nancy Chen, Zhengyuan Liu, and Diyi Yang. CoAnnotating: Uncertainty-guided work allocation between human and large language models for data annotation. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1487–1505, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.92. URL https://aclanthology.org/2023.emnlp-main.92.

[5] Percy Liang and Dan Klein. Online EM for unsupervised models. In Mari Ostendorf, Michael Collins, Shri Narayanan, Douglas W. Oard, and Lucy Vanderwende, editors, *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 611–619, Boulder, Colorado, June 2009. Association for Computational Linguistics. URL https://aclanthology.org/N09-1069.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. URL https://arxiv.org/abs/1810.04805.

[7] Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D. Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. Universal Dependencies v2: An evergrowing multilingual treebank collection. In Nicoletta Calzolari, Frédéric Béchet, Philippe Blache, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4034–4043, Marseille, France, May 2020. European Language Resources Association. ISBN 979-10-95546-34-4. URL https://aclanthology.org/2020.lrec-1.497.

[8] Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In Jason Eisner, editor, *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 410–420, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL https://aclanthology.org/D07-1043.

# A EM algorithm with HMM

An HMM is characterised by the following components:

- $N$: the number of possible hidden states;
- $M$: the number of possible observations;
- $A \in \mathbb{R}^{N \times N}$: the state transition probability matrix;
- $B \in \mathbb{R}^{N \times M}$: the emission probability matrix;
- $\pi \in \mathbb{R}^{N}$: the initial state probability vector.

For HMM, the E-step of the EM algorithm uses the forward-backward algorithm to calculate the posterior probabilities of being in state $s$ at timestep $t$ and transitioning from state $s$ at timestep $t$ to state $s'$ at $t+1$, using the forward probabilities $\alpha_t(s) = p(s_t|o_1, \cdots, o_t)$ and the backward probabilities $\beta_t(s) = p(s_t|o_{t+1}, \cdots, o_T)$:

$$\gamma_t(s) := p(s_t|O) = \frac{\alpha_t(s) \cdot \beta_t(s)}{\sum_{p \in S} \alpha_t(p) \cdot \beta_t(p)}$$

$$\xi_t(s, s') := p(s_t, s'_{t+1}|O) = \frac{\alpha_t(s) \cdot A_{s,s'} \cdot B_{s',o_{t+1}} \cdot \beta_{t+1}(s')}{\sum_{p,q \in S} \alpha_t(p) \cdot A_{p,q} \cdot B_{q,o_{t+1}} \cdot \beta_{t+1}(q)}$$

where $S$ represents the set of possible hidden states, and $T$ is the length of the observation sequence.

For the M-step, the classic **batch EM algorithm** updates the HMM parameters by aggregating results over $B$ samples:

$$A_{s,s'} := \frac{\sum_{k=1}^{B} \sum_{t=1}^{T_k - 1} \xi_t^{(k)}(s, s')}{\sum_{k=1}^{B} \sum_{t=1}^{T_k - 1} \sum_{q \in S} \xi_t^{(k)}(s, q)} = \frac{\sum_{k=1}^{B} \sum_{t=1}^{T_k - 1} \xi_t^{(k)}(s, s')}{\sum_{k=1}^{B} \sum_{t=1}^{T_k - 1} \gamma_t^{(k)}(s)}$$

$$B_{s,o} := \frac{\sum_{k=1}^{B} \sum_{t=1}^{T_k} \mathbb{1}_{o = O_t^{(k)}} \cdot \gamma_t^{(k)}(s)}{\sum_{k=1}^{B} \sum_{t=1}^{T_k} \gamma_t^{(k)}(s)}$$

$$\pi_s := \frac{\sum_{k=1}^{B} \gamma_1^{(k)}(s)}{\sum_{k=1}^{B} \sum_{p \in S} \gamma_1^{(k)}(p)}$$

# B CoNLL-U Format

Each word in the CoNLL-U format has ten attributes:

1. **ID**: The word index, starting from 1. This field may include a range for multi-word tokens or decimals for empty nodes.
2. **FORM**: The surface form of the word or punctuation mark.
3. **LEMMA**: (Unavailable in this subset) The lemma or stem of the word.
4. **UPOS**: Universal PoS tags, comprising 17 categories. These include six open classes (e.g., `ADJ` for adjectives, `VERB` for verbs, `NOUN` for nouns) and eight closed classes (e.g., `ADP` for adpositions, `DET` for determiners, `CCONJ` for coordinating conjunctions).
5. **XPOS**: Language- or treebank-specific PoS tags, which include 36 fine-grained PoS tags and 12 additional tags.
6. **FEATS**: (Unavailable in this subset) A list of morphological features for the token.
7. **HEAD**: The head of the token in the dependency tree, represented by its word index or zero.
8. **DEPREL**: The dependency relation between the token and its head.
9. **DEPS**: (Unavailable in this subset) An enhanced dependency graph represented as a list of head-dependency relation pairs.
10. **MISC**: Miscellaneous annotations.

# C Details of V-measure Calculation

The V-measure takes the weighted harmonic mean of the two measures:

$$
\begin{aligned}
v &= \frac{\alpha+\beta}{\frac{\alpha}{h}+\frac{\beta}{c}} \\
&= \frac{(\alpha+\beta)\times h\times c}{\beta\times h+\alpha\times c} \\
h &= \begin{cases} 1 & H(C,K)=0 \\ 1-\frac{H(C|K)}{H(C)} & \text{otherwise} \end{cases} \\
c &= \begin{cases} 1 & H(K,C)=0 \\ 1-\frac{H(K|C)}{H(K)} & \text{otherwise} \end{cases}
\end{aligned}
$$

where $H(\cdot)$ represents the information entropy, $C$ denotes the true classes, $K$ denotes the clusters, and $\alpha$ and $\beta$ denotes the weights for the harmonic mean. In this project, we take $\alpha=\beta=1$.

The following shows the equivalence of the V-measure and the normalised mutual information.

$$
\begin{aligned}
v &= \frac{(\alpha+\beta)\times h\times c}{\beta\times h+\alpha\times c} \\
&= \frac{(\alpha+\beta)\times(1-\frac{H(C)-I(C;K)}{H(C)})\times(1-\frac{H(K)-I(K;C)}{H(K)})}{\beta(1-\frac{H(C)-I(C;K)}{H(C)})+\alpha(1-\frac{H(K)-I(K;C)}{H(K)})} \\
&= \frac{(\alpha+\beta)\times\frac{[I(C;K)]^2}{H(C)H(K)}}{\frac{\beta\times I(C;K)}{H(C)}+\frac{\alpha\times I(C;K)}{H(K)}} \\
&= \frac{(\alpha+\beta)\times I(X;Y)}{\alpha\times H(C)+\beta\times H(K)} \\
&= \mathrm{NMI}(X,Y)
\end{aligned}
$$