CS5012-D1 : Parts Of Speech Tagging

Matriculation ID : 190015412
Mark-Jan Nederhof

March 6, 2024

Contents

1 Introduction **1**

2 HMM Training **1**

3 HMM Algorithms **2**
3.1 Setup and Infrastructure . . . 2
3.2 Eager Algorithm . . . 2
3.3 Viterbi Algorithm . . . 3
3.4 Individual Most Probable Tag Algorithm . . . 3

4 Evaluation **4**
4.1 Chosen Metrics . . . 4
4.2 Universal Dependency Corpora . . . 4
4.3 Accuracy . . . 4
4.4 Comparison of Languages . . . 4
4.4.1 Korean . . . 4
4.4.2 Chinese . . . 5
4.4.3 Swedish . . . 5
4.4.4 Differences in Accuracy For Each Algorithm . . . 5

5 Conclusion **6**

6 Appendix **7**
6.1 Tag Frequencies . . . 7
6.2 Confusion Matrix . . . 8

1 Introduction

The purpose of this practical is to explore the use of Hidden Markov Models (HMM) for the task of part-of-speech (POS) tagging of sentences using the Universal Dependencies tagset and treebanks. In particular, we explore and compare the functionality and performance of three different HMM based POS tagging algorithms on a given corpus of sentences: eager algorithm, the Viterbi algorithm, and individually most probable tag (IMPT) algorithm. We additionally explore how the relative performance between each algorithm changes when applied to different languages in addition to English. In my experiments, I evaluated the performance of the three algorithms against the English, Swedish, Korean, and Chinese corpora.

2 HMM Training

For this practical, we work with first order HMM POS-tagging algorithms. In particular, this defines the following assumptions for estimating emission and transition probabilities:

1) Probability of an emission (token) only depends on its corresponding part-of-speech tag

$$P(w_1^n|t_1^n) = \prod_{i=1}^{n} P(w_i|t_i) \tag{1}$$

2) Probability of a given state (tag) only depends on its immediate previous state

$$P(t_1^n) = (\prod_{i=1}^{n} P(t_i|t_{i-1}))P(t_{n+1}|t_n) \tag{2}$$

In order to estimate transition and emission probabilities to "train" our HMM algorithms, we perform relative frequency estimation over a given training corpus of tagged sentences from the Universal Dependencies treebank. In base form, relative frequency estimation of transitions and emissions can be given by the following:

Transitions

$$P(t_i|t_{i-1}) = \frac{C(t_{i-1}t_i)}{C(t_{i-1}} \tag{3}$$

Emissions

$$P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i} \tag{4}$$

To implement relative frequency estimation for transitions, using the `ngrams` and `FreqDist` utilities of the `nltk` library, I created a frequency distribution over all tag-bigrams in the given training corpus – in particular, for each tag T, I created a frequency distribution over all possible tags to follow T. For training emissions, I again use the `FreqDist` function to create a frequency distribution over all possible tags and their corresponding emissions in the training corpus.

In order to obtain emission and transition probabilities, their frequency distributions must be normalised. To account for the sparse data problem, I am using the `WittenBellProbDist` function of the `nltk` library to normalise the frequency distributions with Witten-Bell smoothing which defines the probability of an event as

$$\begin{aligned} &P_{WB}(w_i|w_{i-1}) = \\ &\quad \lambda_{w_{i-1}} P(w_i|w_{i-1}) + (1 - \lambda_{w_{i-1}}) P_{WB}(w_i) \end{aligned} \tag{5}$$

where

$$\lambda_{w_{i-1}} = \frac{C(w_{i-1})}{C(w_{i-1}) + |v|C(w_{i-1}) > 0|}$$

This adjusts estimated transition and emission probabilities to account for events that have not yet been observed in the training data to avoid zero probabilities. Like many other smoothing techniques, Witten-Bell smoothing aims to re-distribute probability mass from observed events to unseen events. It is a suitable smoothing technique for natural language modelling, which contains large vocabularies, due to its mixed usage of interpolation and backoff, allowing for a nuanced and dynamic smoothing approach.

In particular, for first order HMM, Witten-Bell smoothing defines the probability of an event as a weighted combination of a finer model (bigram) and a coarser model (Witten-Bell unigram). In estimating the probability of an event w, the weighting of the two models is dependent on the right-context of the previous event w_{i-1}. If w_{i-1} has a small right context, it means it is more useful/reliable for predicting w_i, and therefore more weight is delegated to the finer bigram model in calculating the probability. Conversely, If w_{i-1} has a rather diverse (large) right context, it means it is more difficult to predict w_i given w_{i-1}, therefore the method 'backs off' (gives more weight) to the coarser unigram model. The unigram model uses a form of smoothing similar to Good-Turing smoothing given by

$$P_{WB}(t) = \begin{cases} \frac{C(w)}{n+m}, & C(w) > 0 \\ \frac{m}{z(n+m}, & otherwise \end{cases} \tag{6}$$

where

n : normal tokens
m : pseudo events where next token unseen so far
z : number of types unseen so far

So the unigram model essentially makes the assumption that the probability of a given event is related to the diversity of words in the corpus (number of unique events) so far, scaling the allocation of probability mass based on this diversity.

I implemented the Witten-Bell probability distributions for transitions and emissions using dictionaries. For my transitions dictionary, keys are tags T_i where each corresponding value is a probability distribution over all possible tags T_{i+1} indicating the probability of T_{i+1} following T_i. Similarly, for the emissions dictionary, keys are tags and values are probability distributions over the vocabulary indicating the probability of a given tag emitting each word of the vocabulary.

3 HMM Algorithms

3.1 Setup and Infrastructure

I decided to implement automatic tagging functionality for all three algorithms in a class called `Tagger`. Upon initialising a `Tagger` object for a particular corpus of the given Universal Dependency Treebank data, all sentences in the training and testing data of the corpus are pre-processed by adding start-of-sentence and end-of-sentence markers to each sentence and converting each token into a tuple of form `(token, tag)`. Then emission and transition probability distributions are created as described in the previous section.

3.2 Eager Algorithm

The eager algorithm for this practical is a naive HMM-based algorithm. This means it utilises a more localised approach for predicting tags that doesn' t consider much context around which a given word appears. In particular, this algorithm predicts tags for tokens based only on the tag of the previous token and is unaware of the rest of the sentence for which a token belongs to. This eager algorithm is given by

$$\hat{t}_i = \arg\max_{t_i} P(t_i|\hat{t}_{i-1}) * P(w_i|t_i) \tag{7}$$

So for a given token w_i, this algorithm finds the tag t_i which produces the highest probability of following its previous tag t_{i-1} and emitting the token w. As a more localised algorithm, Eager algorithm is less complex to compute and relatively efficient to set up and execute.

3.3 Viterbi Algorithm

The Viterbi algorithm can be viewed as a more nuanced extension of the eager algorithm. For a token w_i, rather than only use the tag of the previous token to predict its tag t_i, the Viterbi algorithm finds the most probable sequence (path) of tags starting from the very first token position of the sentence to token w_i. The algorithm is given by

$$\hat{t}_1 ... \hat{t}_n = \arg\max_{t_i ... t_n} (\prod_{i=1}^{n} P(t_i|t_{i-1}) P(w_i|t_i)) P(t_{n+1}|t_n) \tag{8}$$

The algorithm maintains a table, `viterbi[q,i]`, denoting the probability of the best path to get to state (tag) q at position i of a sentence/input. I have implemented the table as a list of dictionaries. Each dictionary corresponds to an input position i, and stores for each tag possible tag t_i for position i, the highest probability achievable for each possible previous tag t_{i-1}. This is done for each step of the forward pass of the algorithm. To find the most probable sequence of tags for a sentence, after building the table, the algorithm must perform backtracking through the table:

$$q_n = \arg\max_{q'} viterbi[q', n] * \alpha(q', q_F)$$

and for $i = n, n-1, ..., 2$

$$q_{i-1} = \arg\max_{q'} viterbi[q', i-1] * \alpha(q', q_i) * \beta(q_i, w_i) \tag{9}$$

To implement this backtracking functionality, I am using a dedicated backtracking matrix, implemented as a list of dictionaries. As a form of dynamic programming, this backtracking matrix is updated alongside the construction of the Viterbi table, assembling the information necessary to recover the path of tags with the highest probability. Each dictionary of the list corresponds to an input position i in a given sentence and stores for each possible tag t_i the most probable previous tag t_{i-1}. This information is used to backtrack from the last observation (token) to the first, reconstructing the most probable sequence of tags after the forward pass through of the Viterbi algorithm.

3.4 Individual Most Probable Tag Algorithm

The Individual Most Probable Tag (IMPT) algorithm computes the most probable tag for each token of a sentence by considering forward and backward probabilities as defined by the Welch-Baum algorithm. For each position i in a sentence, the algorithm computes the most probable tag t_i as

$$\hat{t}_i = \arg\max_{t_i} \sum_{t_1 ... t_{i-1}} \left(\prod_{k=1}^{i} P(t_k|t_{k-1}) P(w_k|t_k) \right) * \sum_{t_{i+1} ... t_n} \left(\prod_{k=i+1}^{n} P(t_k|t_{k-1}) P(w_k|t_k) \right) * P(t_{n+1}|t_n) \tag{10}$$

So for each possible tag t_i at position i, we combine (multiply) the forward probability up to position i with the backward probability from position $i+1$ to the end of the sentence. We then sum over all possible sequences of tags that include tag t_i at position i. The tag, t_i, that maximises this sum of products is chosen as the most probable tag for that position.

This algorithm computes the most probable tag for each position independently. It takes into account all possible paths that pass through each position and their probabilities (essentially collecting marginals), unlike the Viterbi algorithm which aims to find a single connected path of tags with the highest probability (essentially maximizing joint probability of a path of tags). Another way of viewing the IMPT algorithm is that it computes a weighted sum of all possible paths through a position i for each possible tag t_i and chooses the tag with the highest weighted sum at each position. Furthermore, the IMPT algorithm is in a way bi-directional compared to the previous two algorithms as it considers context on both sides of a position i in a sentence when determining the most probable tag for the token at that position.

The IMPT algorithm does not take into account the notion of sequence coherence. In other words, by only predicting the most probable tag at each position of a sequence individually (as described in section 3.4), it does not necessary produce the most likely sequence of tags, and could in fact theoretically produce a sequence of tags that is very unlikely when the tags are viewed in conjunction [6].

The IMPT algorithm can be described as a posterior decoding problem[1]. It incorporates Bayesian principles of considering both prior and posterior probabilities. The algorithm computes the posterior probability ($P(Hypothesis|Evidence)$) for each state (tag) at each position, updating a notion of 'belief' about the state given the total observed evidence (where evidence is gathered using the Forward and Backward algorithms)[1]. In other words, for each position i of a given sequence, we use our evidence which are the emissions (tokens) of the entire sequence to compute the most likely hypothesis that explains the evidence where the hypothesis is a particular tag for the position i.

4 Evaluation

In this section, we examine and compare the difference in performance between each algorithm within the English language, and additionally examine these differences against a number of other languages, namely Korean, Swedish and Chinese (Simplified). In doing so we examine the unique morphological and syntactical properties of these languages.

4.1 Chosen Metrics

The metrics I used to evaluate the three algorithms are accuracy and confusion matrix to offer insights into per-tag precision and recall and provide comprehensive visualizations of each algorithm's performance. Note that I only included confusion matrices for the IMPT algorithm for each language, since the IMPT algorithm on average performs the best for each language. These figures are listed in the the Appendix below in section 6.2.

The accuracy metric is rather straightforward and simply measures the proportion of correct predictions over all predictions. This can be viewed as an overall metric of each algorithm's performance. The confusion matrices show the probability that each tag is mistaken for another tag, providing insight into how differences in each language affect the predictability of certain tags.

4.2 Universal Dependency Corpora

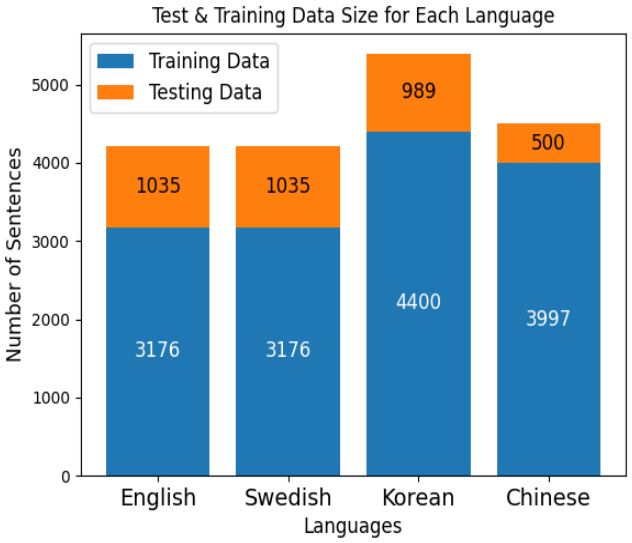

Figure 1: Testing and training data sizes for each language

Before discussing evaluation, it is important to note the differences between the Universal Dependency corpora for each of the languages I have chosen. Fig. 1 shows the differences in training and testing data size between the various languages. I considered normalizing the corpus for each language but decided against it since it seems this would require a very nuanced approach with understanding of the underlying text for each corpus.

4.3 Accuracy

	Eager	Viterbi	IMPT
English	90.03	93.09	93.07
Swedish	87.51	92.04	92.07
Korean	83.54	84.58	84.86
Chinese	84.93	85.08	85.38

Table 1: Accuracy Results (Percent)

As shown in Table 1 above, on average for each language, the IMPT algorithm has the highest accuracy, the Viterbi algorithm comes in second, and as expected the Eager algorithm has the lowest accuracy of the three.

An interesting phenomenon is that for all four languages, the Viterbi algorithm and IMPT algorithm have rather similar accuracy (often a decimal difference in percentage) despite the fact that the two algorithms approach the task of prediction rather differently. The fact that the IMPT algorithm slightly outperforms Viterbi suggests that it is the more robust algorithm. This would suggest that the joint-probability of a sequence of tags (as defined by the Viterbi algorithm) is perhaps a sub-optimal metric compared to posterior decoding for making predictions in this case.

As mentioned in section 3.4, the IMPT algorithm does have its draw backs by predicting each tag individually and not considering the coherence of transitions through the entire sequence. This could explain why in my testing, the IMPT algorithm slightly under performs compared to the Viterbi algorithm for the English corpus.

4.4 Comparison of Languages

As we can see from the accuracy results in Table 1, the algorithms have better performance when executed against the alphabet-based languages compared to the character-based languages. In this section, we analyze the differences in morphological complexities between each language as to offer an explanation for the differences in accuracy for the three POS-tagging algorithms.

4.4.1 Korean

Similar to English, Korean (Hangul) is an agglutinative language that utilizes word-space orthography [8]. This is to say that the language makes use of inflection and affixes to modify roots words in order to convey aspects such as tense or mood and separates words

using spaces in the same way that English does. Interestingly, while Korean is considered a character-based language and is deeply influenced by Chinese, it still defines a notion of consonants and vowels to assemble words. These vowels and consonants can be combined syllable blocks to form words [3].

Compared to English, Korean relies more heavily on inflection and the use of affixes to convey additional grammatical information. The language utilizes a large number of bound morphemes which can be attached in series (as suffixes in particular) to convey a number of grammatical meanings [2]. Each suffix can convey a vast number of grammatical meanings such as tense, mood, and unlike English, also includes a system for conveying respect and politeness [8][3]. While English also utilizes inflection, it relies more heavily on aspects such as word-order, prepositions, and auxiliary verbs to convey grammatical information and relationships. These aspects suggest that the Korean language is more difficult to tag due to the lack of information provided through features such as word order and inclusion of auxiliary words such as prepositions and modifiers. The task of POS-tagging for Korean would likely benefit from deeper morphological analysis of each word in order to extract information to decipher ambiguities [2].

Looking at the tag frequencies of the Universal Dependency Training corpora for each language as shown in Figures [2-5] in section 6, we can see that the distribution for the Korean training corpus is much more skewed compared to the other three languages, showing how Korean relies less on auxiliary words to convey grammatical meaning and rather has high agglutinativity.

4.4.2 Chinese

Chinese offers a different paradigm for character-based language formation and grammar. Its use of morphemes differs substantially. Rather than use inflection to convey grammatical information, the Chinese language relies on a mixture of auxiliary words, semantic knowledge, and context. In a way the language lacks a lot of formal morphological devices such as for conveying tense and plurality that are often used in other languages to provide clues for processing syntax [9]. The language inherently has more instances of ambiguity that depend on semantics and context to decipher[9][10] and is consequently less productive.

The Chinese language lacks conjugation for verbs. Certain particles can help convey grammatical information such as the word ” 了” which can follow a verb to convey past-tense, however the general lack of conjugation presents many ambiguities for POS-tagging [4]. For example, the sentence ” 我喜欢跑步”, can translate to both ”I like to run” and ”I like running” where the word ” 跑步” can mean both ”run” as a verb or ”running” as a gerund noun. In this case, there is no use of particle or other auxiliary word to help disambiguate. So for this Chinese phrase, the left context of the word for ” 跑步” (run) has no effect in deciphering this ambiguity, however there is no difference between the two interpretations semantically.

Another aspect of Chinese that complicates POS tagging is its flexibility in word-ordering. For example, English utilises the subject-verb-object order. Chinese, however, supports both subject-verb-object as well as subject-object-verb. For example, the phrase ” 你吃饭了吗?” literally translates to ”you eat food (yet)?”, however, the phrase ” 你饭吃了吗?” is identical semantically but literally translates to ”you food eat (yet)?” without the addition of any auxiliary words or affixes (simply a re-ordering of the same terms). This flexibility in word ordering means that the transitions in the Chinese language are inherently more difficult to model.

4.4.3 Swedish

Swedish is much more similar to English as the two languages are both alphabet-based and Germanic, sharing foundational linguistic elements [5]. Swedish, however, has more extensive use of inflection compared to English. For example, Swedish nouns can be inflected to both convey plurality and definiteness whereas English utilises articles to demonstrate definiteness [7]. Additionally, verbs, nouns, and adjectives must be inflected to convey gender (common, neuter) [5] in Swedish, adding more complexity to the morphological structure of words. This added complexity may explain the slight difference in accuracy of the three tagging algorithms when applied against the English corpus compared to the Swedish corpus.

4.4.4 Differences in Accuracy For Each Algorithm

Another interesting phenomena is that the difference in accuracy between the naive eager algorithm and the more informed algorithms, Viterbi and IMPT, is larger for English and Swedish compared to Korean and Chinese. In particular, the accuracy between the eager algorithm and Viterbi differ by multiple whole integer percent values for English and Swedish, whereas the corresponding differences for the Korean and Chinese are 1.04% for Korean and 0.15% for Chinese.

This could also be indicative of how Chinese and Korean are inherently more ambiguous and complex languages to model. The minimal difference in predictive accuracy between these methods may indicate that, due to the inherent ambiguity present within languages, the additional information and context considered for tagging sentences in Chinese and Korean do not significantly enhance reliability for the given amount of training data. Furthermore, this suggests that for the

amount of training data provided, a basic bigram HMM proves to be rather effective, underscoring the notion that more complex models may not always yield substantially better performance given certain linguistic complexities and dataset sizes.

5 Conclusion

Overall this practical was an interesting exploration into the performance of three different approaches of using first order HMM for the task of POS-tagging. My experiment results suggest that the IMPT algorithm is the most robust method for tag predictions, with Viterbi coming in close second. The results also align with inherent differences in morphology, complexity and ambiguity between each the four languages I chose to evaluate. If I had more time, I would have like to see the effect of changing the training data size on accuracy. In particular, I would like to see if the difference in accuracy between the naive eager algorithm and the Viterbi and IMPT algorithms increase with more training data for Korean and Chinese. Another experiment I would conduct with more time is utilising a larger tagset as to capture nuances in word forms that are specific to certain language morphologies. This would be particularly interesting for languages that utilise a high level of inflection to modify grammatical meaning of words.

References

[1] Manolis Kellis et al. *Posterior Decoding*. URL: https://web.mit.edu/6.047/book-2012/Lecture08_HMMSII/Lecture08_HMMSII_standalone.pdf.

[2] Matteson et al. "Rich character-level information for Korean morphological analysis and part-of-speech tagging". In: (2018). DOI: 1806.10771. URL: https://aclanthology.org/C18-1210/.

[3] M. Ferdock and K. Sharma. "A Brief Linguistic Analysis of the Korean Language". In: (June 2018). URL: https://ucisportfolios.pitt.edu/madelynferdock/research-2/figure1/#:~:text=Although%20Korean%20limits%20the%20order.

[4] Jerome L. Packard. *The Morphology of Chinese : A linguistic and cognitive approach*. 2000. URL: https://public.websites.umich.edu/~duanmu/RevPackard02.pdf.

[5] Erik M. Petzell. "Swedish". In: (Apr. 2023). DOI: 10.1093/acrefore/9780199384655.013.945. URL: https://doi.org/10.1093/acrefore/9780199384655.013.945.

[6] Lawrence R. Rabiner. "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition". In: *Proceedings of the IEEE* 77 (2 Feb. 1989), pp. 257–286. DOI: 10.1109/5.18626. URL: https://www.cs.ubc.ca/~murphyk/Bayes/rabiner.pdf.

[7] Renate Raffelsiefen. "Morphological word structure in English and Swedish: the evidence from prosody". In: (2005). URL: https://www.ids-mannheim.de/fileadmin/gra/texte/raf_mmm5.pdf.

[8] Jun-ichi Tsujii Sang-Zoo Lee and Hae-Chang Rim. "Hidden Markov Model-Based Korean Part-of-Speech Tagging Considering High Agglutinativity, Word-Spacing, and Lexical Correlativity". In: *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics - ACL 00* (2000). DOI: 10.3115/1075218.1075266. URL: https://aclanthology.org/P00-1048.pdf.

[9] Weiwei Sun and Xiaojun Wan. "Towards Accurate and Efficient Chinese Part-of-Speech Tagging". In: *Computational Linguistics* 42 (3), pp. 391–419. DOI: 10.1162/coli_a_00253.

[10] Zhai Xun and Yoo Eun Hee. "Comparative Study on the Structures of Chinese and Korean Compound Words". In: *Lingua Cultura* 10 (1 May 2016). DOI: 10.21512/lc.v10i1.884.

6 Appendix

6.1 Tag Frequencies

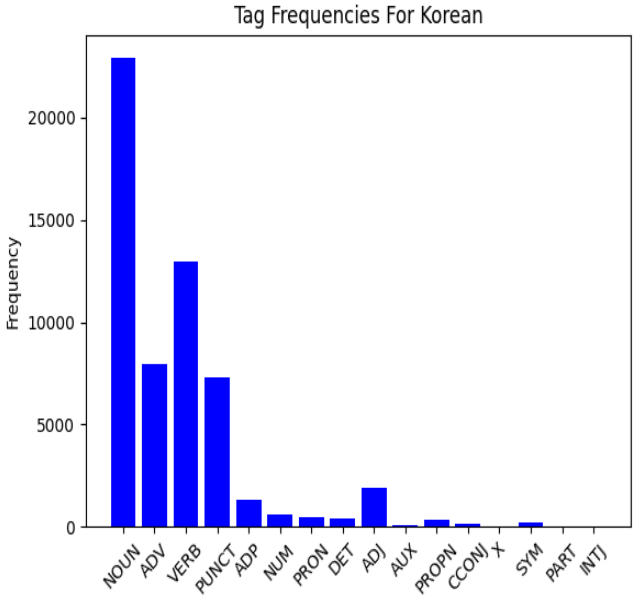

Figure 2: Tag Frequencies for Korean Corpus : can see this distribution is more skewed compared to the other languages, likely due to its highly inflectional nature

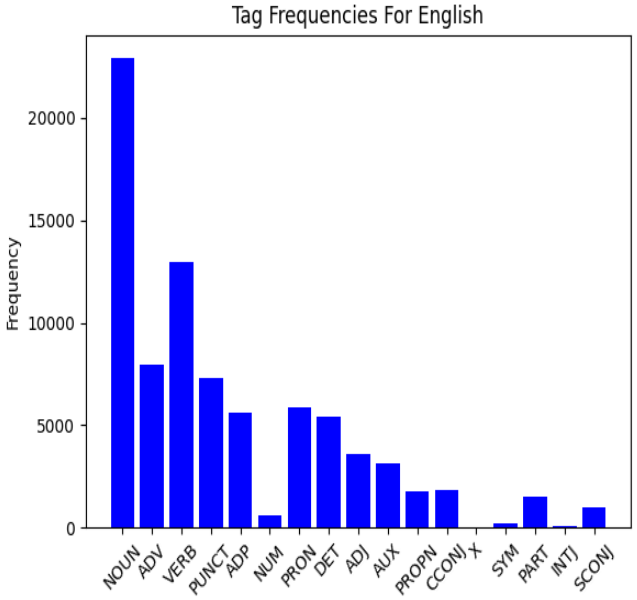

Figure 3: Tag Frequencies for English Corpus

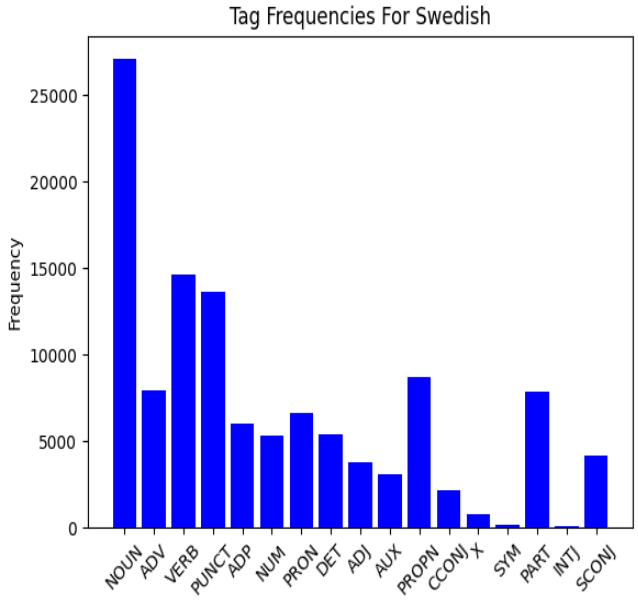

Figure 4: Tag Frequencies for Swedish Corpus

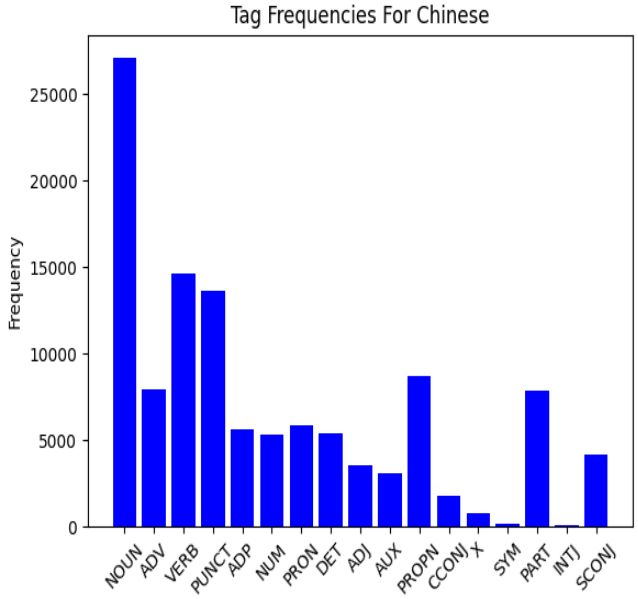

Figure 5: Tag Frequencies for Chinese Corpus

6.2 Confusion Matrix

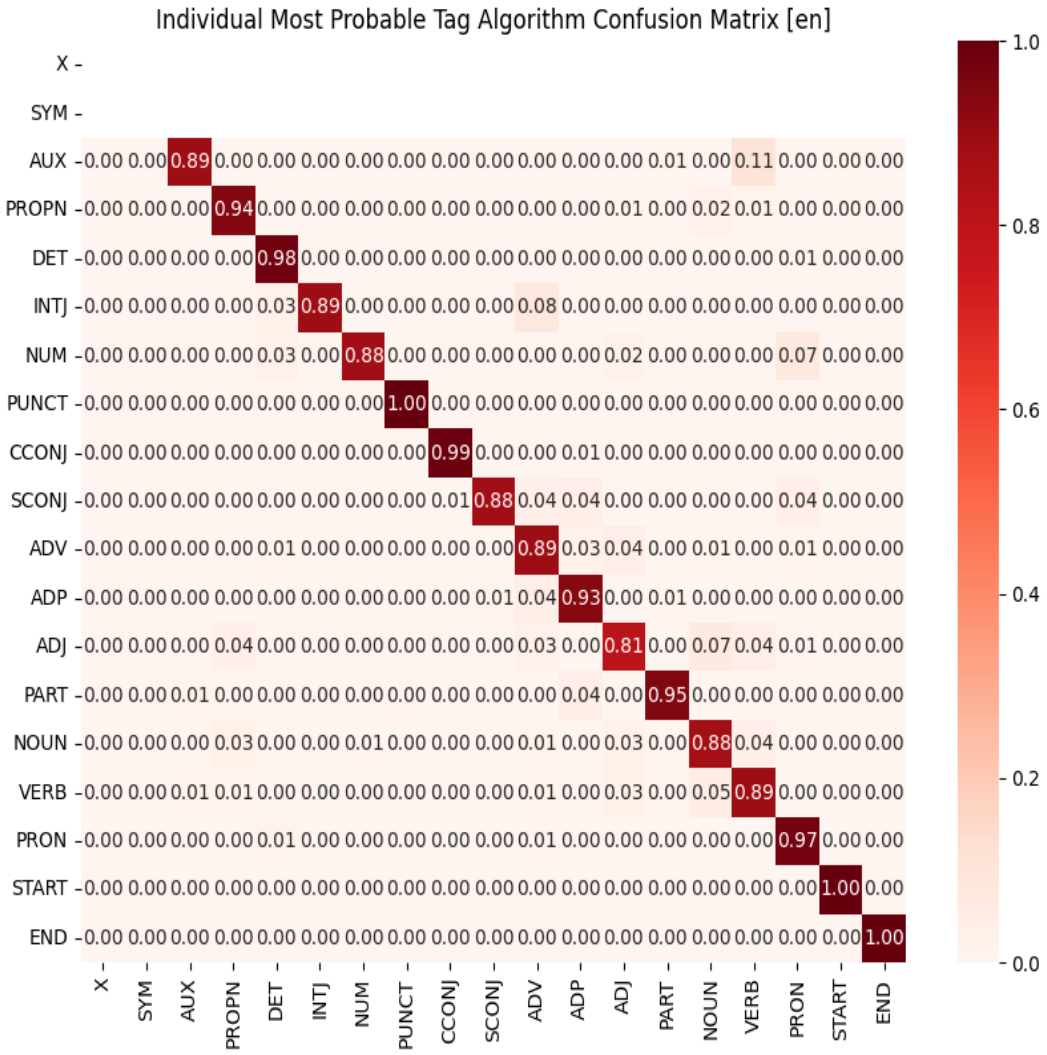

Figure 6: IMPT Confusion Matrix for English

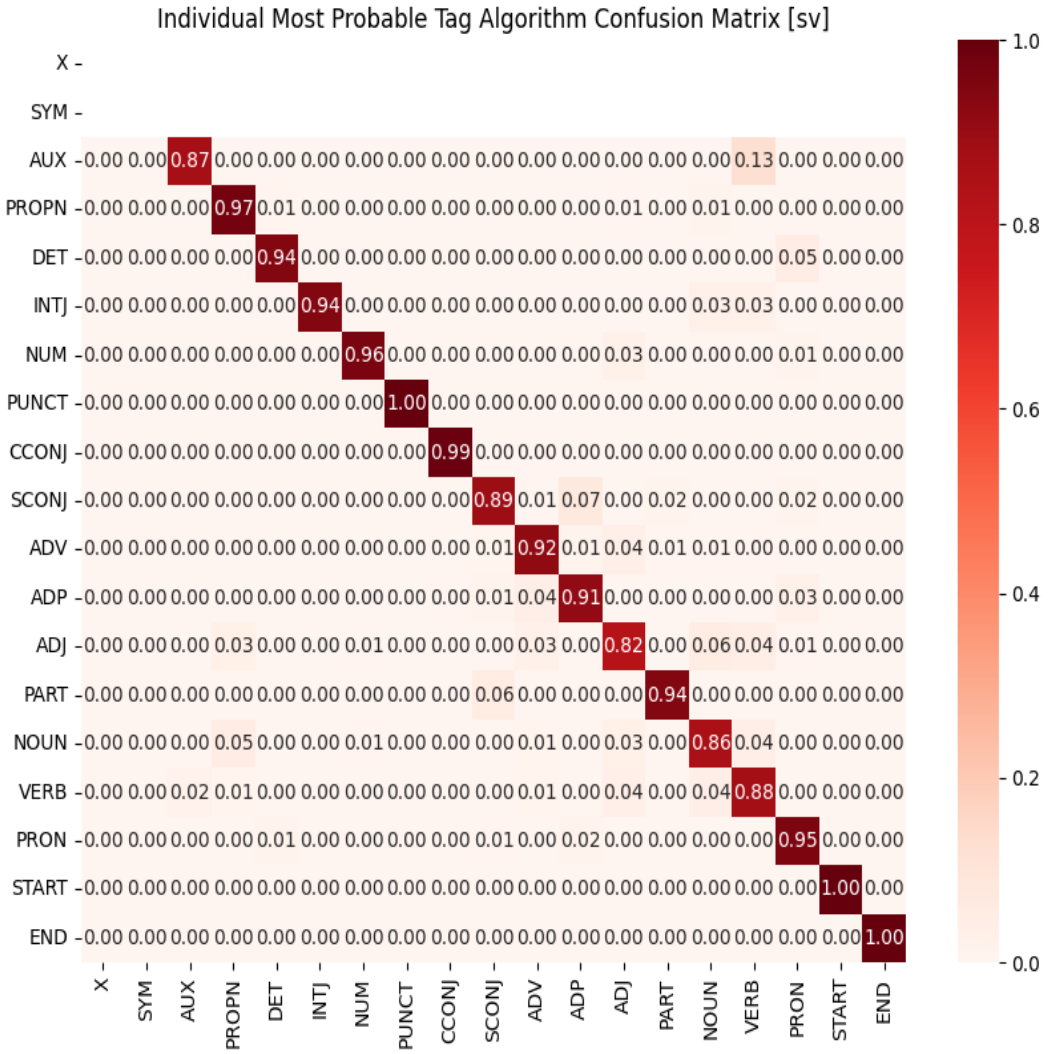

Figure 7: IMPT Confusion Matrix for Chinese

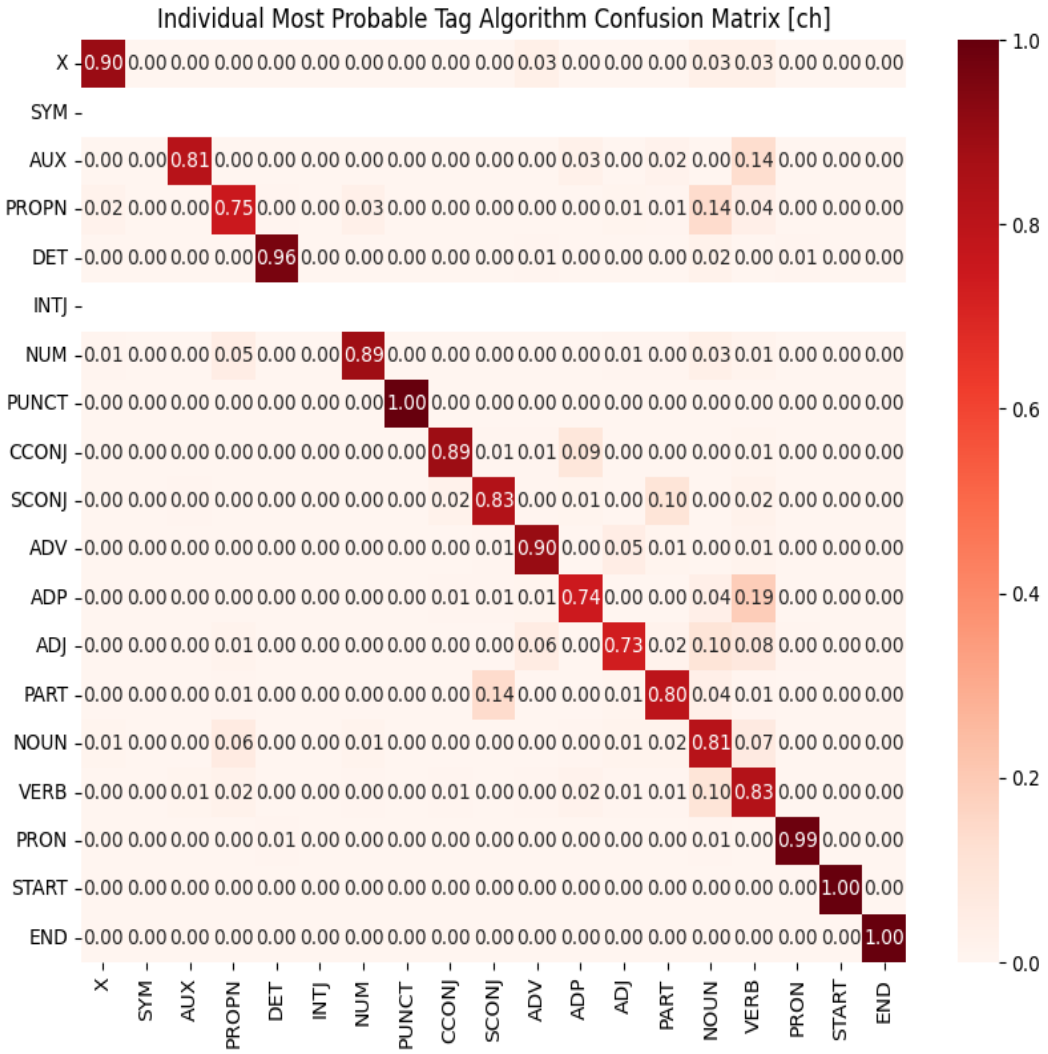

Figure 8: IMPT Confusion Matrix for Swedish

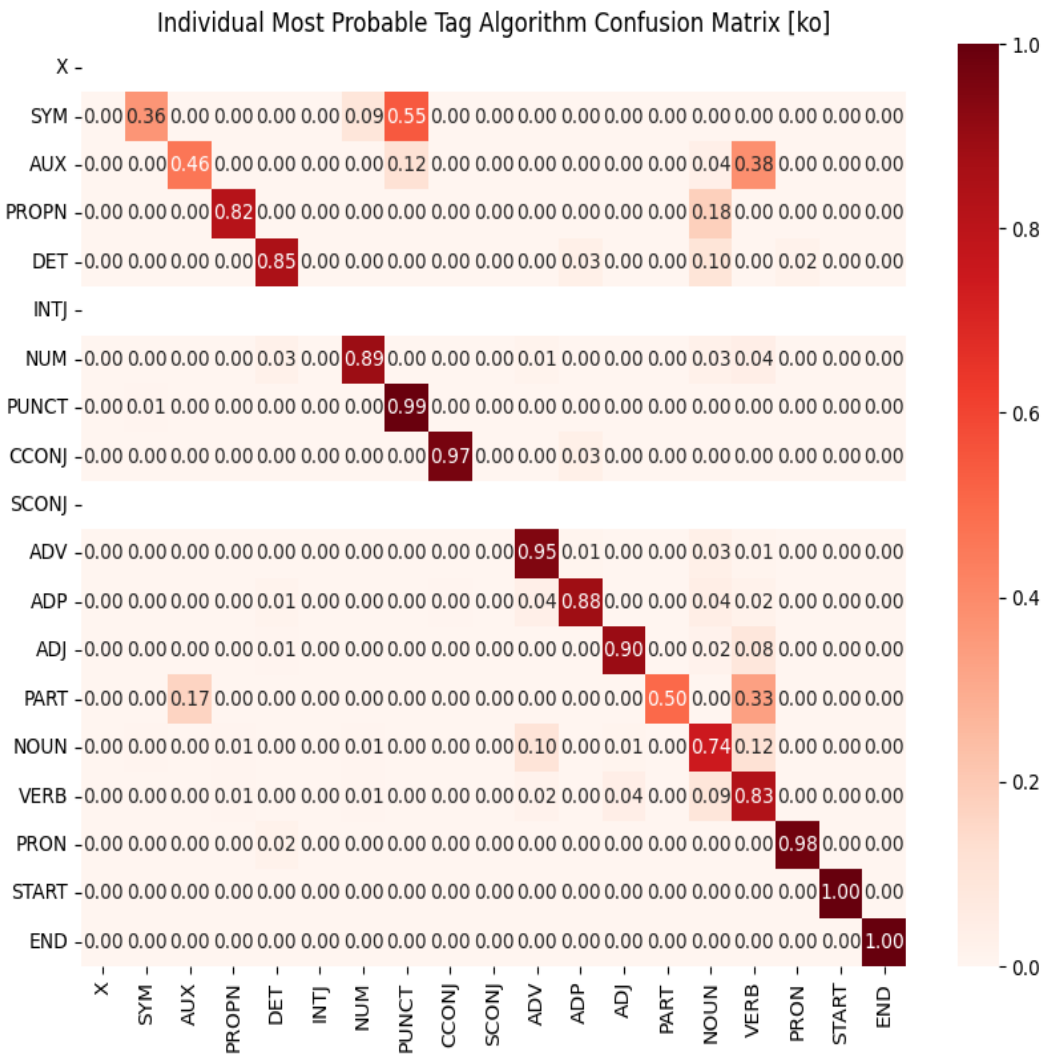

Figure 9: IMPT Confusion Matrix for Korean