Appendix - B

Working of Hindi POS Tagger

A POS tagger based on HMM assigns the best tag to a word by calculating the forward and backward probabilities of tags along with the sequence provided as an input. The following equation explains this phenomenon.

\[ P(t_i|w_i) = P(t_i|t_{i-1}).P(t_{i+1}|t_i).P(w_i|t_i) \]  \hspace{1cm} (B.1)

Here, \( P(t_i|t_{i-1}) \) is the probability of a current tag given the previous tag and \( P(t_{i+1}|t_i) \) is the probability of the future tag given the current tag. This captures the transition between the tags. These probabilities are computed using equation B.2.

\[ P(t_i|t_{i-1}) = \frac{\text{freq} \ (t_{i-1}, t_i)}{\text{freq} \ (t_{i-1})} \]  \hspace{1cm} (B.2)

Each tag transition probability is computed by calculating the frequency count of two tags seen together in the corpus divided by the frequency count of the previous tag seen independently in the corpus. This is done because we know that it is more likely for some tags to precede the other tags. For example, an adjective (JJ) will be followed by a common noun (NN) and not by a postposition (PSP) or a pronoun (PRP). Figure B.1 shows this example.

| अच्छा लड़का  | (*) अच्छा के  | (*) अच्छा तुम  |
| JJ | NN | JJ | PSP | JJ | PRP |

**Figure B.1:** Tag transition probabilities

By looking at this figure, we know that probability of \( P(JJ|NN) \) will fetch a high score then \( P(JJ|PSP) \) and \( P(JJ|PRP) \). Since the last two are wrong, we might not get even a single count for them.
We also computed the word likelihood probabilities using $P(w_i|t_i)$ i.e. the probability of the word given a current tag. This probability is computed using equation B.3.

$$P(w_i|t_i) = \frac{freq(t_i, w_i)}{freq(t_i)}$$  \hspace{1cm} (B.3)

Here, the probability of word provided a tag is computed by calculating the frequency count of the tag in question and the word occurring together in the corpus divided by the frequency count of the occurrence of the tag alone in the corpus.

We used two special tags <S> to denote the starting of the sentence and </S> denoting the ending of the sentence which was added to all the sentences of the training corpus. Using the above two equations we created a tag-tag database which computed all the tag transition probabilities of tag combinations available in the corpus and a word-tag database which computed all word likelihood probabilities available in the corpus.

Suppose if we have a word which is an open class word i.e. a noun or verb or adjective or adverb. Then it is a possibility that it might be assigned to multiple tags and we may face the ambiguity issue. For example, as shown in table B.1, we have an ambiguous word which is assigned to a noun and a verb. A human expert can very easily distinguish the two contexts and thus assign a different POS tag to the words. Using HMM this phenomenon can be captured intuitively as we are considering the context of tags (before and after) with respect to the current tag. This context description is a powerful feature of HMM which can decides the tag for a word by looking at the tag of the previous word and the tag of the future word. Figure B.2 shows this phenomenon which is a generative model where there is a hidden
underlying generator of observable events (tag-tag probabilities) and this hidden generator can be modeled as a set of states. Our goal is to find the underlying state sequence from the observed events.

Let us consider this computation using the example “उसका दिल सोने का है”. Here, “सोने” is an ambiguous word which can either have an NN or a VM tag. Figure B.3 and B.4 show the context dependency of the possible tags.

Figure B.2: Context Dependency of Hidden Markov Model

Figure B.3: Context Dependency of “उसका दिल सोने का है” with VM assigned to “सोने”
Since all the other word-tag combinations are same for the ambiguous words, their computation will also be the same. It is the ambiguous word-tag context which will make the difference to the final score. The one which is higher will get selected. So, the computation of \( P(VM|NN) \times P(PSP|VM) \times P(सोने|VM) \) is computed to be 0.03945 and the computation of \( P(NN|NN) \times P(PSP|NN) \times P(सोने|NN) \) is computed to be 0.0092. As the computation of VM is higher than that of NN, VM gets selected for “सोने”. Once all the tags are identified for the input words. They are displayed to the user.