COMP550 Natural Language Processing
Assignment 2

Jonathan Guymont

October 16, 2018

Question 1

(a) **MAP estimate.** The set of possible tag is {N, C, V, J} and the lexicon is {that, is, not, it, good}. The add-1 smoothed initial probabilities are given by

$$\pi_i = \frac{count(Q_1 = i) + 1}{|sentence| + |corpus|}$$

where $|lexicon| = 5$, $|corpus| = 4$ because we only consider beginning of sentence, and $count(Q_1 = i)$ is the number of times a sentence start by the tag i.

$$\pi_N = \frac{0+1}{4+4} = 1/8, \quad \pi_C = \frac{2+1}{4+4} = 3/8, \quad \pi_V = \frac{2+1}{4+4} = 3/8, \quad \pi_J = \frac{0+1}{4+4} = 1/8$$

The add-1 smoothed transition probabilities are given by

$$a_{ij} = \frac{count(Q_{t+1} = j, Q_t = i) + 1}{count(Q_t = i) + 1 * |tags|}$$

where $|tags| = 4$ is the number of different tags.

	N	C	V	J	$count(Q_t = i)$
N	2	0	3	1	6
C	2	0	0	0	2
V	4	0	1	0	5
J	0	0	0	0	0

Table 1: Transition count

$$a_{NN} = \frac{2+1}{6+4} = 3/10, \quad a_{NC} = \frac{0+1}{6+4} = 1/10, \quad a_{NV} = \frac{3+1}{6+4} = 4/10, \quad a_{NJ} = \frac{1+1}{6+4} = 2/10,$$

$$a_{CN} = \frac{2+1}{2+4} = 3/6, \quad a_{CC} = \frac{0+1}{2+4} = 1/6, \quad a_{CV} = \frac{0+1}{2+4} = 1/6, \quad a_{CJ} = \frac{0+1}{2+4} = 1/6,$$

$$a_{VN} = \frac{4+1}{5+4} = 5/9, \quad a_{VC} = \frac{0+1}{5+4} = 1/9, \quad a_{VV} = \frac{1+1}{5+4} = 2/9, \quad a_{VJ} = \frac{0+1}{5+4} = 1/9,$$

$$a_{VN} = \frac{4+1}{5+4} = 5/9, \quad a_{VC} = \frac{0+1}{5+4} = 1/9, \quad a_{VV} = \frac{1+1}{5+4} = 2/9, \quad a_{VJ} = \frac{0+1}{5+4} = 1/9,$$

$$a_{JN} = \frac{0+1}{0+4} = 1/4, \quad a_{JC} = \frac{0+1}{0+4} = 1/4, \quad a_{JV} = \frac{0+1}{0+4} = 1/4, \quad a_{JJ} = \frac{0+1}{0+4} = 1/4,$$

$$A = \begin{pmatrix} 0.3 & 0.1 & 0.4 & 0.2 \\ 0.5 & 1/6 & 1/6 & 1/6 \\ 5/9 & 1/9 & 2/9 & 1/9 \\ 0.25 & 0.25 & 0.25 & 0.25 \end{pmatrix}$$

	that	is	not	it	good	$count(Q_t = i)$
N	4	0	2	2	0	8
C	2	0	0	0	0	2
V	0	6	0	0	0	6
J	0	0	0	0	1	1

Table 2: Emissions count ($count(O_t = k, Q_t = i)$)

The MLE of the emissions probability is given by $b_{i,k} = count(O_t = k, Q_t = i)/count(Q_t = i)$. The $count(\cdot, \cdot)$ are shown in table 2. To smooth the emissions probability with add-1, we add 1 to the numerator of the MLE and we add 5 to the numerator. The matrix of emissions probability is given by

$$B = \begin{pmatrix} 5/13 & 1/13 & 3/13 & 3/13 & 1/13 \\ 3/7 & 1/7 & 1/7 & 1/7 & 1/7 \\ 1/11 & 7/11 & 1/11 & 1/11 & 1/11 \\ 1/6 & 1/6 & 1/6 & 1/6 & 2/6 \end{pmatrix}$$

(b) **Viterbi.** The probability in the first column are given by $P(O_1, Q_1) = P(Q_1)P(O_1|Q_1) = \pi_{Q_1} = \pi_{Q_1} b_{O_1,Q_1}$. For example, the first entry is given by

$$\delta_N(1) = \pi_N b_N(that) = 1/8 \cdot 3/10 = 3/80$$

The value in the second column are given by $\max_i P(Q_{t-1} = i, O_{t-1})P(Q_t = j|Q_{t-1} = i)P(O_t|Q_t = j) = \max_i \delta_i(t-1)a_{ij}b_j(O_t)$. For example, the first entry of the second column is given by

$$\begin{aligned} \delta_N(2) =& \max_i \delta_i(1) a_{iN} b_N(is) \\ =& \frac{1}{13} \max\{\frac{5}{104} \cdot \frac{3}{10}, \frac{9}{56} \cdot \frac{1}{2}, \frac{3}{88} \cdot \frac{5}{9}, \frac{1}{48} \cdot \frac{1}{4}\} \\ =& \frac{5}{809} \end{aligned} \tag{1}$$

	that	is	good
N	5/104	9/1456	5/6864
C	9/56	3/784	1/3696
V	3/88	3/176	1/2904
J	1/48	1/224	1/1584

Table 3: Trellis of the Viterbi algorithm

Thus the most likely tags are:

$$(that, C), (is, V), (good, N)$$

(c) **EM.** First, we reestimate B with add-1 smoothing so the word *bad* do not have zero probability to occur.

$$B = \begin{pmatrix} 5/14 & 1/14 & 3/14 & 3/14 & 1/14 & 1/14 \\ 3/8 & 1/8 & 1/8 & 1/8 & 1/8 & 1/8 \\ 1/12 & 7/12 & 1/12 & 1/12 & 1/12 & 1/12 \\ 1/7 & 1/7 & 1/7 & 1/7 & 2/7 & 1/7 \end{pmatrix}$$

The parameters Π and A do not change, since the new examples are completely untaged. The next step is to use viterbi to make prediction on the tags of the new sentence. Then we should use those taged examples to retrain the model.

The prediction for the examples (following the same methods as in part b)) are respectively $[(bad, C), (is, V), (not, N), (good, J)$ and $[(is, V), (it, N), (bad, V)]$. Now we can add these two tagged examples to the training set and recompute the parameters Π, A, and B just like in part (a).

Question 3

1. Brief summary

Theoritical framework. The authors develop a novel PCFG approach for parsing part of speech in order to predict POS tags. They focused on improving the features/targets used to train the POS tagger without discussing about the machine learning algorithm. They used an unlexicalized methods (as opposed to lexicalized) and report an accuracy closed to the state of the art, while requiring less memory and being simpler in general. Improving the structure of the targets lead to significant gain in the model prediction, showing that in a task such as POS tagging, the structure of the prediction as a huge role on the performance of the model.

Experiments. The part of speech tagger was trained on sections 2-21 of the WSJ section of the Penn treebank. The first 20 files of the section 22 were used as validation set. The section 3 was used as the test set. The authors do not mentioned anything about model specific hyperparameters tuning during the training process, but they tuned features and output structure. For example, the children of S should be NP and VP or NPŜ and VPŜ; features that gives the best accuracy on the validation set are kept.

Definition 1. *In* ***lexicalized PCFG parsing****, the main word of a part of speech (thehead) is used has an additional feature to predict the tags in a part of speech (POS). The paper does not give examples of methods to infer the head of a POS. The paper does not explain how one could use this additional feature (the head) to make better tag prediction.*

Definition 2. *Unlexicalized PCFG parsing No specification at the lexical level is make. With this approach, features are refined by improving the parsing three. For instance, basic tag like NP can be subcategorized with NP-Pl and NP-Sg. Also, their is often more then one way to split a POS in sub-POS.*

In this case,the grammar is not systematically specified to the level of lexical items. A standard technique in unlexicalized parsing we compare against is vertical markovization, i.e. to refine nonterminals by annotating them with their parent (or grandparent). subcategorization of POS categories modifying make up of existing tags :
NP [birds] is not allowed
NP-Pl is fine

2. Evaluation and synthesis of what I learned in the paper

step 1. They trained their models on a part of WSJ section of the Penn treebank. They used another section (23) for the test set. To parse the grammar, they used an array-based Java implimentation of CKY parser.
step2. Then they introduced series of sparsing techniques which helps them to split the sentences in smaller parts: To use the unlexicalized parsing, the authors used vertical and horizontal Markovization. The vertical history includes the node itself and parent and grandparent and so on, and in a similar way only the previous horizontal ancestor matter.Then they use the two major annotation strategy which are the External and internal annotation. And the parent annotation is the most important feature of external annotation and influences on internal annotations. Then they used the Unary-DT and Unary-Adv that shows if the adverb and determiner are depended on their parents or not. The they used Tag splitting make the POS fined grained. for example parent annotation tag (TAG-PA), which marks all the pre-terminals with their parent

category. Also they add -TMP tag on Np's to show that the NP with the S parent can be a temporal NP$\cdots$. They also head annotation, since the head word indicates how the constituent will behave for example if the left most child is a possessive-NP or not. Also using SPLIT-VP the annotated the verb part (VP) by their head tags and bring all finite forms to a single tag VBF, that significantly improved their result for 2.66% improvement.
Advantages: The model is giving a result as good as lexicalized while it is easy to understand and easy to parse. They introduced methods of parsing that have a big influence on their result and can be used in lexicalized parsers too.

limitations They did their training and testing on a corpus probably with a specific subject. So it might not be very efficient for new corpus with different subjects. They have not provided detail for all additional tags and did not give some key parts of their programming part.(like pseudocode)
relevance to the class This result shows that formal understanding of a language grammar and sub-categorization of categories, let us develop appropriate algorithms for dealing with syntax, as we discuss it in class.

3. Three questions related to the paper

(1) Does it give us a better result, If we use the same parsing rules for lexicalazied PCFGs?

(2) Is it possible that we reduce the number of rules from the model and get a better or at least same result?

(3) Is it possible to make the similar model for other languages like french?