Solver: CAO LIU

Email Address: CAOL0002@e.ntu.edu.sg

1.

(a) Stop words: {“A”, “is”, “an”, “of”}
S1: sequence ordered collection entities
S2: set unordered collection unique entities
S3: bag unordered collection entities

(b) After removing stop words and applying stemming algorithm
S1: sequence order collect entity
S2: set order collect unique entity
S3: bag order collect entity

(c) N = 13, V = 7
Table of unigram counting

sequence	order	collect	entity	set	unique	bag
1	3	3	3	1	1	1

Table of bigram using Laplace smoothing:

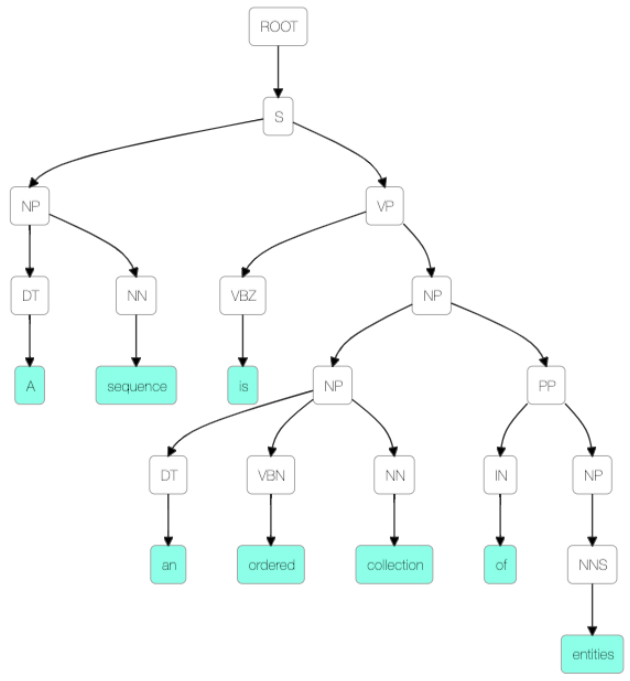
	sequence	order	collect	entity	set	unique	bag
sequence	1/8	2/8	1/8	1/8	1/8	1/8	1/8
order	1/10	1/10	4/10	1/10	1/10	1/10	1/10
collect	1/10	1/10	1/10	3/10	1/10	2/10	1/10
entity	1/10	1/10	1/10	1/10	1/10	1/10	1/10
set	1/8	2/8	1/8	1/8	1/8	1/8	1/8
unique	1/8	1/8	1/8	2/8	1/8	1/8	1/8
bag	1/8	2/8	1/8	1/8	1/8	1/8	1/8

(d) Assumption: applying stop words removal and stemming. P(order|<UNK>)=1/8

$$\mathrm{P(A\ list\ of\ ordered\ entities)} = \mathrm{P(list\ order\ entity)}$$

$$= \mathrm{P(< UNK >\ order\ entityP(order| < UNK >)x\ P(entity|order)} = \frac{1}{8} * \frac{1}{10} = \frac{1}{80}$$

(e) (b)



(c) Many ambiguous words appear frequently in corpus. Small number of words are high commonly used

2.

(a)

(i). Regular expression: a compact sequence of textual strings for specifying patterns in programs. e.g. "/[t/T]he/".

(ii). Finite state automata: An abstract machine that can be in exactly one of a finite number of states at any given time. The FSM can change from one state to another in response of some external inputs. FSM consists of the set of states:

- The set of states: Q
- A finite alphabet: Σ
- A starting state
- A set of accept/final states
- A transition function $Q \times \Sigma \rightarrow Q$

(iii). Non-deterministic finite state automata: A NFA is a kind of FSA, which allows epsilon transitions (empty inputs) and possibly multiple different next states under the same input. Transition function of NFA is δ: $Q \times \Sigma \rightarrow P(Q)$, where δ may return a set of states.

(iv). Perplexity: Perplexity is the probability of the test set (assigned by the language model), normalized by the number of words:

$$Perplexity(W) = P(w_1 w_2 \dots w_N)^{\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1 w_2 \dots w_N)}}$$

(b) Word segmentation: identifying the tokens (words, symbols) in a text that may want to deal with. Maximum matching word segmentation: start a pointer at the beginning of the string, then find the longest word in dictionary that matches the starting at pointer, finally move the pointer over the word in string. When applying maximum matching word segmentations, an FSA which capture the word morphology can be applied to find the longest word matching in the dictionary.

(c) N-gram model can be used to predict the next word based on the previous N-1 word.
We can use a 3-gram model. The model can be derived from training corpus. During spelling error correction, 3-gram model will check the probability of the coming word based on the previous 2 words. For example, in the sentence "Please help mi". Model will check P("mi" | "Please help") and report spelling errors based on the probability.

(d) HMM is a weighted finite-state automate where each are is associated with transition probability and state is associated with observation likelihoods. The transition probability indicates the probability of transiting from one state to another, and observation likelihoods indicates the probability of observation at a given state. HMM can be used to solve the problem of POS tagging. The POS tagging problem is to determine the POS tag for particular instance of a word. The problem can be described as

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n|w_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} P(w_i|t_i)P(t_i|t_{i-1})$$

omit Viterbi Algorithm.

Another problem is: Given an observation sequence, what is the probability of that sequence given a model$P(seq|model)$. The problem can be solved with Forward Algorithm.

3.

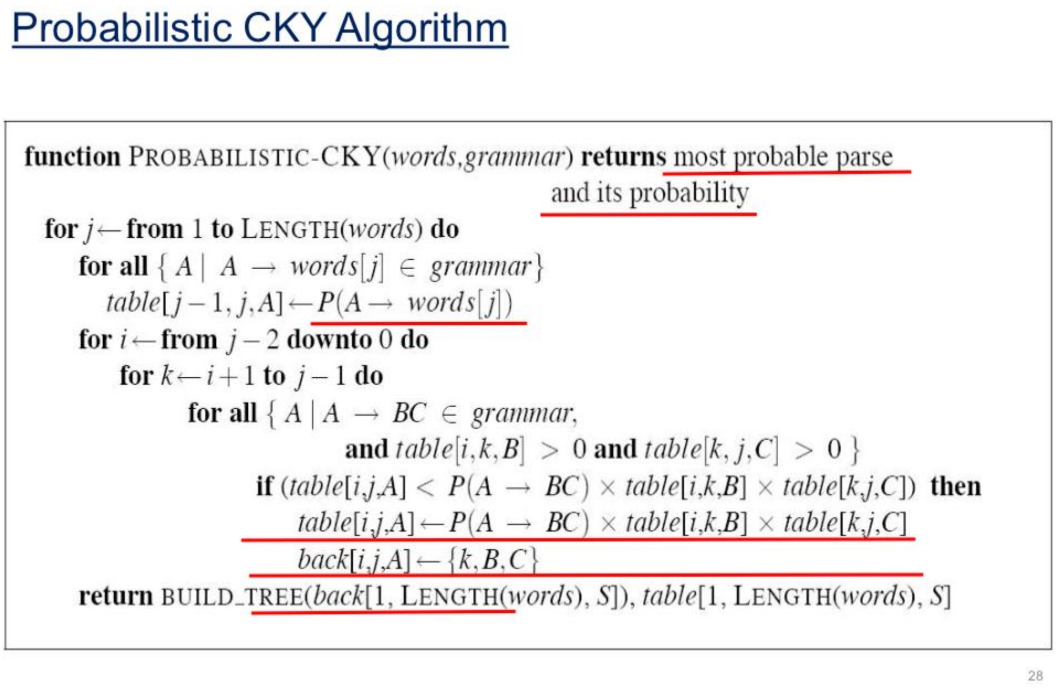
(a) CNF: an RHS must have exactly two non-terminals (X -> Y Z) or have one terminal item (X -> w). Rules which are not in CNF: S -> VP, NP -> Pronoun, VP -> Verb,

(b)

$\gamma \rightarrow \bullet$ S	[0, 0]	Dummy start state
S $\rightarrow \bullet$ NP VP	[0, 0]	Predictor
S $\rightarrow \bullet$ VP	[0, 0]	Predictor
NP $\rightarrow \bullet$ Det Nominal	[0, 0]	Predictor
NP $\rightarrow \bullet$ Pronoun	[0, 0]	Predictor
VP $\rightarrow \bullet$ Verb	[0, 0]	Predictor
VP $\rightarrow \bullet$ Verb NP	[0, 0]	Predictor
Det $\rightarrow \bullet$ this	[0, 0]	Predictor
Pronoun $\rightarrow \bullet$ I	[0, 0]	Predictor
Verb $\rightarrow \bullet$ like	[0, 0]	Predictor
Verb $\rightarrow \bullet$ book	[0, 0]	Predictor

Pronoun → I •	[0, 1]	Scanner
NP → Pronoun •	[0, 1]	Completer
S → NP • VP	[0, 1]	Completer
VP → • Verb	[1, 1]	Predictor
VP → • Verb NP	[1, 1]	Predictor
Verb → • like	[1, 1]	Predictor
Verb → • book	[1, 1]	Predictor
Verb → like •	[1, 2]	Scanner
VP → Verb •	[1, 2]	Completer
VP → Verb • NP	[1, 2]	Completer
NP → • Det Nominal	[2, 2]	Predictor
NP → • Pronoun	[2, 2]	Predictor
Det → • this	[2, 2]	Predictor
Pronoun → • I	[2, 2]	Predictor

(b)



Probabilistic CKY Algorithm

```
function PROBABILISTIC-CKY(words,grammar) returns most probable parse
                                                  and its probability
  for $j \leftarrow$ from 1 to LENGTH(words) do
     for all $\{ A \mid A \rightarrow words[j] \in grammar\}$
        $table[j-1, j, A] \leftarrow P(A \rightarrow words[j])$
     for $i \leftarrow$ from $j-2$ downto 0 do
        for $k \leftarrow i+1$ to $j-1$ do
            for all $\{ A \mid A \rightarrow BC \in grammar,$
                      and $table[i,k,B] > 0$ and $table[k,j,C] > 0 \}$
                 if $(table[i,j,A] < P(A \rightarrow BC) \times table[i,k,B] \times table[k,j,C])$ then
                    $table[i,j,A] \leftarrow P(A \rightarrow BC) \times table[i,k,B] \times table[k,j,C]$
                    $back[i,j,A] \leftarrow \{k,B,C\}$
  return BUILD_TREE(back[1, LENGTH(words), S]), table[1, LENGTH(words), S]
```

28

(c)

PCFG = (T, N, S, R, P), where P(R) gives the probability of each rule.

$$\forall X \in N, \sum_{X \rightarrow y \in R} (X \rightarrow y) = 1$$

Using a treebank, the probability of each rule to apply is $P(X \rightarrow \beta | X)$

$$P(X \to \beta | X) = \frac{count(X \to \beta)}{\sum_{\gamma} count(X \to \gamma)} = \frac{count(X \to \beta)}{count(X)}$$

The probability of each rule $P(X \to \beta | X)$ in PCFG is obtained by the total number of the non-terminal X appears divided by the number of times the rule $X \to \beta$ applied in a treebank.

4.

(a) For the rules that involve NP ---- S → **NP** VP and VP → Verb **NP**, In the first rule, the NP has a higher chance to be interpreted as pronoun, while in the second rule, the NP has a higher chance to be interpreted as non-pronoun. After encoding contextual dependencies into PCFG symbols. The rules become S → **NP^S** VP and VP → Verb **NP^VP**. Different probabilities can be assigned to the rules regarding to NP^S and NP^VP.

(b) Morphological Difference, Lexical Difference, Syntactic Difference, Semantic Difference

(c) **Lexical Affinity**: assigning arbitrary words probable “affinity” to particular emotions (e.g. “accident” has 75% probability of indicating a negative affect). This method is easy to use but may be tricked by negation or context-dependent words.
Statistical method: feeding a ML algorithm with a large training corpus. Statistical methods are able to learn the valence of keywords as well as other arbitrary words in the training corpus. Statistical methods are powerful but need for a large training set and depend on the domain of the training corpus.

(d)

(i). Challenges: The reviews rely on **common knowledge and common sense**: obvious things people know and usually unstated. For example, galaxyS3 is a phone. **Informal text, acronyms, emoticons** and even **misspelling** exist in the review text. All these need to be translated into plain English. **Anaphora** need to be resolved. People may use **sarcasm** in the review. Sarcasm transforms the polarity of positive utterance into its opposite.

(ii). Techniques: Micro-text Analysis. Informal text, acronyms and emoticons can be normalized and translated to plain English. Micro-text analysis can be used during pre-processing the corpus.
Topic Spotting consists in assigning category tag to a piece of text, it includes subtasks such as NER and WSD, which help categorize the reviews

Sarcasm Detection help detect sarcasm used in the text and help correctly assign polarity to the text.

All the best for your exams!