Natural Language Processing

A Machine Learning Approach

INSTRUCTOR MANUAL

YUE ZHANG
Westlake University

Acknowledgements

I am grateful for Pai Liu and Boyuan Zheng for their assistance in typesetting the document and working on the initial solutions for some of the final reference answers.

Introduction

Overall, this book has been designed as a textbook to support senior undergraduate and graduate courses on natural language processing. The focus is on the algorithmic side rather than the linguistic side of the field. Depending on the curriculum and educational objectives, content selection can be made for a balance of coverage and depth. For example, if a linguistics course has not been taught as a prerequisite, then more time can be spent on introducing linguistic knowledge such as what is syntax and what are part-of-speech tags. If general machine learning courses are taught as pre-requisite classes, then relevant content can be taught at faster paces and deeper machine learning techniques for natural language processing can be taught.

Below we first discuss some general suggestions for using the book, before introducing the content organization and correspondingly term plans for different typical courses.

Teaching Suggestions

Unless machine learning courses have been takes in-depth as prerequisites, it can be infeasible to cover all content of the book in a single semester. Depending on the target student and teaching goals, a selection of relevant content is necessary. In addition, the level and details and amount of time dedicated to each concept are also worth consideration. To our experience, it is of great importance to choose a proper level and details in order to make the learning curve gentle. We aim to provide teaching slides with as much detail as possible on the book website, so that content selection can be made easy. The slides are updated periodically.

The book offers introduction, equations, algorithms and proofs, which are the main content classroom teaching. For student learning, written exercises and programming are also necessary. We provide example answers in this instructor manual, which will be updated periodically with corrections and additions. Example code is also provided at GitHub, a link for which can be found at the book website. Outstanding student code will be uploaded to the website so that the students have multiple references. With the above materials, a standard semester plan can include **lecture sessions**, small group **tutor sessions** and **lab sessions**. **Evaluation** can be made with both written exercises and coding projects.

For each major algorithm, there are three rough levels of understanding depth. At the shallowest level, the student gains awareness of the algorithm, knowing the basic principles and the situations where it is used. At a deeper level, the student should know the algorithm at the pseudo code level, being able to tell important characteristics such as the asymptotic complexity, use software libraries and make simple modifications. At the deepest level, the student is trained to understand the algorithm in full detail, so that they can code the algorithm from scratch and make innovative adaptations to different task scenarios. For example, in sequence labeling tasks, Viterbi algorithm and forward/backward algorithms for conditional random fields (CRF) had wide influence in NLP. Therefore, research students are recommended to understand them in the coding level. However, a course that focus on neural NLP may choose to introduce the above algorithms only at the shallowest level. Exercises and coding projects should be given according to the outcome expectations.

Content

The book is organised in three main parts. Part I introduces the basic concepts in NLP and its relevant machine learning principles. It is recommended to teach this part in detail, because it lays a foundation for introducing the second and third parts. For students who have learned machine learning in general, this part can be taught at a faster pace. However, it is recommended not to skip it because the concepts are discussed from the NLP perspective and connects strongly with subsequent contents. Even student who are familiar with relevant machine learning algorithms can benefit from learning the content.

Part II discusses structured prediction, which consists of tasks that are rather unique for NLP (and a few relevant fields such as bioinformatics). This part focus on statistical methods, which are a natural extension of the methods discussed in Part I for classification, thereby demonstrat the challenges brought up by complex output structures. As examples of structured prediction, sequence labeling, sequence segmentation and tree prediction are discussed. In addition, transition-based structured prediction is discussed as a framework for discriminative modeling, and Bayesian learning is introduced as a general extension to the generative models.

Part III focuses on neural models for NLP. It extends the linear models in Parts I and II into deep neural networks, drawing the connections and differences. While discussing neural frameworks for text classification, sequence labeling, sequence segmentation and tree prediction, this part also introduces tasks that are more easily solved using neural network models only, such as sequence-to-sequence tasks. In addition, pre-training is discussed in detail, and neural latent variable models are introduced as general extensions to discrete latent variable models.

Themes

Connections between chapters can be viewed in themes. Below we list typical themes for facilitating the selection of contents.

Generative models. Generative models are first introduced in Chapter 2, where the concept of parameterisation is introduced, and basic probabilistic modeling is discussed. We discuss a set of techniques for parameterising a generative model, which breaks a joint probability into less sparse parameters by using Bayes rule, probability chain rule and making independence assumptions. The same techniques are used to parameterise n-gram language models, Naive Bayes text classifiers (Chapter 2), Hiddden Markov Models (Chapter 7) and Probabilistic Context Free Grammar models (Chapter 10). We draw plate notations to describe such generative models, which are then unified into Bayesian networks in Chapter 12.

Discriminative models. We introduce feature vectors as a means to turn linguistic tasks into mathematical problems in Chapter 3, where classification an clustering are discussed as two related problems in the vector space. Then we introduce perceptrons, SVMs and log-linear models are three typical discriminative text classifiers, unifying them into one generalised perceptron model in Chapter 4, showing that the main difference between them is the loss function (including the regularisation term). In Part II, the same discriminative modeling techniques are applied for structured tasks, resulting in structured SVM, conditional random fields(CRF), semi-CRFs etc, and the underlying principles is repeatedly seen on different structures. In Part III the generalised perceptron is extended into a multi-layer perceptron, which leads to discussion of representation learning.

Neural models and statistical models. Neural models always reflect their counterpart statistical models. Chapters 13 and 14 correspond to Chapters 3 and 4 for discriminative text classification. Chapter 15 corresponds to Chapters 7, 8, 9 and 10 for graph-based structured prediction, and Chapter 11 for transition-based structured prediction.

Latent variables. Chapters 6, 12 and 18 are dedicated to latent variables. In Chapter 6, the problem of hidden variables is presented, and several basic ideas for dealing with hidden variables are given. Expectation-maximisation (EM) algorithm are discussed in detail. Chapter 12 revisits hidden variables under a Bayesian network setting, where sampling techniques are used. Chapter 18 discusses latent variables in neural networks, introducing generalised EM algorithms and variational models.

Optimisation techniques. The only optimiser that we choose to introduce is stochastic gradient descent (SGD), which is used to train SVMs, perceptrons, log-linear models(Chapter 4) and neural networks(Chapter 13 and 14), for both text classification and various structured prediction tasks.

Example Course Structures

Undergraduate course with a focus on engineering

This course is designed for computer science students to learn the introductory level knowledge concerning NLP, being able to build neural network models for solving typical problems. It should be taught in 14 — 18 weeks with 2 hours lecture each week. Consider 1 hour tutoring session (problem solving and question answering) and 1 hour lab session.

Chapters 1 — 5 are taught in the first 5 — 7 weeks. The goal is to teach students the basics of generative and discriminative modeling, the generalised perceptron algorithm and the basics of information theory and loss function design. The students should have hands-on programming of Naive Bayes text classification, TF-IDF document vector calculation, k-means clustering, at least one model out of SVM, perceptron and logistic regression, and basic word vector forms. Sections 5.1.1 to 5.1.3 can be optional.

Chapters 13 — 14 are taught in 4 weeks. Section 14.2 can be optional given limited time. The main goal of teaching is to learn the basics of neural network modeling. The students should have hands-on programming for multi-layer perceptrons, CNN, pooling, LSTM and use various SGD variants such as AdaGrad.

Chapter 15 should be taught in 2 weeks, with only Section 15.1 and 15.2. The goal is to learn the basis of structured prediction. There is no need to discuss local modeling and global modeling because Part II of the book is skipped, and by default all the models taught in this course are local models. More time can be spent introducing sequence labeling tasks and tree prediction tasks such as parsing. Chapter 11 should be taught as background where necessary, when transition-based models are discussed. The students should have hands-on programming on both graph-based and transition-based models.

Section 16.1 should be taught in 1–2 weeks. The goal is to learn the basics of a sequence to sequence problem. The students should have hands-on coding using Transformer for a machine translation or text summarization task.

Chapter 17 should be taught in 1 — 2 weeks. The goal is to learn the basic knowledge of word embeddings. The students should have hands-on programming using word2vec and GloVe, viewing the t-SNE visualizations. In addition, they should have hands-on programming using BERT or other pre-trained language models. Section 17.3 can be made optional.

Undergraduate course with a research focus or junior graduate course

This course prepares junior research students to enter the NLP field. It should be taught in 14 — 18 weeks with 2 hours lecturing each week, with one hour or more tutoring (problem solving and question answering session) and 2 hours practical session.

The first 6 chapters should be taught in 6 — 8 weeks. It gives the students the foundations of generative and discriminative modeling. Relevant content can be taught in faster pace if the students already have machine learning background.

Programming exercises should ideally be done on most concepts.

Chapters 7 and 8 and Section 9.1 should be taught in 2 — 3 weeks. The goal is to teach the challenges of sequence labeling, understanding how the techniques of generative and discriminative modeling can be extended to structured prediction tasks from text classification tasks. The students should have programming exercises on HMM training and Viterbi algorithms. It is recommended that both CRF and structured perceptron are assigned for programming exercises. According to student background and time, forward-backward algorithm and EM algorithms can be skipped.

Chapter 11 should be taught in 1–2 weeks with necessary discussion of Section 9.3. Introduction should be made on segmentation tasks and parsing. Programming may be taken optional largely due to the fact that feature engineering is not particularly relevant in current NLP research.

Chapters 13 — 14 should be taught in 2–3 weeks. The students are expected to build neural network models for classification using CNN, pooling and LSTM. They should have programming exercises for all relevant neural models, playing with hyperparmeters and optimisation algorithms. Section 14.2 can be made optional if time is limited.

Chapter 15 should taught in 1 week with Section 15.3 being optional. The students should learn how to use neural networks for structured prediction, building graph- and transition-based models for sequence labeling and parsing. This chapter should be taught with a reference to Chapters 7, 8 and 11. Programming is recommended for building a transition-based model also.

Chapter 16 should be taught in 1 or 2 weeks. The students should learn sequence-sequence modeling, text matching and multi-hop reasoning models. Programming exercises should be made using Transformer for text-to-text tasks and machine reading comprehension tasks.

Chapter 17 should be taught in 1 or 2 weeks. The students should learn the basics of word embeddings and its intrinsic and extrinsic evaluation methods. They should be assigned programming exercises using BERT and other pre-trained language models. They should know the basics of transfer learning.

If more time is available Chapter 12 or Chapter 18 can be selected for 2 weeks teaching.

More in-depth courses on NLP

If more time can be devoted to the task, the instructor can consider Chapter 10 for parsing, which gives much information about extending generative and discriminative techniques to tree structure prediction. When this is given, Section 15.3 can be taught also. Chapter 9 can be given in more detail with regard to semi-CRF.

Exercises

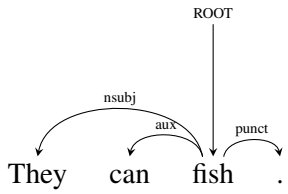
Most of the exercises are comprehensive. It can take much time to answer questions such as 2.5 and 6.8. Correctly answering such questions allows a student to fully understand the algorithms discussed. This should be considered when

making homework assignments.

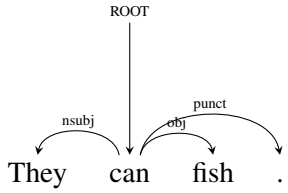
Chapter 1 Reference Answers

1.1 (a) I 'm a student .
(b) He didn 't return Mr. Smith 's book.
(c) We have no useful information on whether users are at risk , said James A. Talcott of Boston 's Dana-Farber Cancer Institute .

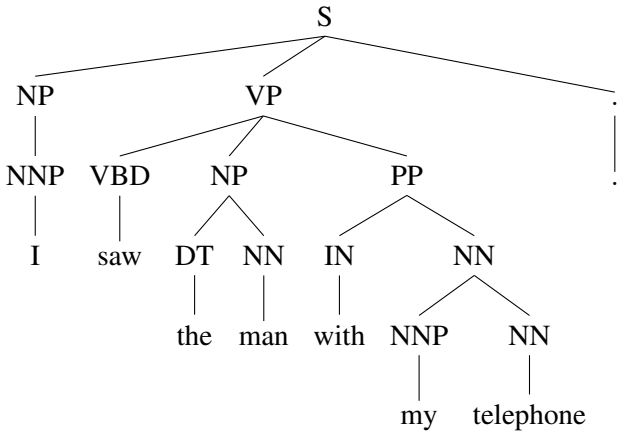
1.2 "They can fish"
(1) PRP MD



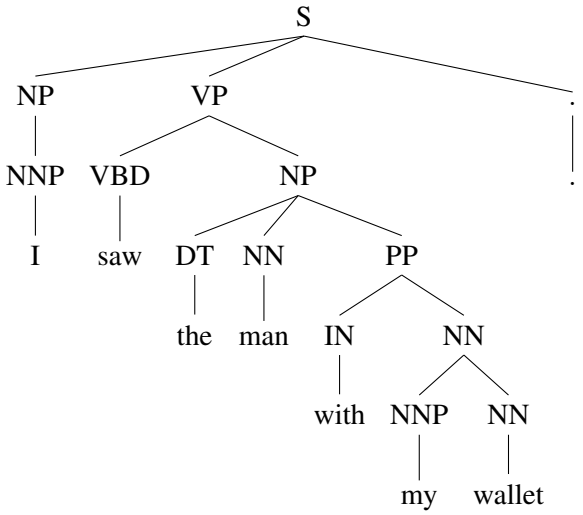
(2) PRP VB NN



1.3 (a)

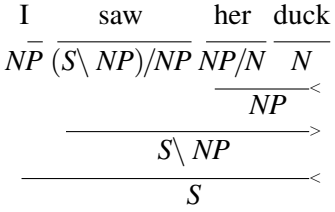


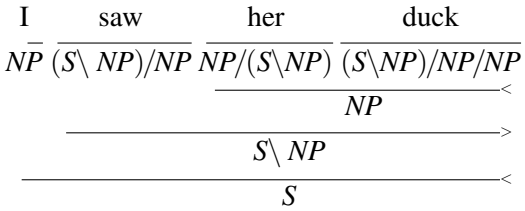
(b)



Difference: In (a) “telephone” is with me while the “wallet” is with the man in (b).

1.4



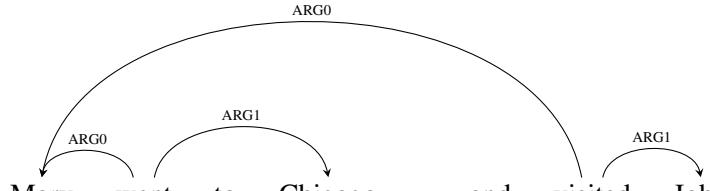


1.5 They both identify spans in the input sentence that correspond to certain nominal entities. While NP chunking extracts all noun phrases, NER extracts only named entities.

1.6 6 lemmas. Noun: a power tool for cutting wood; hand tool having a blade for cutting; a condensed but embody some important fact of experience that is taken as true by many people. Verb: cut with a saw; to move sth backwards and forwards on sth as if using a saw; the past tense of see.

1.7 They can occur in similar contexts. For instance, one can mention that "the lake is very *big*" and also "the lake is very *small*". This provides us a basis to mine antonyms by first identifying words that frequently occur in similar contexts, and then conduct further semantic filtering.

1.8



Similarities: They all have different relationships to different parts of the sentence. Difference:Dependency tree shows the relationship between each word; Predicate-argument relations shows the main relationship between predicate and argument

1.9 buy(Tim,book) & price(book,$1)

1.10 Not all of Jason's classmate like Jason. $\exists x($ classmate $(x, Jason) \& (\neg \text{like}(x, Jason))$

None of Jason's classmate like Jason.$\forall x($ classmate $(x, Jason) \& (\neg \text{like}(x, Jason))$

The scopes of negations are different. In the first sentence, there exist student(s) that are Jason's classmate and like(s) him. In the second sentence, such students do not exist.

1.11 Anaphora resolution considers the fact that a term (anaphor) refers to a preceeding term (antecedant) in text. Coreference resolution considers the fact that multiple terms in a text refer to the same entity. They are related but not the same. On the one hand, coreference resolution considers all entity mentions in a document but anaphora resolution considers individual references. On the other hand, anaphora resolution typically investigates a wider range of references compared to coreference solution, including associative anophora or bridging anaphora, where different terms follow part-whole or other relations beyong identity relation. For instance, in the sentence "*The students went home, the boys running and the girls walking*", both "*the boys*" and "it the girls" refer to "*the students*".

1.12 named entity recognition; discourse segmentation; machine translation

1.13 POS-tagging; semantic role labelling; relation extraction

1.14 Part-of-speech tagging and dependency parsing can be eprformed jointly. Compared with a pipelined approach that performs POS-tagging first and then dependency parsing, the joint model has two salient advantages. First, the two tasks can share mutually beneficial information. Second, error propagation from POS-tagging to dependency parsing can be avoided.

1.15 Extractive summarization can be relatively more faithful, because the content is directly taken from the input. However, it can give unnecessary information because sentences in the document do not always serve as a concise summary. In addition, coherence between different sentences can be low because they can be taken from different parts of a discourse. Abstractive summarization can give a relatively fluent and succinct output thanks to the use of a strong language model. However, the faithfulness to the original document can be a challenge.

1.16 A personal assistant robot can talk to a user in a daily basis, giving the user emotional comfort and companion, while helping with online shopping, travel booking and other daily chores. A research assistant robot can read the literature and answer questions from a researcher, finding relevant information quickly. A meeting assistant robot can record the conversation, automatically generating meeting minutes and answer queries concerning the meeting details. A teaching assistant robot can record classroom activities and grade homework automatically, telling the students where to put more effort to.

Chapter 2 Reference Answers

2.1 The outcomes for casting a dice are: {1,2,3,4,5,6}
The parameters for each outcome are: $\{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6\}$
There is also a constraint for the parameters: $\sum_{i=1}^{6} \theta_i = 1$
Assume that k_i out of N samples have outcome i in a dataset. Using Maximum Likelihood Estimation,

$$\log P(D) = \sum_{i=1}^{6} k_i \log \theta_i$$

$$\Lambda = \sum_{i=1}^{6} k_i \log \theta_i + \lambda \left(\sum_{i=1}^{6} \theta_i - 1 \right) \ (Lagrangian\ equation)$$

$$\frac{\partial \Lambda}{\partial \theta_i} = \frac{k_i}{\theta_i} + \lambda = 0 \Rightarrow \theta_i = -\frac{k_i}{\lambda}$$

$$\sum_{i=1}^{6} \theta_i = 1, \lambda = -\mathrm{N}$$

It can be derived that $\theta_i = \frac{k_i}{N}$.

2.2 We take language models for an example. The unigram language model is parameterised by word probabilities, which we already know how to estimate. Formally, a unigram LM consists of only one parameter type, which is the probability of a word. On the other hand, it contains $|V|$ parameter instances, corresponding to the probability of each word in a vocabulary V.

2.3 **Unigram**: Tim
Bigram: Tim bought; bought a; for $1
Trigram:a book for

2.4 (a) $P(\text{ all } | < \text{s} >) = \frac{1}{3}, P(\text{a} | < \text{s} >) = \frac{1}{3}, P(\text{ some } | < \text{s} >) = \frac{1}{3}$
$P(\text{ models } | \text{ all }) = 1, P(\text{ model } | \text{ a}) = 1, P(\text{ models } | \text{ some }) = 1$
$P(\text{ are } | \text{ models }) = 1, P(\text{ wrong } | \text{ are }) = \frac{1}{2}, P(\text{ useful } | \text{ are }) = \frac{1}{2}$

$P(< /s >|\text{ wrong }) = 1, P(< /s >|\text{ useful }) = 1$

(b) We can assume that all possible bigrams are in the bigram vocabulary before doing smoothing, adding 1 to the count of each bbigram. Take $P(*|\langle s\rangle)$ for example. 10 different words can follow $\langle s\rangle$, including$\langle /s\rangle$), *a*, *all*, *are*, *is*, *model*, *models*, *some*, *useful* and *wrong*. In the dataset, $\langle s\rangle$ *all*, $\langle s\rangle$ *a* and $\langle s\rangle$ *some* are seen, with a count of 1 each. Thus after add one smoothing their counts become 2. The other 7 bigrams were unseen but receive count 1 after smoothing. Thus

$$P(all|\langle s\rangle) = P(a|\langle s\rangle) = P(some|\langle s\rangle) = 2/13$$

$$P(\langle /s\rangle|\langle s\rangle) = ... = 1/13$$

(c) $P(\text{ models | all }) = P(\text{ model | a}) = P(\text{ models | some }) = P(< /s >|\text{ wrong }) = P(< /s >|\text{ useful }) = \frac{1}{10}$
$P(\text{ all }|< s >) = P(a\ |< s >) = P(\text{ some }|< s >) = \frac{3}{10}$
$P(\text{ wrong | are }) = P(\text{ useful | are }) = \frac{21}{50}$
$P(\text{ models | } a) = \frac{1}{30}$
When $\alpha = 0.05$, the results remain the same except: $P(\text{ model | a}) = \frac{1.05}{1.1}$, $P($ models | a$) = \frac{0.05}{1.1}$
When $\alpha = 0.15$, the results remain the same except: $P(\text{ model | a}) = \frac{1.15}{1.3}$, $P($ models | a$) = \frac{0.05}{1.3}$

(d) $P(< s >) = \frac{3}{18} = \frac{1}{6}; P(< /s >) = \frac{3}{18} = \frac{1}{6}$
$P(a) = \frac{1}{18}; P(all) = \frac{1}{18}; P(models) = \frac{1}{18}$
$P(models) = \frac{2}{18} = \frac{1}{9}; P(are) = \frac{2}{18} = \frac{1}{9}; P(some) = \frac{1}{18}$
$P(useful) = \frac{1}{18}; P(wrong) = \frac{2}{18} = \frac{1}{9}; P(is) = \frac{1}{18}$
$P(all|< s >) \approx \lambda * \frac{1}{3} + (1-\lambda) * P(all)$
When $\lambda = 0.95$, $P(all|< s >) = 0.95 * \frac{1}{3} + 0.05 * \frac{1}{18} = 0.319$
When $\lambda = 0.75$, $P(all|< s >) = 0.75 * \frac{1}{3} + 0.25 * \frac{1}{18} = 0.264$

2.5 (a)

There are 11 unigrams in Exercise 2.4. $< s >$ and $< /s >$ occur 3 times; *models*, *are* and *wrong* occur 2 times; *a*, *all*, *is*, *model*, *some* and *useful* occur 1 time. In addition, $\langle$INK$\rangle$ occurs 0 times.

As a result, $N_3 = 2$, $N_2 = 3$, $N_1 = 6$, $N_0 = 1$ and $N_r = 0$ for $r > 3$.

(b)

$$c_3 = 4 \times \frac{N_4}{N_3} = 0;\ c_2 = 3\frac{N_3}{N_2} = 2;\ c_1 = 2\frac{N2}{N_1} = 1;\ c_0 = 1\frac{N_1}{N_0} = 6;$$

$$\sum_r c_r = 18$$

P(w) = 0 for all n-grams that occur 3 times (i.e.,$< s >$ and $< /s >$).
P(w) = 2/18 for all n-grams that occur 2 times (i.e., models, are and wrong).
P(w) = 1/18 for all n-grams that occur 1 time (i.e., a, all, is, model, some, useful)
P($< unk >$) = 6/18
Note that the unknown word receives a relatively high probability because we assumed that there is only one distinct unseen word, which is different from the realistic scenario discussed in Section 2.2.3.
(c) According to the previous question the probabilities of $\langle s \rangle$ and $\langle /s \rangle$ are both 0. We calculate their probabilities in this question without discounting frequencies. Therefore, the probabilities are calculated using the original counts, equaling 3/18.
(d) In this section, we denote P(N_r) as the probability of words that appears r times in the corpus.

Thus, the sum of all unigrams is as follows:

$$\sum_r P(w_r) = P(w_0)+P(w_1)+P(w_2)+P(w_3)+P(w_4) = \frac{6}{18}+\frac{1}{18}+\frac{2}{18}+0+0 = \frac{9}{18} = \frac{1}{2}$$

The normalization process is as follows:

$$P(w_r) = \frac{P(w_r)}{\sum_r P(w_r)}$$

Thus, the normalized probability is as follows:

$$P(w_0) = \frac{2}{3};\ P(w_1) = \frac{1}{9};\ P(w_2) = \frac{2}{9}$$

(e)

We redistribute this total count into unigrams with frequencies 3, 4 and 5, ensuring that the lower frequency unigrams have higher counts.

In this case, $N_5 = 0.5, N_4 = 0, N_3 = 1.5, N_2 = 2$ and $N_1 = 6$. We can reestimate N_4 using $N-5$ and N_3, where $N_4 = (N_5+N_3)/2 = 1.5$. As a result,

$$c_4 = 5\frac{N_5}{N_4} = 10/3;\ c_3 = 4\frac{N_4}{N_3} = 3;\ c_2 = 3\frac{N_3}{N_2} = 3;\ c_1 = 2\frac{N_2}{N_1} = 2/3;\ c_0 = 1\frac{N_1}{N_0} = 6$$

The total count is $= N_5 \times c_5 + N_4 \times c_4 + N_3 \times c_3 + n_2 \times c_2 + N_1 \times c_1 + N_0 \times c_0 = 1\times 0 + 0\times(10/3) + 2\times 3 + 2\times 3 + 6\times 2/3 + 1\times 6 = 22$

$P(w) = 3/22$ for words occurring 3 times. $P(w) = 3/22$ for words occurring 2 times. $P(w) = (10)/22 = 5/11$ for words occurring 1 time. $P(unk) = 6/22 = 3/11$ for unknown words.

Following question (c), we separately calculate the probability of *wrong* using the original counts, which is 6/21 = 2/7.

Thus the probabilities can be normalised as:

$P(w) = \frac{3}{22}$ for the words that occurs 5 times; $P(w) = \frac{3}{22}$ for words that occur 3 times; $P(w) = \frac{3}{22}$ for words that occur 2 times; $P(w) = \frac{2}{22} = \frac{1}{11}$ for words that occur 1 time; $P(unk) = \frac{3}{22}$ for words not in the vocabulary.

2.6 (a) For each $|u|$, a probability mass is $\frac{\delta}{\sum_w \#(wu)}$ discounted for each $|w|$ such that $uw \in D$. Therefore, the total discounted value is $1 - \sum_w P_{AD}(w|u) = \frac{\delta}{\sum_w \#(uw)} \mid \{w : wu \in D\} \mid = \frac{\delta}{\sum_u} \mid \{w : wu \in D\} \mid$.
In addition,

$$\begin{aligned}\sum_w P(w \mid u) &= \sum_w P_{AD}(w \mid u) + \sum_w \lambda_u P_{KN}(w) \\ &= \sum_w P_{AD}(w \mid u) + \lambda_u \left(\sum_w P(w) = 1 \right) \\ &= 1\end{aligned}$$

Thus,

$$\lambda_u = \frac{\delta}{\#u} \mid \{w : uw \in D\} \mid$$

(b)

	unigrams	**bigrams**
$C_{KN}(\langle s \rangle) = 0$	$\#\langle s \rangle = 3$	$\#(\langle s \rangle\ all) = 1 \quad \#(\langle s \rangle\ all) = 1$
$C_{KN}(\langle /s \rangle) = 3$	$\#\langle /s \rangle = 3$	$\#(\langle s \rangle\ some) = 1$
$C_{KN}(all) = 1$	$\#\langle all \rangle = 1$	$\#(all\ models) = 1$
$C_{KN}(a) = 1$	$\#\langle a \rangle = 1$	$\#(a\ model) = 1$
$C_{KN}(are) = 1$	$\#\langle are \rangle = 2$	$\#(are\ useful) = 1$
$C_{KN}(is) = 1$	$\#\langle is \rangle = 1$	$\#(is\ wrong) = 1$
$C_{KN}(model) = 1$	$\#\langle model \rangle = 1$	$\#(model\ is) = 1$
$C_{KN}(models) = 2$	$\#\langle models \rangle = 2$	$\#(models\ are) = 2$
$C_{KN}(some) = 1$	$\#\langle some \rangle = 1$	$\#(some\ models) = 1$
$C_{KN}(useful) = 1$	$\#\langle useful \rangle = 1$	$\#(useful\ \langle /s \rangle) = 1$
$C_{KN}(wrong) = 2$	$\#\langle wrong \rangle = 2$	$\#(wrong\ \langle /s \rangle) = 2$

$$P_{AD}(all \mid \langle s \rangle) = \frac{1-0.75}{3} = \frac{1}{12}$$
$$P_{AD}(a \mid \langle s \rangle) = \frac{1-0.75}{3} = \frac{1}{12}$$
$$P_{AD}(\text{ some } \mid \langle s \rangle) = \frac{1-0,75}{3} = \frac{1}{12}$$
$$P_{AD}(\text{ models } \mid \text{ all }) = \frac{1-0.75}{1} = \frac{1}{4}$$
$$P_{AD}(\text{ are } \mid \text{ models }) = \frac{2-0.75}{2} = \frac{5}{8}$$
$$P_{AD}(\text{ wrong } \mid \text{ are }) = \frac{1-0.75}{2} = \frac{1}{8}$$
$$P_{AD}(\langle /s \rangle \mid wrong) = \frac{2-0.75}{2} = \frac{5}{8}$$
$$P_{AD}(\text{ model } \mid \text{ a }) = \frac{1-0.75}{1} = \frac{1}{4}$$
$$P_{AD}(\text{ is } \mid \text{ model }) = \frac{1-0.75}{1} = \frac{1}{4}$$
$$P_{AD}(\text{ wrong } \mid \text{ is }) = \frac{1-0.75}{1} = \frac{1}{4}$$
$$P_{AD}(\text{ models } \mid \text{ some }) = \frac{1-0.75}{1} = \frac{1}{4}$$
$$P_{AD}(\text{ useful } \mid \text{ are }) = \frac{1-0.75}{2} = \frac{1}{8}$$
$$P_{AD}(\langle /s \rangle \mid useful) = \frac{1-0.75}{1} = \frac{1}{4}$$

(c)

λ	P_{KN}
$\lambda_{\langle/s\rangle} = \frac{0.75}{3} \times 0 = 0$	$P_{KN}(\langle/s\rangle) = 0$
$\lambda_{all} = 0.75$	$P_{KN}(all) = \frac{1}{14}$
$\lambda_{a} = 0.75$	$P_{KN}(a) = \frac{1}{14}$
$\lambda_{\text{are}} = 0.75$	$P_{KN}(are) = \frac{1}{14}$
$\lambda_{is} = 0.75$	$P_{KN}(is) = \frac{1}{14}$
$\lambda_{\text{model}} = 0.75$	$P_{KN}(model) = \frac{1}{14}$
$\lambda_{\text{models}} = \frac{0.75}{2} \times 1 = \frac{3}{8}$	$P_{KN}(models) = \frac{1}{7}$
$\lambda_{some} = 0.75$	$P_{KN}(some) = \frac{1}{14}$
$\lambda_{useful} = 0.75$	$P_{KN}(useful) = \frac{1}{14}$
$\lambda_{wrong} = \frac{0.75}{2} \times 1 = \frac{3}{8}$	$P_{KN}(wrong) = \frac{1}{7}$

$$P(all \mid \langle S\rangle) = P_{AD}(all \mid \langle s\rangle) + \lambda_{\langle s\rangle} P_{kN}(all) = \frac{1}{12} + 0.75 * \frac{1}{14} = \frac{23}{168}$$

$$P(a \mid \langle S\rangle) = P_{AD}(a \mid \langle s\rangle) + \lambda_{\langle s\rangle} P_{kN}(a) = \frac{1}{12} + 0.75 * \frac{1}{14} = \frac{23}{168}$$

$$P(some \mid \langle S\rangle) = P_{AD}(some \mid \langle s\rangle) + \lambda_{\langle s\rangle} P_{kN}(some) = \frac{1}{12} + 0.75 * \frac{1}{14} = \frac{23}{168}$$

$$P(models \mid all) = P_{AD}(models \mid \langle s\rangle) + \lambda_{all} P_{kN}(models) = \frac{1}{4} + 0.75 * \frac{1}{7} = \frac{5}{14}$$

$$P(are \mid models) = P_{AD}(are \mid \langle s\rangle) + \lambda_{models} P_{kN}(are) = \frac{5}{8} + \frac{3}{8} * \frac{1}{14} = \frac{73}{112}$$

$$P(wrong \mid are) = \frac{1}{8} + \frac{3}{4} * \frac{1}{7} = \frac{13}{56} \qquad P(wrong \mid is) = \frac{1}{4} + \frac{3}{4} * \frac{1}{7} = \frac{5}{14}$$

$$P(\langle/s\rangle \mid wrong) = \frac{1}{8} + \frac{3}{4} * \frac{1}{7} = \frac{5}{8} \qquad P(models \mid some) = \frac{1}{4} + \frac{3}{4} * \frac{1}{7} = \frac{5}{14}$$

$$P(model \mid a) = \frac{1}{8} + \frac{3}{4} * \frac{1}{7} = \frac{17}{56} \qquad P(useful \mid are) = \frac{1}{4} + \frac{3}{4} * \frac{1}{7} = \frac{5}{28}$$

$$P(is \mid model) = \frac{1}{8} + \frac{3}{4} * \frac{1}{7} = \frac{17}{56} \qquad P(\langle/s\rangle \mid useful) = \frac{1}{4} + \frac{3}{4} * \frac{1}{7} = \frac{1}{4}$$

2.7 (a)True
(b)False
(c)True
(d)False
(e)True

2.8 **Parameters** can be automatically learned from the data

Hyper-Parameters are the parameters that used to determine the model configuration, like learning rate. They are pre-defined before model training. Different hyper-parameters usually lead to different models.

2.9 Hyper-parameters are tuned over (b) development data

2.10 Yes.

The form of Naive Bayes text classifier could be derived as follows:

$$P(c \mid d) = \frac{P(d \mid c)P(c)}{P(d)} = \frac{P(d \mid c)P(c)}{\sum_{c' \in C} P(d \mid c')P(c')}$$

$$P(d \mid c) = P(w_1 w_2 w_1 \ldots w_n \mid c) = P(w_1 \mid c)\, P(w_2 \mid w_1, c) \ldots P(w_n \mid w_1 \ldots w_{n-1}, c)$$

Different from the Naive Bayes text classifier mentioned in Section 2.3.1, we use the concept of bigram language modelling here instead of assuming all the words are conditionally independent. This results in the following formula:

$$P(d \mid c) = P(w_1 \mid c)\, P(w_2 \mid w_1, c)\, P(w_3 \mid w_2, c) \ldots P(w_n \mid w_{n-1}, c)$$

So the final form of Naive Bayes text classifier is as follows:

$$P(c \mid d) \propto P(d \mid c)P(c) \approx \prod P(w_i \mid w_{i-1}, c)\, P(c)$$

Given $D = \{(d_i, c_i)\}_{i=1}^{N}$, the probability $P(c)$ is typically not sparse and can be estimated with MLE as follows:

$$P(c) = \frac{\#c \in D}{\sum_{c'} (\#c' \in D)} = \frac{\#c \in D}{|D|}$$

$P(w \mid c)$ for each word or n-gram and c pair could also be estimated with MLE as follows:

$$P(w_i \mid w_{i-1}, c) = \frac{\#(w_{i-1} w_i, c) \in D}{\sum_{w \in V} (\#(w_{i-1} w, c) \in D)}$$

Then, Back-off could be adopted here to reduce sparisity of a bigram $w_1 w_2$. When a bigram is unseen, the probability $P(w_2 \mid w_1)$ can be replaced with $P(w_2)$. The new probability is as follows:

$$P_{\text{backoff}}\ (w_2 w_1) = \lambda P(w_2 \mid w_1) + (1 - \lambda)\lambda P(w_2)$$

2.11 (a)Bag-of-Bigrams is a subset of bag-of-words

(b)Prefix and suffix are subset of features of full word

(d)The first letter of a word contains two features: 1.whether it is capitalized 2.the letter. So the first feature is overlapped with the latter feature.

(f) word class pairs contains word information

Chapter 3 Reference Answers

3.1

(a) See Table 1.

$v(d_1) = \langle 0,1,0,0,1,1,1,0,0 \rangle$

$v(d_2) = \langle 0,1,0,0,1,1,1,0,0 \rangle$

$v(d_3) = \langle 0,1,0,0,1,1,1,0,0 \rangle$

(b) Random initialize d_1 and d_2 as cluster center point c_1 and c_2.

$dist(d_3,c_1) = \sqrt{1^2+1^2+1^2+1^2+1^2} = \sqrt{5}$

$dist(d_3,c_2) = \sqrt{1^2+1^2+1^2+1^2+1^2+1^2+1^2+1^2+1^2} = 2\sqrt{2}$

Thus there are two clusters, the first consisting of d_1 and d_3, and the second consisting of d_2. Then, calculate cluster centers:

c_1=average(d_1,d_3)=(0,0,$\frac{1}{2}$,$\frac{1}{2}$,1,1,$\frac{3}{2}$,$\frac{1}{2}$,0)

c_2=average(d_2,d_3)=(1,0,0,0,0,0,1,0,1)

The same process repeats.

$dist(d_1,c_1) = \sqrt{1^2+\frac{1}{2}^2+\frac{1}{2}^2+\frac{1}{2}^2+\frac{1}{2}^2} = \sqrt{2}$

$dist(d_3,c_1) = \sqrt{\frac{1}{2}^2+\frac{1}{2}^2+\frac{1}{2}^2+\frac{1}{2}^2} = 1$

$dist(d_1,c_2) = \sqrt{1^2+1^2+1^2+1^2+1^2} = \sqrt{5}$

$dist(d_3,c_2) = \sqrt{1^2+1^2+1^2+1^2+1^2+1^2+1^2+1^2+1^2} = 2\sqrt{2}$

Thus $d_1 \in cluster1; d_2 \in cluster2; d_3 \in cluster1$.

The clusters stabilize and the algorithm stops.

(c)See Table 2

Using add-one smoothing.

$P(c_1 \mid \text{the dog sat}) = P(c_1)P(the \mid c_1)P(dog \mid c_1)P(sat \mid c_1) = \frac{1}{64}$

$P(c_2 \mid \text{the dog sat}) = P(c_2)P(the \mid c_2)P(dog \mid c_2)P(sat \mid c_2) = \frac{1}{512}$

$P(c_1 \mid \text{the dog sad }) > P(c_2 \mid \text{the dog sad })$ Thus "the dog sat" belongs to the first class

(d)

c_1 = animal, c_2 = vehicle

Let us define the feature template as

$v(d_1,c_1) =< 0,1,0,0,1,1,1,0,0\ 0,0,0,0,0,0,0,0,0 >$

$v(d_1,c_2) =< 0,0,0,0,0,0,0,0,0,\ 0,1,0,0,1,1,1,0,0 >$

$v(d_2\ c_1) = \langle 1,0,0,0,0,0,1,0,1\ 0,0,0,0,0,0,0,0,0 >$

$v(d_2,c_2) =< 0,0,0,0,0,0,0,0,0,\ 1,0,0,0,0,0,1,0,1 >$

$v(d_3,c_1) =< 0,0,1,1,1,1,2,1,0\ 0,0,0,0,0,0,0,0,0 >$

documents \ words	**cat**	**dog**	**car**	**bus**	**ran**	**fast**	**the**	**and**	**sat**
d_1	0	1	0	0	1	1	1	0	0
d_2	1	0	0	0	0	0	1	0	1
d_3	0	0	1	1	1	1	2	1	0

Table 1.1: 3.1(a) count-based vectors

words \ classes	**cat**	**dog**	**car**	**bus**	**ran**	**fast**	**the**	**and**	**sat**
c_1	$\frac{1+1}{7+1}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{3}{8}$	$\frac{1}{8}$	$\frac{1}{4}$
c_2	$\frac{0+1}{7+1}$	$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{3}{8}$	$\frac{1}{4}$	$\frac{1}{8}$

Table 1.2: 3.1(c) unigram features

$v(d_3,c_2) =< 0,0,0,0,0,0,0,0,0,0, 0,0,1,1,1,1,2,1,0 >$.
$v(\mathrm{d_4},\mathrm{c_1}) =< 0,1,0,0,0,0,1,0,1, 0,0,0,0,0,0,0,0,0,0 >$
$v(\mathrm{d_4},\mathrm{c_2}) =< 0,0,0,0,0,0,0,0,0,0, 0,1,0,0,0,0,1,0,1 >$

3.2

$d\backslash w$	*a*	*ah*	*book*	*bought*	*boy*	*I*	*is*	*know*	*ready*	*saw*	*Tim*	,	.
d_1	0.415	0	0.415	2.0	0	0	0	0	0	0	0.415	0	1.0
d_2	0.415	0	0.415	0	0	0	2.0	0	1.0	0	0.415	0	0
d_3	0	2.0	0	0	0	1.0	0	2.0	0	0	0.415	2.0	1.0
d_4	0.83	0	0.415	0	2.0	1.0	0	0	1.0	2.0	0	0	0

2-means:
$cluster_1 = \{d_1,d_2,d_4\}$, $c_1 = \langle 0.553,0,0.415,0.667,0.667,0.333,0.667,0,0.667,$
$0.667,0.277,0,0.333\rangle$,
$cluster_2 = d_3, c_2 = \langle 0,2.0,0,0,0,1.0,0,2.0,0,0,0.415,2.0,1.0\rangle$
$dist(d_1,c_1) < dist(d_1,c_2)$, therefore $d_1 \in cluster_1$
$dist(d_2,c_1) < dist(d_2,c_2)$, therefore $d_2 \in cluster_1$
$dist(d_4,c_1) < dist(d_4,c_2)$, therefore $d_4 \in cluster_1$
$dist(d_3,c_1) < dist(d_3,c_2)$, therefore $d_3 \in cluster_1$
Thus the current clusters are stable.
3-means:
$cluster_1 = \{d_1,d_2\}$, therefor $c_1 = \langle 0.415,0,0.415,1.0,0,0,1.0,0,0.5,0,0.415,0,0.5\rangle$
$cluster_=\{d_1,d_2\}$, therefor $c_1 = \langle 0.415,0,0.415,1.0,0,0,1.0,0,0.5,0,0.415,0,0.5\rangle$
$min\{dist(d_1,c_1),dist(d_1,c_2),dist(d_1,c_3)\} = dist(d_1,c_1)$, therefore $d_1 \in cluster_1$
$min\{dist(d_2,c_1),dist(d_2,c_2),dist(d_2,c_3)\} = dist(d_2,c_1)$, therefore $d_2 \in cluster_1$
$min\{dist(d_3,c_1),dist(d_3,c_2),dist(d_3,c_3)\} = dist(d_3,c_2)$, therefore $d_3 \in cluster_2$
$min\{dist(d_4,c_1),dist(d_4,c_2),dist(d_4,c_3)\} = dist(d_4,c_3)$, therefore $d_4 \in cluster_3$
Thus the current clusters are stable.

3.3 Assuming that the number of features is N and the average number of non-zero features is M, the hash table complexity is $O(M)$, which is far less than the

array complexity $O(N)$.

3.4 (a)Theoretically, Naive Bayes takes longer to train because of the need to compute probability parameters; in contrast, kNN training is trivial, involving the saving of new training instances into the vector space. For testing, the cost for Naive Bayes lies mainly on the computing of joint probabilities, while that for kNN lies mainly on the finding of nearest neighbors in the vector space. Given a large number of training instances, the latter can be computationally more expensive despite that various algorithms have been developed to optimise this process.(see for example https://github.com/facebookresearch/DPR)
(b) No, this can be a big advantage.
(c) Yes. For instance, suppose that the closest neighbor of a test instance belongs to class c_1, but the second and third closest neighbors belong to class c_2. For $k = 1$ the output is c_1 but for $k = 3$ the output is c_2. In practice, the value of k can be chosen empirically over a set of development data, or using cross-validation.

3.5 First, you have an affine hyperplane defined by $\overrightarrow{w} \cdot \overrightarrow{v} + b = 0$ and a vectorv_0. Suppose that $\overrightarrow{V_p}$ is a vector satisfying $\overrightarrow{w} \cdot \overrightarrow{V_p} + b = 0$ (i.e. it is a vector on the plane).

$$d = \left\| \text{proj}_w \left(v_0 - \overrightarrow{V_p} \right) \right\| = \left\| \frac{\left(v_0 - \overrightarrow{V_p}\right) \cdot w}{w \cdot w} w \right\| = \left| v_0 \cdot w - \overrightarrow{V_p} \cdot w \right| \frac{\|w\|}{\|w\|^2} = \frac{\left| v_0 \cdot w - \overrightarrow{V_p} \cdot w \right|}{\|w\|}$$

Because $w \cdot \overrightarrow{V_p} = -b$ ($proj_w$ means the projection of a vector on a plane is its orthogonal projection on that plane.)

$$d = \|\text{proj}_w (v_0 - V)\| = \frac{|v_0 \cdot w + b|}{\|w\|} \qquad Q.E.D$$

3.6 (a) $|\,c\,| + |\,w\,| \cdot |\,c\,| + |\,b\text{i}\,| \cdot |\,c\,|$.
(b) The number of c will not change because train set will contain all classes. The number of wc can reduce much because not all words coexist with all class labels. The number of bic will drop significantly.
(c) (1) if a feature instance in an unseen test sample is not an element in the feature vector defined using the training data, the unseen test feature instance will be ignored. If the missing feature is a crucial feature for deciding the class label, the model can make a wrong decision.
(d) Negative features can do some extend ameliorate the loss from unseen feature instances, because (1) they add to the set of model feature instances and (2) they are features from output samples that the model tends to make.

(e) The feature vector is a large sparse vector with the elements corresponding to $\{c = hobby\}, \{w = A, c = hobby\}, \{w = cat, c = hobby\}, \{w = sat, c = hobby\}, \{w = on, c = hobby\}, \{w = the, c = hobby\}, \{w = mat, c = hobby\}, \{w = ., c = hobby\}, \{bi =$

$Acat, c = hobby\}, \{bi = catsat, c = hobby\}, \{bi = saton, c = hobby\}, \{bi = onthe, c = hobby\}, \{bi = themat, c = hobby\}$ and $\{bi = mat., c = hobby\}$ being 1, and the other elements being 0.

(f) The feature vector is a large sparse vector with the elements corresponding to $\{c = sports\}, \{w = The, c = sports\}, \{w = cat, c = sports\}, \{w = sat, c = sports\}, \{w = on, c = sports\}, \{w = the, c = sports\}, \{w = mat, c = sports\}, \{w = ., c = sports\}, \{bi = The\ cat, c = sports\}, \{bi = cat\ sat, c = sports\}, \{bi = sat\ on, c = sports\}, \{bi = on\ the, c = sports\}, \{bi = the\ mat, c = sports\} and \{bi = mat\ ., c = sports\}$, of which $\{w = cat, c = sports\}, \{w = sat, c = sports\}, \{w = on, c = sports\}, \{w = the, c = sports\}, \{w = mat, c = sports\}, \{w = ., c = sports\}, \{bi = cat\ sat, c = sports\}, \{bi = sat\ on, c = sports\}, \{bi = on\ the, c = sports\}, \{bi = the\ mat, c = sports\}$ and $\{bi = mat\ ., c = sports\}$ are likely bad features.

3.7 (a)

$$P(y \mid x) = P(y \mid w, c) \propto P(y) \cdot P(w, c \mid y) = P(y) \cdot P(w \mid y) \prod_{w' \in c} P(w' \mid y)$$

in which w is the target word and $w' \in c$ is a context word.
(b) The bag of features includes the features in (a) and also collocational features. The model is not a theoretically correct generative model because there are overlapping features.
(c)

$$|C| \cdot |V| \cdot (2k+1) + |C|$$

(d)

$$|C|(|L| + |V|) \cdot (2k+1) + |C|$$

3.8

$$\left(\vec{w}^\top \vec{v}(x_i, c_i) + b_{c_i}\right) - \left(\vec{w}^\top \vec{v}(x_i, \mathrm{c}) + b_c\right) \geq 2 \text{ for all } c \neq c_i$$
$$\Rightarrow \vec{w}^\top \vec{v}(x_i, c_i) - \vec{w}^\top \vec{v}(x_i, c) \geq 2 + b_c - b_{c_i}, \text{ for all } c \neq c_i$$

Score margin 2 is set to $1 \Rightarrow \vec{w} = \arg\min_{\vec{w}} \frac{1}{2}\|w\|^2$, such that $\vec{w}^\top \vec{v}(x_i, c_i) - \vec{w}^T \vec{v}(x_i, c) \geq 1 + b_c - b_{c_i}$. If we treat $\vec{w'} = \vec{w} \bigoplus < 1 >$ and $v'(\vec{x_i}, c_i) = v(x_i, c_i) \bigoplus < b_{c_i} >$ where $\bigoplus$ denotes vector concatenation, then b_{c_i} corresponds to $P(c)$ in Naive Bayes.

Chapter 4 Reference Answers

4.1 They differ in two main aspects. First, SVM updates only happens in the condition that:

$$\vec{\theta} \cdot \left(\vec{\phi}\left(x_i, y_i\right) - \vec{\phi}\left(x_i, c\right) \right) < 1$$

In contrast, Log-linear models update for every batch of data based on the update rule of SGD according to the value of

$$\left(\phi\left(\vec{x_i}, c\right) \right) - \left(\phi\left(\vec{x_i}, y_i\right) \right)$$

Second, the amount of parameter update is fixed for SVM, where only the gold output and the most violated constraints are used to update model parameters. In contrast, for log-linear models, every possible output is used to update model parameters, with the update scaled by a probabilistic factor.

4.2 Compared with log-linear models, the indicator loss leads to more output classes potentially participating in the parameter update, but still depending on whether there is a violation (i.e., the incorrect output score is higher than the score of the correct output). The update rule, on the other hand, is still similar to the perceptron update in the sense that no probabilistic weights are added to feature vectors when they are used for updating model parameters.

(a) For multi-class SVM with indicated loss

$$\begin{aligned}
\frac{\partial L}{\partial \vec{\theta}} &= \frac{\partial \left(\sum_{i=1}^{N} \sum_{c} \max\left(0, 1 - \vec{\theta} \cdot \vec{\phi}\left(x_i, y_i\right) + \vec{\theta} \cdot \vec{\phi}\left(x_i, c\right)\right) + \frac{1}{2}\lambda \|\vec{\theta}\|^2 \right)}{\partial \vec{\theta}} \\
&= \frac{\partial \left(\sum_{i=1}^{N} \sum_{c} \max\left(0, 1 - \vec{\theta} \cdot \vec{\phi}\left(x_i, y_i\right) + \vec{\theta} \cdot \vec{\phi}\left(x_i, c\right)\right) \right)}{\partial \vec{\theta}} + \frac{\partial \left(\frac{1}{2}\lambda \|\vec{\theta}\|^2 \right)}{\partial \vec{\theta}} \\
&= \sum_{i=1}^{N} \sum_{c} \max\left(0, -\vec{\phi}\left(x_i, y_i\right) + \vec{\phi}\left(x_i, c\right)\right) + \lambda \vec{\theta}
\end{aligned}$$

(b) See Algorithm 1. For SVM with indicator loss function, the update rule using SGD training algorithm is shown in Algorithm 1. Compared with log-linear models, the indicator loss leads to more output classes potentially participating in

the parameter update, but still depending on whether there is a violation (i.e., the incorrect output score is higher than the score of the correct output). The update rule, on the other hand, is still similar to the perceptron update in the sense that no probabilistic weights are added to feature vectors when they are used for updating model parameters.

Algorithm 1: SGD Training for multi-class SVM with Indicator Loss

Inputs: $D = \{(x_i, y_i)\}_{i=1}^{N}, y_i \in C$;
Outputs: $\vec{\theta}$;
Initialisation: $\vec{\theta} \leftarrow 0, t \leftarrow 0$;
while $t < T$ **do**
  for $i \in [1, \ldots, N]$ **do**
    $\vec{g} \leftarrow \vec{\theta}$;
    for $c \in C$ **do**
      $z_i \leftarrow c$;
      if $\vec{\theta} \cdot \phi(\vec{x_i}, y_i) - \vec{\theta} \cdot \phi(\vec{x_i}, z_i) < 1$ **then**
        $\vec{g} \leftarrow \vec{g} - \left(\phi(\vec{x_i}, z_i) - \phi(\vec{x_i}, y_i)\right)$;
    $\vec{\theta} \leftarrow \vec{\theta}(\vec{g} + \lambda\vec{\theta})$;
  $t \leftarrow t + 1$;

4.3 After adding the *L*2 regularisation term, the loss function is as follows:

$$Loss = -\sum_i \left(\vec{\theta} \cdot \varphi(x_i, y_i)\right) - \sum_i \log\left(\sum_{c \in C} e^{\vec{\theta} \cdot \varphi(x_i, c)}\right) + \frac{\lambda}{2}\|\vec{\theta}\|^2$$

So, the gradient can be written as:

$$\begin{aligned}\frac{\partial \text{ Loss}}{\partial \vec{\theta}} &= -\left(\sum_i \vec{\varphi}(x_i, y_i) - \sum_i \frac{\sum_{c \in C} e^{\vec{\theta} \cdot \varphi(x_i, c)} \cdot \vec{\varphi}(x_i, c)}{\sum_{c \in C} e^{\vec{\theta} \cdot \varphi(x_i, c)}}\right) + \lambda\vec{\theta} \\ &= -\sum_i \sum_{c \in C} \left(\vec{\varphi}(x_i, y_i) - \vec{\varphi}(x_i, c)\right) P(y = c \mid x_i) + \lambda\vec{\theta}\end{aligned}$$

Then, the parameters can be updated as follows:

$$\vec{g} = \vec{g} - \sum_{c \in C} \left(\vec{\varphi}(x_i, c) - \vec{\varphi}(x_i, y_i)\right) P(y = c \mid x_i) - \lambda\vec{\theta}$$

4.4 SVM can be almost viewed as a perceptron regularized with L2 regularisation because SVM optimises the following function:

$$\min_{w,b} \frac{1}{N} \sum_{i=1}^{N} \max(0, 1 - y_i(w \cdot x_i + b)) + \lambda \|w\|_2^2$$

The optimization problem of standard perception is as follows:

$$\min_{w,b} \frac{1}{n} \sum_{i=1}^{n} \max\left(0, -y_i\left(w \cdot x_i + b\right)\right)$$

After adding L2 regularization to the perception the optimization objective is as follows:

$$\min_{w,b} \frac{1}{n} \sum_{i=1}^{n} \max\left(0, -y_i\left(w \cdot x_i + b\right)\right) + \lambda \|w\|_2^2 \ ,$$

which is similar to SVM except for the score margin (0 vs 1).

4.5 (a)local update

$$\vec{g_k} \leftarrow \begin{cases} \vec{g_k} - \vec{\phi_k}(x_i, z_i) + \lambda \, if \, \theta_k > 0 \\ \vec{g_k} - \vec{\phi_k}(x_i, z_i) - \lambda \, if \, \theta_k < 0 \end{cases} ,$$

where z_i is the highest-scored incorrect output. $\vec{\phi_k}$ is the k-th element in $\vec{\phi}$.

(b)local update

$$\vec{g_k} \leftarrow \vec{g_k} + \sum_{y} P(y \mid x_i)\left(\phi(x_i, y) - \phi(x_i, y_i)\right) - \lambda \|\vec{\theta}\|$$

(a) cannot be regarded as SVM, perceptron or log-linear model, (b) can be log-linear model with lazy update.

4.6 a) The percentage of unbalanced data could be used in this. Assume there are L data instances in the majority class (+1) and S data instances in the minority class (-1). Then, the threshold can be set as $\frac{L}{L+S}$.

Also, we could doing this empirically by calculating different F1 score corresponding to different threshold using development data. Picking the best threshold.

b) In general, we can set C^+ and C^- for giving the minority class a better chance. N^+ and N^- are positive and negative counts, respectively:

$$C^- \cdot N^+ = C^+ \cdot N^-$$

4.7 (a)

$$Loss = -\sum_i \left(\vec{\theta} \cdot \vec{\varphi}\left(x_i, y_i\right) - \log\left(\sum_{c \in C} e^{\vec{\theta} \cdot \phi(x_i, c)} \right) \right) + \lambda \|\theta\|$$

$$\frac{\partial \, \text{Loss}}{\partial \vec{\theta_k}} = \begin{cases} -\sum_{c \in C} \left(\vec{\varphi_k}\left(x_i, y_i\right) - \vec{\varphi_k}\left(x_i, c\right)\right) P\left(y = c \mid x_i\right) + \lambda, \text{ if } \theta_k \geq 0 \\ -\sum_{c \in C} \left(\vec{\varphi_k}\left(x_i, y_i\right) - \vec{\varphi_k}\left(x_i, c\right)\right) P\left(y = c \mid x_i\right) - \lambda, \text{if } \theta_k < 0 \end{cases} ,$$

where k is the k-th element in $\vec{\theta}$.
(b) See Algorithm 2 for binary classification with L2 regularisation.

Algorithm 2: Training for binary classification with L_2 regularization

Inputs:$D = \{(x_i, y_i)\}|_{i=1}^{N}$;
Initialisation:$\vec{\theta} \leftarrow \vec{0}$; $\alpha \leftarrow \alpha_0$; $\vec{\tau} \leftarrow 0$; t $\leftarrow 0$;
repeat
  for $i \in [1, \ldots, N]$ **do**
    $P(y = +1 \mid x_i) \leftarrow \frac{e^{\vec{\theta}\cdot\vec{\phi}(x_i)}}{1+e^{\vec{\theta}\cdot\vec{\phi}(x_i)}}$;
    if $y_i = +1$ **then**
      $r \leftarrow \alpha \cdot (P(y = +1 \mid x_i) - 1)$;
    else
      $r \leftarrow \alpha \cdot P(y = +1 \mid x_i)$;
    for $k \in NonZeroElementIndice(\vec{\phi}(x_i))$ **do**
      $\vec{\theta}[i] \leftarrow \vec{\theta}[i] - r \cdot (\vec{\phi}(x_i)[k])$;
      $\vec{\theta}[i] \leftarrow \vec{\theta}[i] - r \times \lambda \times \cdot (t + 1 - \vec{\tau}[i]) \times \theta[i]$ // regulariser;
      $\vec{\tau}[i] \leftarrow t + 1$ // last update;
    $t \leftarrow t + 1$
until $t = T$;
Outputs:$\vec{\theta}$;

4.8 See Algorithm 3

Algorithm 3: Tri-training

Inputs: $D = \{(x_i, y_i)\}|_{i=1}^{N}$, $U = \{x_i'\}|_{i=1}^{M}$, models A, B and C;
Initialisation: t $\leftarrow 0$;
repeat
  t $\leftarrow$ t+1;
  TRAIN(A, D);
  TRAIN(B, D);
  TRAIN(C, D);
  for $x_i\prime \in U$ **do**
    $z_A' \leftarrow$ PREDICT(A, x_i');
    $z_B' \leftarrow$ PREDICT(B, x_i');
    $z_C' \leftarrow$ PREDICT(C, x_i');
    if $z_A' = z_B' = z_i'$ **or** $z_B' = z_C' = z_i'$ **or** $z_A' = z_C' = z_i'$ **then**
      ADD(D, (x_i', z_i'));
      REMOVE(U, (x_i', z_i'));
until $t = T$;

4.9 See Algorithm 4

Algorithm 4: Unified semi-supervised algorithm

Inputs: $D = \{(x_i, y_i)\}|_{i=1}^{N_D}$, $U = \{x_i'\}|_{i=1}^{N_U}$, models $M = \{m_i\}_{i=1}^{N_M}$;
Initialisation: t $\leftarrow$ 0;
repeat
  t $\leftarrow$ t+1;
  for $m_i \in M$ **do**
    TRAIN(m_i, D);
  for $x_i\prime \in U$ **do**
    for $m_i \in M$ **do**
      $z_i^{m_i\prime} \leftarrow$ PREDICT(m_i, x_i');
    if $2 \leq N_M \leq 3$ **and** $z_i^{m_i\prime} = z_i^{m_j\prime}$ **and** CONFIDENT$(m_i, x_i', z_i^{m_i\prime})$ **and** CONFIDENT$(m_j, x_i', z_i^{m_j\prime})$ **then**
      ADD(D, $(x_i', z_i^{m_j\prime})$);
      REMOVE(U, $(x_i', z_i^{m_j\prime})$);
    else if $N_M = 1$ **and** CONFIDENT$(m_i, x_i', z_i^{m_i\prime})$ **then**
      ADD(D, $(x_i', z_i^{m_j\prime})$);
      REMOVE(U, $(x_i', z_i^{m_j\prime})$);
until $t = T$;

4.10 Bagging use different subsets of the whole dataset as training data and train k different models with the same models structure, using the k models to perform voting.

In contrast, ensemble use the same dataset for the training and use these models for voting.

They both use voting strategies to leverage different models.

Chapter 5 Reference Answers

5.1 (a)$log_2(\#w \in D)$ where D is the dictionary
(b)information("the")=information("zoo")
(c) information (" t* ") = $log_2(\#w \in D) - log_2(\#"t*" \in D)$ < information (" z* ") = $log_2(\#w \in D)$
(d) $H = -\sum_{w \in D} P(w) \cdot log_2 P(w) = log_2(\#w \in D)$
(e) information("the")<information("zoo"), because "the" is more frequent now in a corpus.

5.2 (a)AC
(b)BDEFGH
(c)ADE
(d)BCFGH
(e)E
(f)G

5.3 The maximum entropy principle leads to the log-linear model. Cross-entropy and KL-divergence have been used for defining the training objectives. Maximising the negative log-likelihood function of the model equals minimising the cross-entropy between model and data distributions. Thus, log-linear models sometimes can be regarded as maximum entropy models.

5.4 Yes. Given two random variables x for word w_i and y for classes overall mutual information: $L(x,y) = \sum P(x,y)_{x,y} \log_2 \frac{P(x,y)}{P(x)P(y)} = \mathrm{E_{x,y}} \left(\log_2 \frac{P(x,y)}{P(x)P(y)} \right)$, where PMI $\log_2 \frac{P(x,y)}{P(x)P(y)} = \log_2 P(x,y) - \log_2 P(x) - \log_2 P(y)$. By calculating the ratios between $P(x,y)$ and$P(x)$ and $P(y)$, we can find whether the word w_i is representative for the class c_j. And based on the $PMI(x,y) > 0$ or $PMI(x,y) < 0$, we can find the correlations between these two variables. In log linear models, the output based feature vectors $\vec{\phi}(x, c_j)$ has feature $\#(w_i, c_j)$. The probability of c_j is $P(c_j \mid x) = \frac{e^{\vec{\theta} \cdot \vec{\phi}\left(x, c_j\right)}}{\sum_{c' \in c} e^{\vec{\theta} \cdot \vec{\phi}(x, c')}}$. The weight in $\vec{\theta}$ that corresponds to $\phi(\vec{w_i}, c_j)$ is trained by MLE.

5.5

$$\sum_x \sum_y p(x,y) \log_2 \frac{p(x,y)}{p(x)p(y)} \geq 0$$
$$P(x) > 0$$
$$P(y) > 0$$
$$P(x,y) \geq 0$$
$$\frac{P(x)p(y)}{P(x,y)} \leq 1$$
$$1 - \frac{P(x)P(y)}{P(x,y)} \geq 0,$$
$$\left(1 - \frac{P(x)P(y)}{P(x,y)}\right) \cdot \log_2 e \geq 0$$
$$\log_2 \frac{P(x,y)}{P(x)p(y)} \geq \left(1 - \frac{p(x)p(y)}{P(x,y)}\right) \cdot \log_2 e \geq 0$$
$$\log_2 \frac{P(x,y)}{P(x)P(y)} \geq 0$$
$$\sum_x \sum_y p(x,y) \cdot \log_2 \frac{P(x,y)}{P(x)f(y)} \geq 0$$

5.6 Distributional word representations make use of words in a context window for the meaning of a target word. We have learned count-based and PPMI vectors for words. The key idea is to measure the relative importance of a context word to a target word. Counts and PPMIs are all statistical measures. t-test can be used to the same end. Given two words u and v, we can assume that their co-occurrence in the same context is by chance (null hypothesis). Then we calculate the probability of the null hypothesis given the co-occurrence statistics of the words in a corpus.

Algorithm 5: Derive SGD training algorithm for the training objective in 5.7

Inputs: $D = \{(x_i, y_i)\}|_{i=1}^{N}$, $D' = \{x'_i\}|_{i=1}^{N'}$, λ, $P(c)(c \in C)$;
Initialisation: $\vec{\theta} \leftarrow 0$; $\alpha \leftarrow \alpha_0$; t $\leftarrow 0$;
repeat
  for $i \in [1, \ldots, N]$ **do**
    $\vec{g} \leftarrow 0$;
    for $c \in C$ **do**
      $P(c \mid x_i) \leftarrow \frac{e^{\vec{\theta}\vec{\phi}(x_i,c)}}{\sum_{c'} e^{\vec{\theta}\vec{\phi}(x_i,c')}}$;
      $\vec{g} \leftarrow \vec{g} + (\vec{\phi}(x_i, c_i) - \vec{\phi}(x_i, c)) \cdot P(c \mid x_i)$;
    $\vec{\theta} \leftarrow \vec{\theta} - \alpha\vec{g}$;
  for $i' \in [1, \ldots, N]$ **do**
    $\vec{g} \leftarrow 0$;
    for $c \in C$ **do**
      $P(c \mid x'_{i'}) \leftarrow \frac{e^{\vec{\theta}\vec{\phi}(x'_{i'},c)}}{\sum_{c'} e^{\vec{\theta}\vec{\phi}(x'_{i'},c')}}$;
      for $c' \in C$ **do**
        $P(c' \mid x'_{i'}) \leftarrow \frac{e^{\vec{\theta}\vec{\phi}(x'_{i'},c')}}{\sum_{c''} e^{\vec{\theta}\vec{\phi}(x'_{i'},c'')}}$;
        $\vec{g} \leftarrow \vec{g} + (\vec{\phi}(x'_{i'}, c'_i) - \vec{\phi}(x'_{i'}, c)) \cdot P(c' \mid x'_{i'}) \cdot \left(\lambda \frac{P(c)}{P(c|x'_{i'})}\right)$;
    $\vec{\theta} \leftarrow \vec{\theta} - \alpha\vec{g}$;
  $t = t + 1$;
until $t = T$;
Ouputs:$\vec{\theta}$

5.7 (a) The value of the hyper-parameter λ can be determined empirically over a set of development data.
(b) No – we need to parameterise $Q(c_i)$ so that the model parameters can be trained through the KL regularisation term.
(c) $Q(x'_j, c_i)$ can be defined as $\tilde{P}(x'_j) = 1/N'$, so that $Q(c_i)$ is calculated as the mathematical expectation of the conditional probability over data. This equation is also a standard marginalisation equation.
(d)loss is $L = \sum_{i=1}^{N} P(c_i \mid x_i) - \lambda \sum_{c \in C} P(c) log_2 \sum_{i=1}^{N'} P(c \mid x'_i)$

$$\frac{\partial L}{\partial \vec{\theta}} = -\sum_{i=1}^{N} \frac{\partial log P(c_i \mid x_i)}{\partial \vec{\theta}} - \lambda \sum_{c \in C} P(c) \cdot \frac{\sum_{i=1}^{N'} \frac{\partial P(c|x'_i)}{\partial \vec{\theta}}}{\sum_{i=1}^{N'} P(c \mid x'_i)}$$
$$= -\sum_{i=1}^{N} \frac{\partial log P(c_i \mid x_i)}{\partial \vec{\theta}} - \lambda \sum_{i=1}^{N'} \sum_{c \in C} \frac{P(c)}{P(c \mid x'_i)} \cdot \frac{\partial P(c \mid x'_i)}{\partial \vec{\theta}}$$

$$= -\sum_{i=1}^{N}\sum_{c\in C}\Big(P(c \mid x_i)(\vec{\phi}(x_i,c_i) - \vec{\phi}(x_i,c))\Big) - \sum_{i'=1}^{N'}\frac{\lambda P(c)}{P(c \mid x'_i)}\sum_{c'\in C}P(c' \mid x'_i)(\vec{\phi}(x'_i,c_i) - \vec{\phi}(x'_i,c'))$$

(e) See Algorithm 5

5.8 The measures are all statistical correlations between words and class labels. The more correlated words may serve as more indicative features. Their relative strengths can be discussed according to the way statistical correlations are calculated, while their effectiveness over a certain dataset can be verified empirically.

Chapter 6 Reference Answers

6.1 Because $P(H \mid o, \Theta)$ is a constant, Θ in this term is fixed to its last value. In contrast, Θ is adjusted(to maximise Q) in $P(o, H \mid \Theta)$.

6.2 (a) Assign each $V_i (i \in [1, ..., N])$ to cluster i mod k.

$$h_{ik} = \begin{cases} 1 \; if \; k = (i \; \% \; k) + 1 \\ 0 \; otherwise \end{cases}$$, where k is the k-th element in $\vec{\theta}$.

(b)

$$\hat{\Theta} = \underset{\Theta}{argmax} \sum_{i=1}^{N} \sum_{k} P(h_i = h \mid o_i, \Theta) \cdot logP(o_i, h \mid \Theta)$$

(c) denote $P(\vec{v}_i, h_i = k \mid \Theta) = \frac{e^{-\|\vec{v}_i - \vec{v}_k\|^2}}{\sum_{k'=1}^{k} e^{-\|\vec{v}_i - \vec{v}_{k'}\|^2}}$, $\Theta = \{\vec{c}_i\} \mid_{i=1}^{k}$. E-step define $h_i k = P(h_i = k \mid \vec{v}_i, \Theta)$ as $\frac{e^{-\|\vec{v}_i - \vec{v}_k\|^2}}{\sum_{k'=1}^{k} e^{-\|\vec{v}_i - \vec{v}_{k'}\|^2}}$.
Q function is $Q(\Theta) = \sum_{i=1}^{N} \sum_{k=1}^{k} h_i k \cdot logP(\vec{v}_i, h_i = k \mid \Theta)$. M-step $\hat{\Theta} = argmax_{\Theta} Q(\Theta)$

6.3 class-2
Iteration-1
we have classes $\hbar_1$ and $\hbar_2$, documents d_1, d_2, ... , d_6 .
Initialise $P(\hbar_1, \Theta) = P(\hbar_2, \Theta) = \frac{1}{2}$, $P(w \mid \hbar, \Theta) = \frac{1}{27}$.
Initialise $P(w_i \mid \hbar_j, \Theta) = \frac{1}{9}$

$$P\left(\hbar \mid d_i, \Theta^t\right) = \frac{P\left(d_i, \hbar \mid \Theta^t\right)}{\sum_{\hbar \in C} P\left(d_i, \hbar \mid \Theta^t\right)} = \frac{P\left(\hbar \mid \Theta^t\right) \prod_{i=1}^{|d_i|} P\left(w_i \mid \hbar, \Theta^t\right)}{\sum_{\hbar \in C} P\left(\hbar \mid \Theta^t\right) \prod_{i=1}^{|d_i|} P\left(w_i \mid \hbar, \Theta^t\right)}$$

$P(\hbar_1 \mid d_1, \Theta) = \frac{\frac{1}{2} \times \frac{1}{9}^4}{\frac{1}{2} \times \frac{1}{9}^4 + \frac{1}{2} \times \frac{1}{9}^4} = \frac{1}{2}$; $P(\hbar_1 \mid d_2, \Theta) = \frac{\frac{1}{2} \times \frac{1}{9}^4}{\frac{1}{2} \times \frac{1}{9}^4 + \frac{1}{2} \times \frac{1}{9}^4} = \frac{1}{2}$;
$P(\hbar_1 \mid d_3, \Theta) = \frac{\frac{1}{2} \times \frac{1}{9}^4}{\frac{1}{2} \times \frac{1}{9}^4 + \frac{1}{2} \times \frac{1}{9}^4} = \frac{1}{2}$; $P(\hbar_1 \mid d_4, \Theta) = \frac{\frac{1}{2} \times \frac{1}{9}^5}{\frac{1}{2} \times \frac{1}{9}^5 + \frac{1}{2} \times \frac{1}{9}^5} = \frac{1}{2}$;
$P(\hbar_1 \mid d_5, \Theta) = \frac{\frac{1}{2} \times \frac{1}{9}^5}{\frac{1}{2} \times \frac{1}{9}^5 + \frac{1}{2} \times \frac{1}{9}^5} = \frac{1}{2}$; $P(\hbar_1 \mid d_6, \Theta) = \frac{\frac{1}{2} \times \frac{1}{9}^5}{\frac{1}{2} \times \frac{1}{9}^5 + \frac{1}{2} \times \frac{1}{9}^5} = \frac{1}{2}$;

$$P(h_2 \mid d_1,\Theta)=\frac{\frac{1}{2}\times\frac{1}{9}^4}{\frac{1}{2}\times\frac{1}{9}^4+\frac{1}{2}\times\frac{1}{9}^4}=\frac{1}{2};\ P(h_2 \mid d_2,\Theta)=\frac{\frac{1}{2}\times\frac{1}{9}^4}{\frac{1}{2}\times\frac{1}{9}^4+\frac{1}{2}\times\frac{1}{9}^4}=\frac{1}{2};$$
$$P(h_2 \mid d_3,\Theta)=\frac{\frac{1}{2}\times\frac{1}{9}^4}{\frac{1}{2}\times\frac{1}{9}^4+\frac{1}{2}\times\frac{1}{9}^4}=\frac{1}{2};\ P(h_2 \mid d_4,\Theta)=\frac{\frac{1}{2}\times\frac{1}{9}^5}{\frac{1}{2}\times\frac{1}{9}^5+\frac{1}{2}\times\frac{1}{9}^5}=\frac{1}{2};$$
$$P(h_2 \mid d_5,\Theta)=\frac{\frac{1}{2}\times\frac{1}{9}^5}{\frac{1}{2}\times\frac{1}{9}^5+\frac{1}{2}\times\frac{1}{9}^5}=\frac{1}{2};\ P(h_2 \mid d_6,\Theta)=\frac{\frac{1}{2}\times\frac{1}{9}^5}{\frac{1}{2}\times\frac{1}{9}^5+\frac{1}{2}\times\frac{1}{9}^5}=\frac{1}{2};$$

$$P(w \mid h,\Theta)=\frac{\sum_{i=1}^{N}P(h \mid d_i,\Theta^t)\sum_{j=1}^{|d_i|}\delta(w_j,w)}{\lambda_h}=\frac{\sum_{i=1}^{N}P(h \mid d_i,\Theta^t)\sum_{j=1}^{|d_i|}\delta(w_j,w)}{\sum_{i=1}^{N}P(h \mid d_i,\Theta^t)\,|d_i|}$$

$$P(Apple \mid h_1,\Theta)=\frac{\frac{1}{2}+\frac{1}{2}+\frac{1}{2}}{\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times5+\frac{1}{2}\times5+\frac{1}{2}\times5}=\frac{1}{9}$$
$$P(released \mid h_1,\Theta)=\frac{\frac{1}{2}+\frac{1}{2}+\frac{1}{2}}{\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times5+\frac{1}{2}\times5+\frac{1}{2}\times5}=\frac{1}{9}$$
$$P(Tom \mid h_1,\Theta)=\frac{\frac{1}{2}+\frac{1}{2}+\frac{1}{2}}{\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times5+\frac{1}{2}\times5+\frac{1}{2}\times5}=\frac{1}{9}$$
$$P(bought \mid h_1,\Theta)=\frac{\frac{1}{2}+\frac{1}{2}+\frac{1}{2}}{\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times5+\frac{1}{2}\times5+\frac{1}{2}\times5}=\frac{1}{9}$$
$$P(one \mid h_1,\Theta)=\frac{\frac{1}{2}+\frac{1}{2}+\frac{1}{2}}{\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times5+\frac{1}{2}\times5+\frac{1}{2}\times5}=\frac{1}{9}$$
$$P(. \mid h_1,\Theta)=\frac{\frac{1}{2}+\frac{1}{2}+\frac{1}{2}}{\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times5+\frac{1}{2}\times5+\frac{1}{2}\times5}=\frac{1}{9}$$
$$P(iPod \mid h_1,\Theta)=\frac{\frac{1}{2}+\frac{1}{2}}{\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times5+\frac{1}{2}\times5+\frac{1}{2}\times5}=\frac{2}{27}$$
$$P(iPhone \mid h_1,\Theta)=\frac{\frac{1}{2}+\frac{1}{2}}{\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times5+\frac{1}{2}\times5+\frac{1}{2}\times5}=\frac{2}{27}$$
$$P(iPad \mid h_1,\Theta)=\frac{\frac{1}{2}+\frac{1}{2}}{\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times5+\frac{1}{2}\times5+\frac{1}{2}\times5}=\frac{2}{27}$$

$$P(Apple \mid h_2,\Theta)=\frac{\frac{1}{2}+\frac{1}{2}+\frac{1}{2}}{\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times5+\frac{1}{2}\times5+\frac{1}{2}\times5}=\frac{1}{9}$$
$$P(released \mid h_2,\Theta)=\frac{\frac{1}{2}+\frac{1}{2}+\frac{1}{2}}{\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times5+\frac{1}{2}\times5+\frac{1}{2}\times5}=\frac{1}{9}$$
$$P(Tom \mid h_2,\Theta)=\frac{\frac{1}{2}+\frac{1}{2}+\frac{1}{2}}{\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times5+\frac{1}{2}\times5+\frac{1}{2}\times5}=\frac{1}{9}$$
$$P(bought \mid h_2,\Theta)=\frac{\frac{1}{2}+\frac{1}{2}+\frac{1}{2}}{\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times5+\frac{1}{2}\times5+\frac{1}{2}\times5}=\frac{1}{9}$$
$$P(one \mid h_2,\Theta)=\frac{\frac{1}{2}+\frac{1}{2}+\frac{1}{2}}{\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times5+\frac{1}{2}\times5+\frac{1}{2}\times5}=\frac{1}{9}$$
$$P(. \mid h_2,\Theta)=\frac{\frac{1}{2}+\frac{1}{2}+\frac{1}{2}}{\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times5+\frac{1}{2}\times5+\frac{1}{2}\times5}=\frac{1}{9}$$
$$P(iPod \mid h_2,\Theta)=\frac{\frac{1}{2}+\frac{1}{2}}{\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times5+\frac{1}{2}\times5+\frac{1}{2}\times5}=\frac{2}{27}$$
$$P(iPhone \mid h_2,\Theta)=\frac{\frac{1}{2}+\frac{1}{2}}{\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times5+\frac{1}{2}\times5+\frac{1}{2}\times5}=\frac{2}{27}$$
$$P(iPad \mid h_2,\Theta)=\frac{\frac{1}{2}+\frac{1}{2}}{\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times5+\frac{1}{2}\times5+\frac{1}{2}\times5}=\frac{2}{27}$$

Iteration-2

$$P(h \mid \Theta)=\frac{\sum_{i=1}^{N}P(h \mid d_i,\Theta^t)}{N}$$

$$P(h_1 \mid \Theta)=\frac{\frac{1}{2}+\frac{1}{2}+\frac{1}{2}+\frac{1}{2}+\frac{1}{2}+\frac{1}{2}}{6}=\frac{1}{2};\ P(h_2 \mid \Theta)=\frac{\frac{1}{2}+\frac{1}{2}+\frac{1}{2}+\frac{1}{2}+\frac{1}{2}+\frac{1}{2}}{6}=\frac{1}{2}$$

$$P(\hbar_1 \mid d_1,\Theta)=\frac{\frac{1}{2}\times\frac{1}{9}^3\times\frac{2}{27}}{\frac{1}{2}\times\frac{1}{9}^3\times\frac{2}{27}+\frac{1}{2}\times\frac{1}{9}^3\times\frac{2}{27}}=\frac{1}{2};\ P(\hbar_1 \mid d_2,\Theta)=\frac{\frac{1}{2}\times\frac{1}{9}^3\times\frac{2}{27}}{\frac{1}{2}\times\frac{1}{9}^3\times\frac{2}{27}+\frac{1}{2}\times\frac{1}{9}^3\times\frac{2}{27}}=\frac{1}{2};$$
$$P(\hbar_1 \mid d_3,\Theta)=\frac{\frac{1}{2}\times\frac{1}{9}^3\times\frac{2}{27}}{\frac{1}{2}\times\frac{1}{9}^3\times\frac{2}{27}+\frac{1}{2}\times\frac{1}{9}^3\times\frac{2}{27}}=\frac{1}{2};\ P(\hbar_1 \mid d_4,\Theta)=\frac{\frac{1}{2}\times\frac{1}{9}^4\times\frac{2}{27}}{\frac{1}{2}\times\frac{1}{9}^4\times\frac{2}{27}+\frac{1}{2}\times\frac{1}{9}^4\times\frac{2}{27}}=\frac{1}{2};$$
$$P(\hbar_1 \mid d_5,\Theta)=\frac{\frac{1}{2}\times\frac{1}{9}^4\times\frac{2}{27}}{\frac{1}{2}\times\frac{1}{9}^4\times\frac{2}{27}+\frac{1}{2}\times\frac{1}{9}^4\times\frac{2}{27}}=\frac{1}{2};\ P(\hbar_1 \mid d_6,\Theta)=\frac{\frac{1}{2}\times\frac{1}{9}^4\times\frac{2}{27}}{\frac{1}{2}\times\frac{1}{9}^4\times\frac{2}{27}+\frac{1}{2}\times\frac{1}{9}^4\times\frac{2}{27}}=\frac{1}{2};$$

$$P(\hbar_2 \mid d_1,\Theta)=\frac{\frac{1}{2}\times\frac{1}{9}^3\times\frac{2}{27}}{\frac{1}{2}\times\frac{1}{9}^3\times\frac{2}{27}+\frac{1}{2}\times\frac{1}{9}^3\times\frac{2}{27}}=\frac{1}{2};\ P(\hbar_2 \mid d_2,\Theta)=\frac{\frac{1}{2}\times\frac{1}{9}^3\times\frac{2}{27}}{\frac{1}{2}\times\frac{1}{9}^3\times\frac{2}{27}+\frac{1}{2}\times\frac{1}{9}^3\times\frac{2}{27}}=\frac{1}{2};$$
$$P(\hbar_2 \mid d_3,\Theta)=\frac{\frac{1}{2}\times\frac{1}{9}^3\times\frac{2}{27}}{\frac{1}{2}\times\frac{1}{9}^3\times\frac{2}{27}+\frac{1}{2}\times\frac{1}{9}^3\times\frac{2}{27}}=\frac{1}{2};\ P(\hbar_2 \mid d_4,\Theta)=\frac{\frac{1}{2}\times\frac{1}{9}^4\times\frac{2}{27}}{\frac{1}{2}\times\frac{1}{9}^4\times\frac{2}{27}+\frac{1}{2}\times\frac{1}{9}^4\times\frac{2}{27}}=\frac{1}{2};$$
$$P(\hbar_2 \mid d_5,\Theta)=\frac{\frac{1}{2}\times\frac{1}{9}^4\times\frac{2}{27}}{\frac{1}{2}\times\frac{1}{9}^4\times\frac{2}{27}+\frac{1}{2}\times\frac{1}{9}^4\times\frac{2}{27}}=\frac{1}{2};\ P(\hbar_2 \mid d_6,\Theta)=\frac{\frac{1}{2}\times\frac{1}{9}^4\times\frac{2}{27}}{\frac{1}{2}\times\frac{1}{9}^4\times\frac{2}{27}+\frac{1}{2}\times\frac{1}{9}^4\times\frac{2}{27}}=\frac{1}{2};$$

$$P(w \mid h,\Theta)=\frac{\sum_{i=1}^{N}P(h \mid d_i,\Theta^t)\sum_{j=1}^{|d_i|}\delta(w_j,w)}{\lambda_h}=\frac{\sum_{i=1}^{N}P(h \mid d_i,\Theta^t)\sum_{j=1}^{|d_i|}\delta(w_j,w)}{\sum_{i=1}^{N}P(h \mid d_i,\Theta^t)|d_i|}$$

$$P(Apple \mid h_1,\Theta)=\frac{\frac{1}{2}+\frac{1}{2}+\frac{1}{2}}{\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times5+\frac{1}{2}\times5+\frac{1}{2}\times5}=\frac{1}{9}$$
$$P(released \mid h_1,\Theta)=\frac{\frac{1}{2}+\frac{1}{2}+\frac{1}{2}}{\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times5+\frac{1}{2}\times5+\frac{1}{2}\times5}=\frac{1}{9}$$
$$P(Tom \mid h_1,\Theta)=\frac{\frac{1}{2}+\frac{1}{2}+\frac{1}{2}}{\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times5+\frac{1}{2}\times5+\frac{1}{2}\times5}=\frac{1}{9}$$
$$P(bought \mid h_1,\Theta)=\frac{\frac{1}{2}+\frac{1}{2}+\frac{1}{2}}{\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times5+\frac{1}{2}\times5+\frac{1}{2}\times5}=\frac{1}{9}$$
$$P(one \mid h_1,\Theta)=\frac{\frac{1}{2}+\frac{1}{2}+\frac{1}{2}}{\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times5+\frac{1}{2}\times5+\frac{1}{2}\times5}=\frac{1}{9}$$
$$P(. \mid h_1,\Theta)=\frac{\frac{1}{2}+\frac{1}{2}+\frac{1}{2}}{\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times5+\frac{1}{2}\times5+\frac{1}{2}\times5}=\frac{1}{9}$$
$$P(iPod \mid h_1,\Theta)=\frac{\frac{1}{2}+\frac{1}{2}}{\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times5+\frac{1}{2}\times5+\frac{1}{2}\times5}=\frac{2}{27}$$
$$P(iPhone \mid h_1,\Theta)=\frac{\frac{1}{2}+\frac{1}{2}}{\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times5+\frac{1}{2}\times5+\frac{1}{2}\times5}=\frac{2}{27}$$
$$P(iPad \mid h_1,\Theta)=\frac{\frac{1}{2}+\frac{1}{2}}{\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times5+\frac{1}{2}\times5+\frac{1}{2}\times5}=\frac{2}{27}$$

$$P(Apple \mid h_2,\Theta)=\frac{\frac{1}{2}+\frac{1}{2}+\frac{1}{2}}{\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times5+\frac{1}{2}\times5+\frac{1}{2}\times5}=\frac{1}{9}$$
$$P(released \mid h_2,\Theta)=\frac{\frac{1}{2}+\frac{1}{2}+\frac{1}{2}}{\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times5+\frac{1}{2}\times5+\frac{1}{2}\times5}=\frac{1}{9}$$
$$P(Tom \mid h_2,\Theta)=\frac{\frac{1}{2}+\frac{1}{2}+\frac{1}{2}}{\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times5+\frac{1}{2}\times5+\frac{1}{2}\times5}=\frac{1}{9}$$
$$P(bought \mid h_2,\Theta)=\frac{\frac{1}{2}+\frac{1}{2}+\frac{1}{2}}{\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times5+\frac{1}{2}\times5+\frac{1}{2}\times5}=\frac{1}{9}$$
$$P(one \mid h_2,\Theta)=\frac{\frac{1}{2}+\frac{1}{2}+\frac{1}{2}}{\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times5+\frac{1}{2}\times5+\frac{1}{2}\times5}=\frac{1}{9}$$
$$P(. \mid h_2,\Theta)=\frac{\frac{1}{2}+\frac{1}{2}+\frac{1}{2}}{\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times5+\frac{1}{2}\times5+\frac{1}{2}\times5}=\frac{1}{9}$$
$$P(iPod \mid h_2,\Theta)=\frac{\frac{1}{2}+\frac{1}{2}}{\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times5+\frac{1}{2}\times5+\frac{1}{2}\times5}=\frac{2}{27}$$
$$P(iPhone \mid h_2,\Theta)=\frac{\frac{1}{2}+\frac{1}{2}}{\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times5+\frac{1}{2}\times5+\frac{1}{2}\times5}=\frac{2}{27}$$

$P(iPad \mid h_2,\Theta) = \frac{\frac{1}{2}+\frac{1}{2}}{\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times4+\frac{1}{2}\times5+\frac{1}{2}\times5+\frac{1}{2}\times5} = \frac{2}{27}$

Coverage.

3 classes

Iteration-1

we have classes $\hbar_1$, $\hbar_2$ and $\hbar_3$, documents d_1, d_2, ... , d_6 .

Initialise $P(\hbar_1,\Theta) = P(\hbar_2,\Theta) = P(\hbar_2,\Theta) = \frac{1}{3}$, $P(w \mid \hbar,\Theta) = \frac{1}{27}$.

Initialise $P(w_i \mid \hbar_j,\Theta) = \frac{1}{9}$

$$P\left(\hbar \mid d_i,\Theta^t\right) = \frac{P\left(d_i,\hbar \mid \Theta^t\right)}{\sum_{\hbar\in C} P\left(d_i,\hbar \mid \Theta^t\right)} = \frac{P\left(\hbar \mid \Theta^t\right)\prod_{i=1}^{|d_i|} P\left(w_i \mid \hbar,\Theta^t\right)}{\sum_{\hbar\in C} P\left(\hbar \mid \Theta^t\right)\prod_{i=1}^{|d_i|} P\left(w_i \mid \hbar,\Theta^t\right)}$$

$P(\hbar_1 \mid d_1,\Theta) = \frac{\frac{1}{3}\times\frac{1}{9}^4}{\frac{1}{3}\times\frac{1}{9}^4+\frac{1}{3}\times\frac{1}{9}^4+\frac{1}{3}\times\frac{1}{9}^4} = \frac{1}{3}$; $P(\hbar_1 \mid d_2,\Theta) = \frac{\frac{1}{3}\times\frac{1}{9}^4}{\frac{1}{3}\times\frac{1}{9}^4+\frac{1}{3}\times\frac{1}{9}^4+\frac{1}{3}\times\frac{1}{9}^4} = \frac{1}{3}$;

$P(\hbar_1 \mid d_3,\Theta) = \frac{\frac{1}{3}\times\frac{1}{9}^4}{\frac{1}{3}\times\frac{1}{9}^4+\frac{1}{3}\times\frac{1}{9}^4+\frac{1}{3}\times\frac{1}{9}^4} = \frac{1}{3}$; $P(\hbar_1 \mid d_4,\Theta) = \frac{\frac{1}{3}\times\frac{1}{9}^5}{\frac{1}{3}\times\frac{1}{9}^5+\frac{1}{3}\times\frac{1}{9}^5+\frac{1}{3}\times\frac{1}{9}^5} = \frac{1}{3}$;

$P(\hbar_1 \mid d_5,\Theta) = \frac{\frac{1}{3}\times\frac{1}{9}^5}{\frac{1}{3}\times\frac{1}{9}^5+\frac{1}{3}\times\frac{1}{9}^5+\frac{1}{3}\times\frac{1}{9}^5} = \frac{1}{3}$; $P(\hbar_1 \mid d_6,\Theta) = \frac{\frac{1}{3}\times\frac{1}{9}^5}{\frac{1}{3}\times\frac{1}{9}^5+\frac{1}{3}\times\frac{1}{9}^5+\frac{1}{3}\times\frac{1}{9}^5} = \frac{1}{3}$;

$P(\hbar_2 \mid d_1,\Theta) = \frac{\frac{1}{3}\times\frac{1}{9}^4}{\frac{1}{3}\times\frac{1}{9}^4+\frac{1}{3}\times\frac{1}{9}^4+\frac{1}{3}\times\frac{1}{9}^4} = \frac{1}{3}$; $P(\hbar_2 \mid d_2,\Theta) = \frac{\frac{1}{3}\times\frac{1}{9}^4}{\frac{1}{3}\times\frac{1}{9}^4+\frac{1}{3}\times\frac{1}{9}^4+\frac{1}{3}\times\frac{1}{9}^4} = \frac{1}{3}$;

$P(\hbar_2 \mid d_3,\Theta) = \frac{\frac{1}{3}\times\frac{1}{9}^4}{\frac{1}{3}\times\frac{1}{9}^4+\frac{1}{3}\times\frac{1}{9}^4+\frac{1}{3}\times\frac{1}{9}^4} = \frac{1}{3}$; $P(\hbar_2 \mid d_4,\Theta) = \frac{\frac{1}{3}\times\frac{1}{9}^5}{\frac{1}{3}\times\frac{1}{9}^5+\frac{1}{3}\times\frac{1}{9}^5+\frac{1}{3}\times\frac{1}{9}^5} = \frac{1}{3}$;

$P(\hbar_2 \mid d_5,\Theta) = \frac{\frac{1}{3}\times\frac{1}{9}^5}{\frac{1}{3}\times\frac{1}{9}^5+\frac{1}{3}\times\frac{1}{9}^5+\frac{1}{3}\times\frac{1}{9}^5} = \frac{1}{3}$; $P(\hbar_2 \mid d_6,\Theta) = \frac{\frac{1}{3}\times\frac{1}{9}^5}{\frac{1}{3}\times\frac{1}{9}^5+\frac{1}{3}\times\frac{1}{9}^5+\frac{1}{3}\times\frac{1}{9}^5} = \frac{1}{3}$;

$P(\hbar_3 \mid d_1,\Theta) = \frac{\frac{1}{3}\times\frac{1}{9}^4}{\frac{1}{3}\times\frac{1}{9}^4+\frac{1}{3}\times\frac{1}{9}^4+\frac{1}{3}\times\frac{1}{9}^4} = \frac{1}{3}$; $P(\hbar_3 \mid d_2,\Theta) = \frac{\frac{1}{3}\times\frac{1}{9}^4}{\frac{1}{3}\times\frac{1}{9}^4+\frac{1}{3}\times\frac{1}{9}^4+\frac{1}{3}\times\frac{1}{9}^4} = \frac{1}{3}$;

$P(\hbar_3 \mid d_3,\Theta) = \frac{\frac{1}{3}\times\frac{1}{9}^4}{\frac{1}{3}\times\frac{1}{9}^4+\frac{1}{3}\times\frac{1}{9}^4+\frac{1}{3}\times\frac{1}{9}^4} = \frac{1}{3}$; $P(\hbar_3 \mid d_4,\Theta) = \frac{\frac{1}{3}\times\frac{1}{9}^5}{\frac{1}{3}\times\frac{1}{9}^5+\frac{1}{3}\times\frac{1}{9}^5+\frac{1}{3}\times\frac{1}{9}^5} = \frac{1}{3}$;

$P(\hbar_3 \mid d_5,\Theta) = \frac{\frac{1}{3}\times\frac{1}{9}^5}{\frac{1}{3}\times\frac{1}{9}^5+\frac{1}{3}\times\frac{1}{9}^5+\frac{1}{3}\times\frac{1}{9}^5} = \frac{1}{3}$; $P(\hbar_3 \mid d_6,\Theta) = \frac{\frac{1}{3}\times\frac{1}{9}^5}{\frac{1}{3}\times\frac{1}{9}^5+\frac{1}{3}\times\frac{1}{9}^5+\frac{1}{3}\times\frac{1}{9}^5} = \frac{1}{3}$;

$P(Apple \mid h_1,\Theta) = \frac{\frac{1}{3}+\frac{1}{3}+\frac{1}{3}}{\frac{1}{3}\times4+\frac{1}{3}\times4+\frac{1}{3}\times4+\frac{1}{3}\times5+\frac{1}{3}\times5+\frac{1}{3}\times5} = \frac{1}{9}$

$P(released \mid h_1,\Theta) = \frac{\frac{1}{3}+\frac{1}{3}+\frac{1}{3}}{\frac{1}{3}\times4+\frac{1}{3}\times4+\frac{1}{3}\times4+\frac{1}{3}\times5+\frac{1}{3}\times5+\frac{1}{3}\times5} = \frac{1}{9}$

$P(Tom \mid h_1,\Theta) = \frac{\frac{1}{3}+\frac{1}{3}+\frac{1}{3}}{\frac{1}{3}\times4+\frac{1}{3}\times4+\frac{1}{3}\times4+\frac{1}{3}\times5+\frac{1}{3}\times5+\frac{1}{3}\times5} = \frac{1}{9}$

$P(bought \mid h_1,\Theta) = \frac{\frac{1}{3}+\frac{1}{3}+\frac{1}{3}}{\frac{1}{3}\times4+\frac{1}{3}\times4+\frac{1}{3}\times4+\frac{1}{3}\times5+\frac{1}{3}\times5+\frac{1}{3}\times5} = \frac{1}{9}$

$P(one \mid h_1,\Theta) = \frac{\frac{1}{3}+\frac{1}{3}+\frac{1}{3}}{\frac{1}{3}\times4+\frac{1}{3}\times4+\frac{1}{3}\times4+\frac{1}{3}\times5+\frac{1}{3}\times5+\frac{1}{3}\times5} = \frac{1}{9}$

$P(. \mid h_1,\Theta) = \frac{\frac{1}{3}+\frac{1}{3}+\frac{1}{3}}{\frac{1}{3}\times4+\frac{1}{3}\times4+\frac{1}{3}\times4+\frac{1}{3}\times5+\frac{1}{3}\times5+\frac{1}{3}\times5} = \frac{1}{9}$

$P(iPod \mid h_1,\Theta) = \frac{\frac{1}{3}+\frac{1}{3}}{\frac{1}{3}\times4+\frac{1}{3}\times4+\frac{1}{3}\times4+\frac{1}{3}\times5+\frac{1}{3}\times5+\frac{1}{3}\times5} = \frac{2}{27}$

$P(iPhone \mid h_1,\Theta) = \frac{\frac{1}{3}+\frac{1}{3}}{\frac{1}{3}\times4+\frac{1}{3}\times4+\frac{1}{3}\times4+\frac{1}{3}\times5+\frac{1}{3}\times5+\frac{1}{3}\times5} = \frac{2}{27}$

$P(iPad \mid h_1,\Theta) = \frac{\frac{1}{3}+\frac{1}{3}}{\frac{1}{3}\times4+\frac{1}{3}\times4+\frac{1}{3}\times4+\frac{1}{3}\times5+\frac{1}{3}\times5+\frac{1}{3}\times5} = \frac{2}{27}$

$$P(Apple \mid h_2, \Theta) = \frac{\frac{1}{3}+\frac{1}{3}+\frac{1}{3}}{\frac{1}{3}\times 4+\frac{1}{3}\times 4+\frac{1}{3}\times 4+\frac{1}{3}\times 5+\frac{1}{3}\times 5+\frac{1}{3}\times 5} = \frac{1}{9}$$
$$P(released \mid h_2, \Theta) = \frac{\frac{1}{3}+\frac{1}{3}+\frac{1}{3}}{\frac{1}{3}\times 4+\frac{1}{3}\times 4+\frac{1}{3}\times 4+\frac{1}{3}\times 5+\frac{1}{3}\times 5+\frac{1}{3}\times 5} = \frac{1}{9}$$
$$P(Tom \mid h_2, \Theta) = \frac{\frac{1}{3}+\frac{1}{3}+\frac{1}{3}}{\frac{1}{3}\times 4+\frac{1}{3}\times 4+\frac{1}{3}\times 4+\frac{1}{3}\times 5+\frac{1}{3}\times 5+\frac{1}{3}\times 5} = \frac{1}{9}$$
$$P(bought \mid h_2, \Theta) = \frac{\frac{1}{3}+\frac{1}{3}+\frac{1}{3}}{\frac{1}{3}\times 4+\frac{1}{3}\times 4+\frac{1}{3}\times 4+\frac{1}{3}\times 5+\frac{1}{3}\times 5+\frac{1}{3}\times 5} = \frac{1}{9}$$
$$P(one \mid h_2, \Theta) = \frac{\frac{1}{3}+\frac{1}{3}+\frac{1}{3}}{\frac{1}{3}\times 4+\frac{1}{3}\times 4+\frac{1}{3}\times 4+\frac{1}{3}\times 5+\frac{1}{3}\times 5+\frac{1}{3}\times 5} = \frac{1}{9}$$
$$P(. \mid h_2, \Theta) = \frac{\frac{1}{3}+\frac{1}{3}+\frac{1}{3}}{\frac{1}{3}\times 4+\frac{1}{3}\times 4+\frac{1}{3}\times 4+\frac{1}{3}\times 5+\frac{1}{3}\times 5+\frac{1}{3}\times 5} = \frac{1}{9}$$
$$P(iPod \mid h_2, \Theta) = \frac{\frac{1}{3}+\frac{1}{3}}{\frac{1}{3}\times 4+\frac{1}{3}\times 4+\frac{1}{3}\times 4+\frac{1}{3}\times 5+\frac{1}{3}\times 5+\frac{1}{3}\times 5} = \frac{2}{27}$$
$$P(iPhone \mid h_2, \Theta) = \frac{\frac{1}{3}+\frac{1}{3}}{\frac{1}{3}\times 4+\frac{1}{3}\times 4+\frac{1}{3}\times 4+\frac{1}{3}\times 5+\frac{1}{3}\times 5+\frac{1}{3}\times 5} = \frac{2}{27}$$
$$P(iPad \mid h_2, \Theta) = \frac{\frac{1}{3}+\frac{1}{3}}{\frac{1}{3}\times 4+\frac{1}{3}\times 4+\frac{1}{3}\times 4+\frac{1}{3}\times 5+\frac{1}{3}\times 5+\frac{1}{3}\times 5} = \frac{2}{27}$$

$$P(Apple \mid h_3, \Theta) = \frac{\frac{1}{3}+\frac{1}{3}+\frac{1}{3}}{\frac{1}{3}\times 4+\frac{1}{3}\times 4+\frac{1}{3}\times 4+\frac{1}{3}\times 5+\frac{1}{3}\times 5+\frac{1}{3}\times 5} = \frac{1}{9}$$
$$P(released \mid h_3, \Theta) = \frac{\frac{1}{3}+\frac{1}{3}+\frac{1}{3}}{\frac{1}{3}\times 4+\frac{1}{3}\times 4+\frac{1}{3}\times 4+\frac{1}{3}\times 5+\frac{1}{3}\times 5+\frac{1}{3}\times 5} = \frac{1}{9}$$
$$P(Tom \mid h_3, \Theta) = \frac{\frac{1}{3}+\frac{1}{3}+\frac{1}{3}}{\frac{1}{3}\times 4+\frac{1}{3}\times 4+\frac{1}{3}\times 4+\frac{1}{3}\times 5+\frac{1}{3}\times 5+\frac{1}{3}\times 5} = \frac{1}{9}$$
$$P(bought \mid h_3, \Theta) = \frac{\frac{1}{3}+\frac{1}{3}+\frac{1}{3}}{\frac{1}{3}\times 4+\frac{1}{3}\times 4+\frac{1}{3}\times 4+\frac{1}{3}\times 5+\frac{1}{3}\times 5+\frac{1}{3}\times 5} = \frac{1}{9}$$
$$P(one \mid h_3, \Theta) = \frac{\frac{1}{3}+\frac{1}{3}+\frac{1}{3}}{\frac{1}{3}\times 4+\frac{1}{3}\times 4+\frac{1}{3}\times 4+\frac{1}{3}\times 5+\frac{1}{3}\times 5+\frac{1}{3}\times 5} = \frac{1}{9}$$
$$P(. \mid h_3, \Theta) = \frac{\frac{1}{3}+\frac{1}{3}+\frac{1}{3}}{\frac{1}{3}\times 4+\frac{1}{3}\times 4+\frac{1}{3}\times 4+\frac{1}{3}\times 5+\frac{1}{3}\times 5+\frac{1}{3}\times 5} = \frac{1}{9}$$
$$P(iPod \mid h_3, \Theta) = \frac{\frac{1}{3}+\frac{1}{3}}{\frac{1}{3}\times 4+\frac{1}{3}\times 4+\frac{1}{3}\times 4+\frac{1}{3}\times 5+\frac{1}{3}\times 5+\frac{1}{3}\times 5} = \frac{2}{27}$$
$$P(iPhone \mid h_3, \Theta) = \frac{\frac{1}{3}+\frac{1}{3}}{\frac{1}{3}\times 4+\frac{1}{3}\times 4+\frac{1}{3}\times 4+\frac{1}{3}\times 5+\frac{1}{3}\times 5+\frac{1}{3}\times 5} = \frac{2}{27}$$
$$P(iPad \mid h_3, \Theta) = \frac{\frac{1}{3}+\frac{1}{3}}{\frac{1}{3}\times 4+\frac{1}{3}\times 4+\frac{1}{3}\times 4+\frac{1}{3}\times 5+\frac{1}{3}\times 5+\frac{1}{3}\times 5} = \frac{2}{27}$$

Iteration-2

Repeat, coverage.

When the initialization parameters are equal, both 2-class and 3-class converge after two iterations.

6.4 (a)We can follow the EM algorithm and derive the training process for semi-supervised Naive Bayes. The idea is similar to Eq 6.14. In particular, the labeled data instances can be directly used for MLE optimization, while the unlabeled instances can contribute to the overall training loss function through a Q function. The derivation can be

$$Q(\Theta,\lambda) = \sum_{i=1}^{N} \log P(d_i, c_i \mid \Theta) + \sum_{j=N+1}^{N+M} \sum_{c\in C} P\left(d_i, \Theta^t\right) \log P(d_i, c \mid \Theta)$$

$$\begin{aligned}
\wedge(\Theta,\lambda) &= Q\left(\Theta,\Theta^t\right) - \lambda_0 \left(\sum_{c\in C} P(c \mid \Theta) - 1\right) - \sum_{c\in C} \lambda_c \left(\sum_{\omega\in V} P(\omega \mid c, \Theta\right) - 1\Bigg) \\
&= \sum_{i=1}^{N} \log P(d_i, c_i \mid \Theta) + \sum_{j=N+1}^{N+M} \sum_{c\in C} P\left(c \mid d_i, \Theta^t\right) \left(\log P(d_i, c \mid \theta)\right) \\
&\quad - \lambda_0 \left(\sum_{c\in C} P(c \mid \Theta) - 1\right) - \sum_{c\in C} \lambda_c \left(\sum_{w\in V} P(\omega \mid c, \Theta)\right) - 1 \\
&= \sum_{i=1}^{N} \left(\log P(c_i \mid \Theta) + \sum_{j=1}^{|d_i|} \log P(\omega_j^i \mid c_i, \Theta)\right) \\
&\quad + \sum_{i=N+1}^{N+M} \sum_{c\in C} P\left(c \mid d_i, \Theta^t\right) \left(\log P(c \mid \Theta) + \sum_{j=1}^{|d_i|} \log P\left(\omega_j^i \mid c, \Theta\right)\right) \\
&\quad - \lambda_0 \left(\sum_{c\in C} P(c \mid \Theta) - 1\right) - \sum_{c\in C} \lambda_c \left(\sum_{\omega} P(\omega \mid c, \Theta) - 1\right)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial \wedge(\theta,\lambda)}{\partial P(c \mid \Theta)} &= \frac{\sum_{i=1}^{N} \delta(c_i, c)}{P(c \mid \Theta)} + \sum_{i=N+1}^{N+M} \frac{P(c \mid d_i, \Theta^t)}{P(c \mid \Theta)} - \lambda_0 \\
&= \frac{\sum_{i=1}^{N} \delta(c_i, c) + \sum_{i=N+1}^{N+M} P(c \mid d_i, \Theta^t)}{P(c \mid \Theta)} - \lambda_0
\end{aligned}$$

Letting $\dfrac{\partial \wedge(\Theta,\lambda)}{\partial P(c \mid \Theta)} = 0, P(c \mid \Theta) = \dfrac{\sum_{i=1}^{N} \delta(c_i, c) + \sum_{i=N+1}^{N+M} P(c \mid d_i, \Theta^t)}{\lambda_0}$

Under the constraints that $\sum_{c\in C} P(c \mid \theta) = 1$

$$\sum_{c\in C} \frac{\sum_{i=1}^{N} \delta(c_i, c) + \sum_{i=N+1}^{N+M} P(c \mid d_i, \Theta^t)}{\lambda_0} = \frac{N+M}{\lambda_0} = 1$$

$$\lambda = N + M$$

Thus $P(c \mid \Theta) = \dfrac{\sum_{i=1}^{N} \delta(c_i, c) + \sum_{i=N+1}^{N+M} P(c \mid d_i, \Theta^t)}{N+M}$

As can be seen from the equation above, the value of the model parameter $P(c|\Theta)$ is a combination of the objective in Eq 6.11 and the objective in Eq 6.14. For labeled training instances, we take a hard count 1 for its gold-standard class; for unlabeled data instances, we use Θ^t to estimate the count. The total count is the total number of data instances, $N+M$.

The above equation can be inserted back to Algorithm 6.2. Note that we can also use the labeled instances to initialize values of Θ before EM training. The other

parameter type $P(w|c,\Theta)$ can be derived similarly.
(b) The value of λ here controls the contribution of the unlabeled instances. Following the derivation of (a), the parameter estimation for (c) is

$$\frac{\sum_{i=1}^{N}\delta(c_i,c)+\sum_{i=N+1}^{N+M}\lambda P(c\mid d_i,\Theta^t)}{N+M}$$

This regularisation term is different from L2 norm in that it makes use of a set of unlabeled data for better estimating parameters.

6.5 IBM model 1 source code: https://github.com/shawa/IBM-Model-1. You can change the data to different language.

6.6 proof by induction
(1) $|x|=1 \quad \sum_{a_1=0}^{|Y|}\prod_{i=1}^{1}P(x_i\mid y_{a_i})=\sum_{a_i=0}^{|\mathrm{Y}|}P(x_1\mid y_{a_1})=\prod_{i=1}^{1}\sum_{j=0}^{|Y|}P(x_i\mid y_j)$
(2)

$$\begin{aligned}
&\text{if } \sum_{a_1=0}^{|Y|}\sum_{a_2=0}^{|Y|}\ldots\sum_{a_k=0}^{|Y|}\prod_{i=1}^{k}P(x_i\mid y_{a_i})=\prod_{i=1}^{k}\sum_{j=0}^{|\mathrm{Y}|}P(x_i\mid y_j)\\
&\text{then } \sum_{a_1=0}^{|Y|}\sum_{a_2=0}^{|Y|}\ldots\sum_{a_k=0}^{|Y|}\prod_{i=1}^{k+1}P(x_i\mid y_{a_i})\\
&=\sum_{a_1=0}^{|Y|}\sum_{a_2=0}^{|Y|}\ldots\sum_{a_k=0}^{|Y|}\prod_{i=1}^{k}P(x_i\mid y_{a_i})\cdot P(x_{k+1}\mid y_{a_{k+1}})\\
&=\sum_{a_{k+1}=0}^{|Y|}\left(\sum_{a_1=0}^{|Y|}\sum_{a_2=0}^{|Y|}\ldots\sum_{a_k=0}^{|Y|}P(x_i\mid y_{a_i})\cdot P(x_{k+1}\mid y_{a_{k+1}})\right)\\
&=\sum_{a_{k+1}=0}^{|Y|}\left(\prod_{i=1}^{k}\sum_{j=0}^{|Y|}P(x_i\mid y_i)\right)\cdot P(x_{k+1}\mid y_{a_i})\\
&=\left(\prod_{i=1}^{k}\sum_{j=0}^{|Y|}P(x_i\mid y_i)\right)\cdot\left(\sum_{a_{k+1}=0}^{|Y|}\cdot P(x_{k+1}\mid y_{a_{k+1}})\right)\\
&=\prod_{i=1}^{k}\sum_{j=0}^{|Y|}P(x_{k+1}\mid y_{a_{k+1}})
\end{aligned}$$

6.7 By induction
(a)

$$\sum_{A}\prod_{i=1}^{1}\frac{P(x_i\mid y_i)}{\sum_{j=0}^{|Y|}P(x_i\mid y_j)}\cdot\sum_{k=1}^{1}\delta(x,x_k)\,\delta(y,y_{a_k})$$

$$
\begin{aligned}
&= \sum_{a_1=0}^{|Y|} \frac{P(x_1 \mid y_{a_1})}{\sum_{j=0}^{|Y|} P(x_1 \mid y_j)} \cdot \delta(x, x_1) \cdot \delta(y, y_{a_1}) \\
&= \sum_{i=1}^{|X|} \sum_{a_i=0}^{|Y|} \frac{P(x_i \mid y_{a_i})}{\sum_{j=0}^{|Y|} P(x_i \mid y_j)} \cdot \delta(x, x_i) \cdot \delta(y, y_{a_i}) \\
&= \frac{P(x \mid y)}{\sum_{j=0}^{|Y|} p(x \mid y_j)} \sum_{i=1}^{|X|} \delta(x, x_i) \cdot \sum_{j=0}^{|Y|} \delta(y, y_j)
\end{aligned}
$$

(b)

$$
\begin{aligned}
&\text{If} \sum_{A} \prod_{i=1}^{k} \frac{P(x_i \mid y_{a_i})}{\sum_{j=0}^{|Y|} P(x_i, y_j)} \cdot \sum_{k=1}^{k} \delta(x, x_k) \cdot \delta(y, y_{a_k}) \\
&\sum_{A} \prod_{i=1}^{k+1} \frac{P(x_i \mid y_{a_i})}{\sum_{j=0}^{|Y|} P(x_i, y_j)} \cdot \sum_{k=1}^{k} \delta(x, x_{k+1}) \cdot \delta(y, y_{a_k}) \\
&= \sum_{a_{k+1}=0}^{|Y|} \left(\sum_{a_1=0}^{|Y|} \sum_{a_2=0}^{|Y|} \cdots \sum_{a_k=0}^{|Y|} \prod_{i=1}^{k} \frac{P(x_i \mid y_{a_i})}{\sum_{j=0}^{|Y|} P(x_i \mid y_i)} \right) \cdot \frac{P\left(x_{k+1} \mid y_{a_{k+1}}\right)}{\sum_{j=0}^{|Y|} P(x_{k+1} \mid y_j)} \\
&\cdot \left(\sum_{k=1}^{k} \delta(x, x_k) \cdot \delta(y, y_{a_k}) \right) \cdot \delta(x, x_{k+1}) \cdot \delta\left(y, y_{a_{k+1}}\right) \\
&= \sum_{a_{k+1}=0}^{|Y|} \left(\frac{P(x \mid y)}{\sum_{j=0}^{|Y|} P(x \mid y_i)} \cdot \sum_{k=1}^{k} \delta(x, x_{k+1}) \sum_{j=0}^{|Y|} \delta(y, y_j) \right) \\
&\cdot \frac{P\left(x_{k+1} \mid y_{a_{k+1}}\right)}{\sum_{j=0}^{|Y|} P(x_{k+1} \mid y_j)} \cdot \delta(x, x_{k+1}) \cdot \delta\left(y, y_{a_{k+1}}\right) \\
&= \frac{P(x \mid y)}{\sum_{j=0}^{|Y|} P(x \mid y_i)} \cdot \sum_{k=1}^{k} \delta(x, x_{k+1}) \sum_{j=0}^{|Y|} \delta(y, y_j) \cdot \sum_{a_{k+1}=0}^{|Y|} \frac{P\left(x_{k+1} \mid y_{a_{k+1}}\right)}{\sum_{j=0}^{|Y|} P(x_{k+1} \mid y_j)} \\
&\cdot \delta(x, x_{k+1}) \cdot \delta\left(y, y_{a_{k+1}}\right) \\
&= \frac{P(x \mid y)}{\sum_{j=0}^{|Y|} P(x \mid y_i)} \cdot \sum_{k=1}^{k+1} \delta(x, x_k) \sum_{j=0}^{|Y|} \delta(y, y_j)
\end{aligned}
$$

6.8

$d \backslash w$	*WorldCup*	*host*	*Russia*	*bid*	*economy*	*recover*	*continue*	*boost*	*growing*	*oil*	*dependence*
d_1	1	1	1	0	0	0	0	0	0	0	0
d_2	1	1	0	0	1	0	0	1	0	0	0
d_3	1	1	0	1	0	0	0	0	0	0	0
d_4	0	1	0	0	1	0	0	1	0	0	0
d_5	0	1	0	0	1	1	1	0	0	0	0
d_6	0	1	0	0	0	0	0	0	0	1	1

Table 1.3: Table of documents for latent topic analysis

See Algorithm 6

6.9 *Similarities.* Both are iterative algorithms over unlabeled data.

Differences. (1) Hard EM is more general in that it deals with hidden variables. Self-training can be viewed as leveraging unlabeled data by making the output labels hidden variables. However, the hidden variables are constrained to the unseen outputs only.

(2) EM is fully unsupervised but self-training is semi-supervised. EM can be made more similar to self-training by adding labeled data also, as for Exercise 6.4.

(3) Hard EM is a probabilistic model that iteratively estimates hidden variables and model parameters by maximizing the joint data likelihood. In contrast, self-

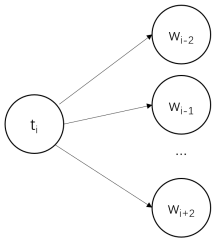
training is not necessarily a probabilistic model trained using MLE.

Algorithm 6: PLSA in 6.8

Inputs: $h \in [1,2]$,$d \in [1, \ldots, 6]$,$w \in [1, \ldots, 11]$, Phd, Pwh;
Initialisation: $h \in [1,2]$,$d \in [1, \ldots, 6]$,$w \in [1, \ldots, 11]$,$total - c \leftarrow$
20 ,$c[1][1] = 1,\ c[2][1] = 1,\ c[3][1] = 1,$
$c[1][2] = 1,\ c[2][2] = 1,\ c[5][2] = 1,$
$c[1][3] = 1,\ c[2][3] = 1,\ c[4][3] = 1,$
$c[2][4] = 1,\ c[5][4] = 1,\ c[9][4] = 1,\ c[10][4] = 1,$
$c[2][5] = 1,\ c[5][5] = 1,\ c[6][5] = 1,\ c[7][5] = 1,$
$c[2][6] = 1,\ c[10][6] = 1,\ c[11][6] = 1$, Phd, Pwh, $Phdw$ according to 1.3 ;
repeat
  $total - Phdw = 0$;
  for $h \in [1,2]$ **do**
    for $d \in [1,...,6]$ **do**
      for $w \in [1,...,11]$ **do**
        $Phdw[h][d][w] = Phd[h][d] \times Pwh[w][h]$;
        $total - Phdw+ = Phdw[h][d][w]$;
  $total - c - phdw = 0$;
  for $h \in [1,2]$ **do**
    for $d \in [1,...,6]$ **do**
      for $w \in [1,...,11]$ **do**
        $Phdw[h][d][w] / = total - phdw$;
        $total - c - Phdw+ = (c[w][d] \times Phdw[h][d][w])$;
  for $h \in [1,2]$ **do**
    for $d \in [1,...,6]$ **do**
      $Phd[h][d] = 0$;
      for $w \in [1,...,11]$ **do**
        $Phdw[h][d]+ = (c[w][d] \times Phdw[h][d][w])$;
  for $h \in [1,2]$ **do**
    for $d \in [1,...,6]$ **do**
      $Phd[h][d] / = total - c$;
  for $w \in [1,...,11]$ **do**
    for $h \in [1,2]$ **do**
      $P[w][h] = 0$;
      for $d \in [1,...,6]$ **do**
        $P[w][h]+ = (c[w][d] \times Phdw[h][d][w])$;
      $P[w][h] / = total - c - phdw$
  print: "$phd\backslash n$";
  for $h \in [1,2]$ **do**
    for $d \in [1,...,6]$ **do**
      $printPhd[h][d]$;
    $print\backslash n$;
  print: "$phd\backslash n$";
  for $w \in [1,...,11]$ **do**
    for $h \in [1,2]$ **do**
      $printP[w][h]$;
    $print\backslash n$;
until *coverage*;

Chapter 7 Reference Answers

7.1 (a) Parameter type are $P(t_i)$ and $P(w_j, t_i)$. Parameter instances are $P(t_i = t)$, $P(w_{i-2} \mid t_i = t)$, $P(w_{i-1} \mid t_i = t)$, $P(w_i \mid t_i = t)$, $P(w_{i+1} \mid t_i = t)$, and $P(w_{i+2} \mid t_i = t)$.



(b) Yes, There are six parameter types, which are class label and five words given class label and different positions. And parameter instances are $P(t_i = t)$, $P(w_j \mid t_i = t, j = i-2)$, $P(w_j \mid t_i = t, j = i-1)$, $P(w_j \mid t_i = t, j = i)$, $P(w_j \mid t_i = t, j = i+1)$, and $P(w_j \mid t_i = t, j = i+2)$.Therefore, the size of model is same as model in (a).

(c)Feature templates are t, and wtp, in which t is POS-tag, w is word, and p is position index. $\vec{v}(w,t,p) =< t_1, t_2, ..., t_{|L|}, w_1 t_1 p_{i-2}, ..., w_{|V|} t_{|L|} p_{i+2} >$. We can add addtional feature such as wilder threshold window of words and bigram condition of POS-tag.

7.2 If $T_{1:i-1}\,(t_{i-1} = \hat{t}_{i-1})$ is not the highest-scored. Assume that $T_{1:i-1}\left(t_{i-1} = \hat{t}'_{i-1}\right)$ is the highest-scored, in which $\hat{t}'_{i-1} \neq \hat{t}_{i-1}$. Because the last tag is $\hat{t}_i$. Therefore, we get the highest-scored sequence $T_{1:i}\left(t_{i-1} = \hat{t}'_{i-1}, t_i = \hat{t}_i\right)$. However, the last two tags of the highest-scored sequence $\hat{T}_{1:i}$ is $\hat{t}_{i-1}$ and $\hat{t}_i$, which contracts to the assumption. Therefore, $\hat{t}'_{i-1} = \hat{t}_{i-1}$.

7.3 Assume that $T_{1:i-1}\left(t_{i-2}t_{i-1} = \hat{t}'_{i-2}\hat{t}'_{i-1}\right)$ of $\hat{T}_{1:i}$ is the highest-scored sequence and $\hat{t}'_{i-2}\hat{t}'_{i-1} \neq \hat{t}_{i-2}\hat{t}_{i-1}$. Because the last tag is $\hat{t}_i$. Therefore, we get the highest-scored sequence $T_{1:i}\left(t_{i-2}t_{i-1} = \hat{t}'_{i-2}\hat{t}'_{i-1}, t_i = \hat{t}_i\right)$. However, the last three tags of the highest-scored sequence $\hat{T}_{1:i}$ is $\hat{t}_{i-2}$, $\hat{t}_{i-1}$ and $\hat{t}_i$, which contracts to the assumption. Therefore, $\hat{t}'_{i-2}\hat{t}'_{i-1} = \hat{t}_{i-2}\hat{t}_{i-1}$.

7.4

	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$
$<B><B>$	1						
$<B>l_1$		0.3					
$<B>l_2$		0.04					
$<B>l_3$		0.16					
l_1l_1			0.012	0.00032	0.0000192	0.000006144	0.000003888
l_1l_2			0.003	0.00192	0.0001152	0.000003072	0.0000015552
l_1l_3			0.012	0.00096	0.0000576	0.000004608	0.0000082944
l_2l_1			0.0048	0.00064	0.000012	0.00002592	0.000005832
l_2l_2			0.0016	0.0024	0.001296	0.00023328	0.0000419904
l_2l_3			0.0012	0.00056	0.000012	0.00000648	0.0000093312
l_3l_1			0.016	0.00096	0.0000768	0.000018432	0.00000162
l_3l_2			0.016	0.00072	0.000576	0.000001536	0.0000003888
l_3l_3			0.004	0.00012	0.0000096	0.0000007680	0.00000103680

7.5

7.6 execute program.

7.7 The first-order HMM in conditioned by previous one POS-tag, while second-order HMM consider previous two POS-tag. The oth HMM model is $P(T_{1:n} \mid W_{1:n}) = \frac{P(W_{1:n}|T_{1:n})P(T_{1:n})}{P(W_{1:n})}$, in which $P(W_{1:n} \mid T_{1:n})$ equals to $\sum_1^n P(W_i \mid T_i)$ and $P(T_{1:n})$ equals to $\sum_1^n P(t_i)$. Compared oth HMM model with first-order and second-order HMMs, it is a greedy search, which only considers current step value. In addition, it predicts each POS-tag given an single word. Therefore, oth HMM model might have lower performance than others, which have more parameter instances.

Chapter 8 Reference Answers

8.1 See Algorithm 7

Algorithm 7: Viterbi decoding for second-order MEMM in 8.1

Inputs: $s = W_{1:n}$, second-order POS tagging model with feature vector $\vec{\phi}(t_i, t_{i-1}, t_{i-2}, W_{1:n})$, and feature weight vector $\vec{\theta}$;
Variables:tb, bp;
Initialization:$tb[t'][t][i] \leftarrow -\infty$; $bp[t'][t][i] \leftarrow NULL$ for $t, t' \in L \cup \{\langle B \rangle\}$, $i \in [0, ..., n]$, $tb[\langle B \rangle][\langle B \rangle][0] \leftarrow 0$;
for $t \in L$ **do**
  $tb[\langle B \rangle][t][1] \leftarrow tb[\langle B \rangle][\langle B \rangle][0] + \vec{\theta} \cdot \vec{\phi}(t, \langle B \rangle, \langle B \rangle, W_{1:n})$;
$y_1 \leftarrow argmax_{\langle B \rangle t}\ tb[\langle B \rangle][t][1]$;
for $t \in L$ **do**
  for $t' \in L$ **do**
    $score \leftarrow \vec{\theta} \cdot \vec{\phi}(t, t', \langle B \rangle, W_{1:n})$;
    if $tb[t'][t][2] < tb[\langle B \rangle][t'][1] + score$ **then**
      $tb[t'][t][2] \leftarrow tb[\langle B \rangle][t'][1] + score$;
      $bp[t'][t][2] \leftarrow \langle B \rangle t'$;
for $i \in [2, \ldots, n]$ **do**
  for $t \in L$ **do**
    for $t' \in L$ **do**
      for $t'' \in L$ **do**
        $score \leftarrow \vec{\theta} \cdot \vec{\phi}(t, t', t'', W_{1:n})$;
        if $tb[t'][t][i] < tb[t''][t'][i-1] + score$ **then**
          $tb[t'][t][i] \leftarrow tb[t''][t'][i-1] + score$;
          $bp[t'][t][i] \leftarrow t''t'$;
$y_n \leftarrow argmax_{t't}\ tb[t'][t][n]$;
for $i \in [n, \ldots, 3]$ **do**
  $y_{i-1} \leftarrow bp[y_i][n]$;
Outputs:$y_1, ..., y_n$

8.2 (a)(b)(e)(f)

8.3 (a) No. The feature should be $\phi(t_i, t_{i-1}, w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2})$.
(b) Because the target is to score the probability of t_i.

(c) No. All w_i is known condition.
(d) Yes. Optimal sub-problem will be affected, which leads to the final result of dynamic programming change.
(e) No. The feature instances is unchangeable. Therefore, the value of i will not affect feature instances.
(f) Yes. The decoding process uses dynamic programming to decrease the complexity of calculation.
(g) See Algorithm 8

Algorithm 8: First-order Forward-Backward Algorithm in 8.3

Inputs: $s = W_{1:n}$, first-order CRF model for POS tagging with feature vector $\vec{\phi}(t_i, t_{i-1}, W_{1:n})$, and feature weight vector $\vec{\theta}$;
Variables:pt, α, β;
$pt \leftarrow 0$;
$\alpha \leftarrow \text{FORWARD}(W_{1:n}, model)$;
$\beta \leftarrow \text{BACKWARD}(W_{1:n}, model)$;
for $i \in [1, \ldots, n]$ **do**
  $Z \leftarrow 0$;
  for $T_i \in [T_1, ..., T_n]$ **do**
    $Z \leftarrow Z + T_i$
  for $t \in L$ **do**
    $pt[i] \leftarrow \frac{\alpha[t][i] \times \beta[t][i]}{Z}$;
Outputs:pt

8.4 See Algorithm 9 for Forward Algorithm for Second-order CRF.

Algorithm 9: Forward Algorithm for Second-order CRF in 8.4

Inputs: $s = W_{1:n}$, second-order CRF model for POS tagging with feature vector $\vec{\phi}(t_i, t_{i-1}, t_{i-2}, W_{1:n})$, and feature weight vector $\vec{\theta}$;
Variables:α;
Initialization:$\alpha[0][\langle B \rangle][\langle B \rangle] \leftarrow 1$;$\alpha$;
for $i \in [1, \ldots, n]$ **do**
  $Z \leftarrow 0$;
  for $T_i \in [T_1, ..., T_n]$ **do**
    $Z \leftarrow Z + T_i$
  for $t \in L$ **do**
    $pt[i] \leftarrow \frac{\alpha[t][i] \times \beta[t][i]}{Z}$;
Outputs:pt

8.5 The features for POS tagging in Table 8.1 are useful for SRL, because POS is of great importance to SRL. Other features are also useful, such as position index, voice, idoim type, hypotaxis and the first and last words of an argument. The verb phrase in syntax tree can be used for SRL. All the features above will affect decoding efficiency, because they are calculated during decoding by dynamic programming.

Task	Generative Model	Discriminative Model
Classification	(d)	Log-linear model Perception SVM (b) (c)
Sequence labelling	(e)	Logistic regression (a)

Table 1.4: 8.7 Correlation between different models

8.6 Discriminative model such as CRF have the advantage of enabling rich features, as compared with HMM. In other word, word segmentation needs rich features to solve the problem. However, the insufficient features of HMM limits the accuracy. In addition, inconsistency between local training and global testing can cause accuracy loss.

8.7 See Table 1.4

8.8 CRFs' output scores are based on probability, while structured perceptrons don't output probability. The training speed of perceptron is fast than CRF because it doesn't have forward-backward process. They have same decoding speed because they both calculate the product of $\vec{\theta}$ and $\vec{\phi}$. Two method also have equal freedom in defining rich feature because they both use dynamic programming to calculate marginal probability.

Chapter 9 Reference Answers

9.1 First, we should calculate the marginal probability of w_j ... Wk. Then we take first-order CRF as an example to derive the marginal probability.

$$
\begin{aligned}
&P(t_j...t_k = B-PER...E-PER \mid W_{1:n}) \\
&= \sum_{t_1 \in L} \cdots \sum_{t_{j-1} \in L} \sum_{t_{k+1} \in L} \cdots \sum_{t_n \in L} \frac{1}{Z} \prod_{i=1}^{n} \exp\left(\vec{\theta} \cdot \vec{\phi}\left(t_i, t_{i-1}, W_{1:n}\right)\right), \\
&\quad where\ t_j...t_k = B-PER...E-PER \\
&= \frac{1}{Z}\left(\sum_{t_1 \in L} \cdots \sum_{t_{j-1} \in L} \prod_{i=1}^{j} \exp\left(\vec{\theta} \cdot \vec{\phi}\left(t_i, t_{i-1}, W_{1:n}\right)\right)\right) \cdot \\
&\quad \left(\sum_{t_{i+1} \in L} \cdots \sum_{t_n \in L} \prod_{i=k}^{n} \exp\left(\vec{\theta} \cdot \vec{\phi}\left(t_i, t_{i-1}, W_{1:n}\right)\right)\right), \\
&\quad where\ t_j...t_k = B-PER...E-PER
\end{aligned}
$$

According to Equation 8.11 and Equation 8.12 in Chapter 8, we can also get

$$
\begin{aligned}
\alpha(j,t) &= \sum_{t' \in L}\left(\alpha\left(j-1,t'\right) \cdot \exp\left(\vec{\theta} \cdot \vec{\phi}\left(t_j = B-PER, t_{j-1} = t', W_{1:n}\right)\right)\right) \\
\beta(k,t) &= \sum_{t' \in L}\left(\beta\left(k+1,t'\right) \cdot \exp\left(\vec{\theta} \cdot \vec{\phi}\left(t_{k+1} = t', t_k = E-PER, W_{1:n}\right)\right)\right)
\end{aligned}
$$

Finally, we can calculate the probability according to Algorithm 8.4 in Chapter 8.

9.2 (a) $\vec{\phi}(w_1,w_2) = \vec{\phi}(\text{"}take\text{"},\text{"}thedaybeforeyesterday\text{"})$
$\vec{\phi}(w_2,w_3) = \vec{\phi}(\text{"}thedaybeforeyesterday\text{"},\text{"}rain\text{"})$
$\vec{\phi}(w_3,w_4) = \vec{\phi}(\text{"}rain\text{"},\text{"}forexample\text{"})$
(b) word w_j, word bigram$w_{j-1}w_j$, whether w_j is a single-character word, SINGLE(w_j), $c_{b(j)}c_{e(j)}$

9.3 Besides the POS-tagging feature can be add to syntactic chunking and NER, previous chunking label can add to feature templates because it can prevent model prediction obvious wrong labels. The main differences between word segmentation and the other two tasks is that syntactic chunking and NER have category labels.

As for the similarity, they all have to segment words. Compared with Table 9.2, Table 9.3 focus on relation between word and POS-tagging. Compared with Table Table 9.2, Table 9.4 uses more features to represent words such as prefix and word shape.

9.4 (a) (b) The three for-loop of Algorithm will be changed. $e \in [2,n]$ keeps

Algorithm 10: Recover the word sequence from the table bp

Inputs: bp,last word's index b and e from the highest-scored $table[b,e]$, Sequence $C_{1:n} = c_1 c_2 ... c_n$;
Initialisation: Segmented sequence $W \leftarrow$ NULL ;
for $b \neq 0$ **do**
  $W.append(C_{b,e})$;
  $b' \leftarrow bp[b,e]$;
  $e \leftarrow b$;
  $b \leftarrow b' - 1$;
Outputs: REVERSE($W_{1:|W|}$);

unchanged. b turns to $b \in [max(2, e-M-1), e]$ and b' turns to $b' \in [max(1, b-M), b-1]$. The asymptotic complexity of new algorithm is $O(M^2 n)$
(c) Yes. decoding algorithm will add one more for-loop to record the third word.

9.5 (a) HSMM and HMM have similar training and decoding process. However, the emission probability of HSMM are composed of several emission probabilities of generating each character given t_i.

9.6 (a) 0th order semi-CRF only considers one output segment. Therefore, the decoding process becomes a greedy search decoding.
(b) Since there is no dependency between two nearby output segments, the process of calculating marginal probabilities of segments become a independent process. So that it is a simple classification process given the current subsequence of characters.
(c) As for large-margin models, it becomes to the same as perceptron.
(d) See Algorithm 11 for decoding.

Algorithm 11: Dynamic programming decoder for sequence segmentation in 9.6

Inputs: Sequence $C_{1:n} = c_1 c_2 ... c_n$, and model parameters $\vec{\theta}$;
Variables:tb, bp;
Initialization:
for $e \in [1, ..., n]$ **do**
  $table[e] \leftarrow -\infty$;
  $bp[e] \leftarrow -1$;
  $table[e] \leftarrow \vec{\theta} \cdot \vec{\phi}_c(C_{1:n}, 0, e, l(0, e))$;
Algorithm:
for $e \in [2, ..., n]$ **do**
  for $e' \in [2, ..., e]$ **do**
    if $table[e'] + \vec{\theta} \cdot \vec{\phi}_c(C_{1:n}, e', e, l(e', e)) > table[e]$ **then**
      $table[e] \leftarrow table[e'] + \vec{\theta} \cdot \vec{\phi}_c(C_{1:n}, e', e, l(e', e))$;
      $bp[e] \leftarrow e'$;
$max\_score \leftarrow table[n]$; $W_{1:|W|} \leftarrow$ back trace with bp;
Outputs:Segmented sequence$W_{1:|W|} = w_1 w_2 ... w_{|W|}$;

(e) See Algorithm 12 for decoding.

Algorithm 12: Dynamic programming decoder for sequence segmentation with label in 9.6

Inputs: Sequence $C_{1:n} = c_1 c_2 ... c_n$, and model parameters $\vec{\theta}$;
Variables:tb, bp;
Initialization:
for $e \in [1, ..., n]$ **do**
  $table[e] \leftarrow -\infty$;
  $bp[e] \leftarrow 0$;
  $table[e] \leftarrow \vec{\theta} \cdot \vec{\phi}_c(C_{1:n}, l(0,0), 0, e, l(0, e))$;
Algorithm:
for $e \in [2, ..., n]$ **do**
  for $e' \in [1, ..., e-1]$ **do**
    for $b \in [1, ..., e'-1]$ **do**
      if $table[e'] + \vec{\theta} \cdot \vec{\phi}_c(C_{1:n}, l(b, e'-1), e', e, l(e', e)) > table[e]$
      then
        $table[e] \leftarrow table[e'] + \vec{\theta} \cdot \vec{\phi}_c(C_{1:n}, l(b, e'-1), e', e, l(e', e))$;
        $bp[e] \leftarrow e'$;
$max\_score \leftarrow table[n]$; $W_{1:|W|} \leftarrow$ back trace with bp;
Outputs:Segmented sequence$W_{1:|W|} = w_1 w_2 ... w_{|W|}$;

9.7 For Algorithm 9.1, Algorithm 9.2, Algorithm 9.3, and Algorithm 9.4, two for-loop should add to take labels of two output segments, in which $l, l' \in L$.

9.8 $\vec{\theta} \leftarrow \vec{\theta} + \vec{\phi}(pos) - \vec{\phi}(neg)$ performs early update. Fourth line from last **return;** ensures that the algorithm does not proceed with the current training instance after the update is executed. $\vec{\theta} \leftarrow \vec{\theta} + \vec{\phi}\left(G_{1:n}\right) - \vec{\phi}(\text{ Top-k (agenda, 1)[0]})$ performs the final update as in the standard structured perceptron. In the early update the local features range from step one to step i, while final update the $\vec{\phi}_{\Delta}(W_{1:n}, T_{1:i-1}, t)$ ranges from one to n.

9.9 According to Equation 8.23, we add $\Delta\left(\hat{W}_i', W_i\right)$ to Equation 9.15 $\frac{1}{2}\|\vec{\theta}\|^2 + C\left(\sum_{i=1}^{N} \max\left(0, 1 - \vec{\theta}\cdot\vec{\phi}\left(W_i\right) + \max_{W' \neq W_i}\left(\vec{\theta}\cdot\vec{\phi}\left(W'\right)\right)\right)\right)$, then we get $\frac{1}{2}\|\vec{\theta}\|^2 + C\left(\sum_{i=1}^{N} \max\left(0, \Delta\left(\hat{W}_i', W_i\right) - \vec{\theta}\cdot\vec{\phi}\left(W_i\right) + \left(\vec{\theta}\cdot\vec{\phi}\left(\hat{W}_i'\right)\right)\right)\right)$, where $\hat{W}_i' = \max_{W' \neq W_i}$ $\left(\Delta\left(W', W_i\right) + \vec{\theta}\cdot\vec{\phi}\left(W'\right)\right)$. In particular,$\Delta\left(\hat{W}_i', W_i\right) = \sum_{i=1}^{n} \delta\left(w_i', w_i\right)$. Therefore, we get cost-augmented decoding that is same as Equation 8.24.

Chapter 10 Reference Answers

10.1 (a) NP: boy; VP: likes
(b)
(c)

10.2 Total number of possible Binary Search Trees with n different keys ($countBST(n)$) = Catalan number $Cn = \frac{(2n)!}{(n+1)!*n!}$ For $n = 0,1,2,3,\ldots$ values of Catalan numbers are $1,1,2,5,14,42,132,429,1430,4862,\ldots$. So are numbers of Binary Search Trees. Total number of possible Binary Trees with n different keys ($countBT(n)$) = $countBST(n) * n!$

10.3 Assume that $score(\hat{T}(i,i+s-1,c))$ of $\hat{T}_{1:i}$ is the highest-scored tree where $\hat{j}$ makes it equal to $\big(score(\hat{T}(i,j-1,c_1)) + score(\hat{T}(j,i+s-1,c_2)) + logP(c \Rightarrow c_1c_2)\big)$. If $\max\limits_{c_1,c_2 \in C, j \in [i+1,\ldots,i+s-1]} \big(score(\hat{T}(i,i+s-1,c)) + score(\hat{T}(i,i+s-1,c)) + logP(c \Rightarrow c_1c_2)\big)$ is the highest score when $j' \neq \hat{j}$. Then we have $score(T'i,i+s-1,c)) > score(\hat{T}(i,i+s-1,c))$, which contracts to the assumption. Therefore, $j' = \hat{j}$ and $score(\hat{T}(i,i+s-1,c))$ of $\hat{T}_{1:i}$ is the highest-scored tree.

10.4 See Algorithm 13 for algorithm FINDDERIVATION for CKY.

Algorithm 13: FINDDERIVATION for CKY when given the pointer bp in 10.4

Inputs: bp, $bp[s][i][c]$;
Variables: j, c_1, c_2;
Initialization:
for $e \in [1,...,n]$ **do**
    $table[e] \leftarrow -\infty$;
    $bp[e] \leftarrow 0$;
    $table[e] \leftarrow \vec{\theta} \cdot \vec{\phi}_c(C_{1:n}, l(0,0), 0, e, l(0,e))$;
Algorithm:
j,c_1,c_2 = FINDDERIVATION($bp[s][i][c]$);
if $j-1=1$ **then**
    SETLEFTNODE(w_i,c_1,c);
else
    FINDDERIVATION($bp[j-1][i][c_1]$);
if $s-j+1=1$ **then**
    SETRIGHTNODE(w_j,c_2,c);
else
    FINDDERIVATION($bp[s-j+1][j][c_2]$);
Outputs:Tree root node c;

10.5 For unary rule, We can update the score of chart cell. The loop rule might lead to infinite loop.

10.6 Equation 10.6 separately calculates the total probability of $outside(i,j,c)$, while Algorithm 10.3 calculates two segment at each step. The ranges of s, i, j, c_1, c_2 and c are the same.

10.7 See Algorithm 14 for CKY algorithm.

Algorithm 14: CKY algorithm in 10.7

Inputs: $W_{1:n} = w_1 w_2 ... w_n$, $PCFGmodelP(c \rightarrow c_1 c_2)$, $P(c \rightarrow w)$;
Variables:$chart$, bp;
Initialization:
for $i \in [1, ..., n]$ **do**
  for $c \in C$ **do**
    $chart[1][i][1][c] \leftarrow logP(c \rightarrow w_i)$;
for $s \in [2, ..., n]$ **do**
  for $i \in [1, ..., n-s+1]$ **do**
    for $c \in C$ **do**
      for $m \in [1, .., k]$ **do**
        $chart[s][i][m][c] \leftarrow -\infty$;
        $bp[s][i][c] \leftarrow$ -1;
Algorithm:
for $s \in [2, ..., n]$ **do**
  for $i \in [1, ..., n-s+1]$ **do**
    for $m \in [1, .., k]$ **do**
      for $m' \in [1, .., k]$ **do**
        for $c, c_1, c_2 \in C$ **do**
          $score \leftarrow chart[j-1][i][m][c_1] + chart[s-j+1][j][m'][c_2] + logP(c \rightarrow c_1 c_2)$;
          if TOPK$(chart[s][i][m][c]) > score$ **then**
          else
            $s, i, m, c =$MIN$(chart[s][i][m][c])$;
            $chart[s][i][m][c] \leftarrow score$;
            $bp[s][i][c] \leftarrow (j, c_1, c_2)$;
Outputs:FINDDERVATION$(bp[n][1][$TOPK$(chart[n][1][m][c])])$;

10.8 Local rules and Non-local rules can be treated differently to get better performance. Local features such as $VP \rightarrow VBDNPPP$ is useful. Non-local patterns such as ParentRule and NGramTree. For example, $< VP \rightarrow VBDNPPP|S >$ and $< VP(VBDsaw)(NP(DTthe)) >$.

10.9

10.10

10.11 We can use beam search to save k highest score each step, then we get k-best output. However, because best step might lost during beam search the k-best output candidates are not the k highest-scored outputs.

10.12 Hidden variables H is POS sequence corresponding to word sequence.The observed variables O the sequence of words.

Chapter 11 Reference Answers

11.1 Compared with features F_1 in Table 9.5, features F_2 in Table 11.2 are more non-local. We can see the definition of features F_2 are not restricted by dynamic programming so that it has large span, which doesn't limit to two step. Besides, F_1 and F_2 both have unigram and bigram features.

11.2 SEE BOOK NOTE

11.3 Similar to the actions of transition-based word segmentation in Table 11.1, the state transition system for joint word segmentation and POS-tagging just needs to adopt SEP to SEP-P, which assign POS-tag of each word when separating.

11.4 I have found the slides. Typing...

11.5 Draft P.53

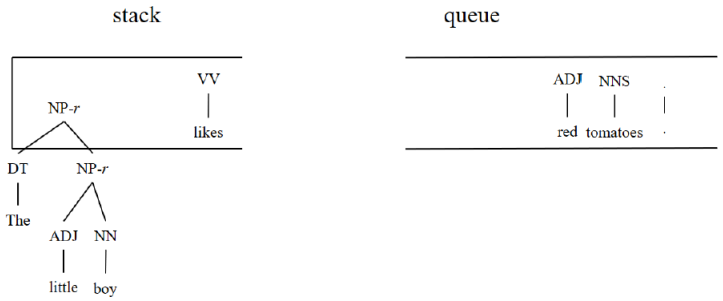
11.6 Difference bewteen Table 11.6? (Transition-based Dependency Parsing with Rich Non-local Features)

11.7 Dependency parsing only has a set of $n-1$ dependency arcs between words, while constituent parsing doesn't have such restriction. As a result, constituent parsing just combine previous nodes into one node rather than poping words out, leading to more complex phrase structure hierarchies.

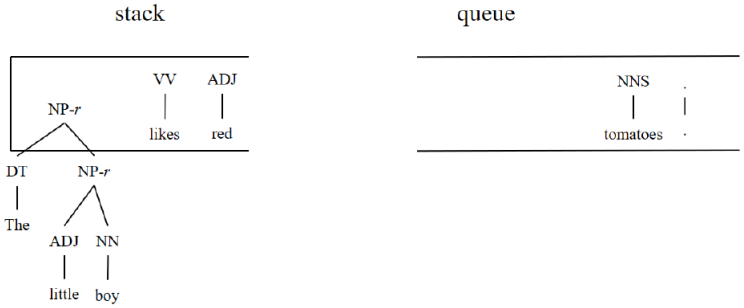
11.8 For character-based Chinese parsing, part-of-speech information can be leveraged to segmentation of words, which considers internal structures of words that have syntactic relations when forming Chinese words.
HOW CAN TRANSTITION SYSTEM BE EXTENDED TO...

11.9 We design model that modifies arc-standard model with annotated intra-word dependencies and real inter-word dependencies.The annotated intra-word dependencies refer to the dependencies extracted from annotated word structures,

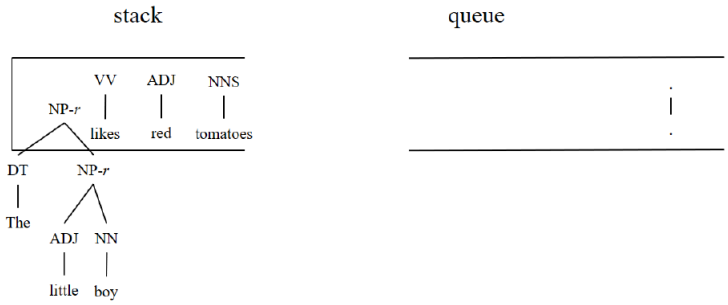
○ Shift



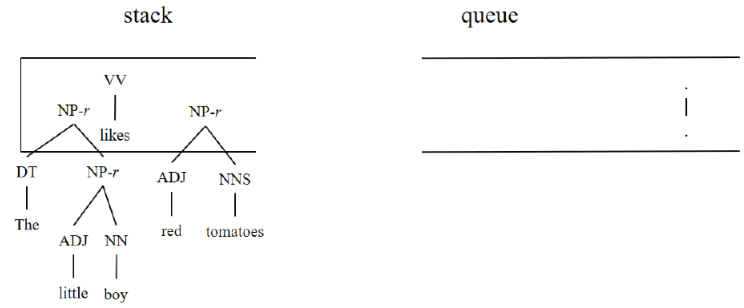
○ Shift



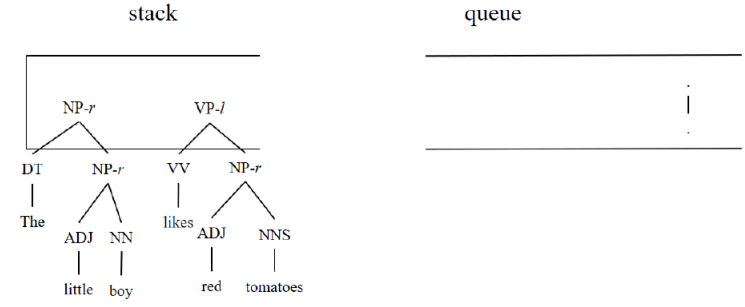
○ Reduce-R-NP



○ Reduce-L-VP



○ Shift



○ Reduce-L-S

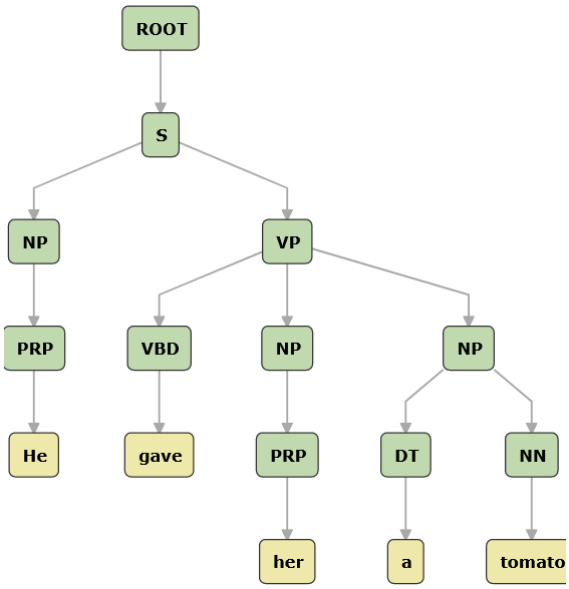


Figure 1.2: 10.7 constituent tree

Category	Feature templates	When to Apply
Structure features	S_0ntl S_0nwl S_1ntl S_1nwl S_2ntl S_2nwl S_3ntl S_3nwl, Q_0c Q_1c Q_2c Q_3c $Q_0c \cdot Q_1c$ $Q_1c \cdot Q_2c$ $Q_2c \cdot Q_3c$, $S_{0l}twl$ $S_{0r}twl$ $S_{0u}twl$ $S_{1l}twl$ $S_{1r}twl$ $S_{1u}twl$, $S_0nw \cdot S_1nw$ $S_0nw \cdot S_1nl$ $S_0nl \cdot S_1nw$ $S_0nl \cdot S_1nl$, $S_0nw \cdot Q_0c$ $S_0nl \cdot Q_0c$ $S_1nw \cdot Q_0c$ S_1nlQ_0c, $S_0nl \cdot S_1nl \cdot S_2nl$ $S_0nw \cdot S_1nl \cdot S_2nl$ $S_0nl \cdot S_1nw \cdot S_2nl$ $S_0nl \cdot S_1nl \cdot S_2nw$, $S_0nw \cdot S_1nl \cdot Q_0c$ $S_0nl \cdot S_1nw \cdot Q_0c$ $S_0nl \cdot S_1nl \cdot Q_0c$, **$S_0ncl$ S_0nct S_0nctl S_1ncl S_1nct S_1nctl, S_2ncl S_2nct S_2nctl S_3ncl S_3nct S_3nctl, $S_0nc \cdot S_1nc$ $S_0ncl \cdot S_1nl$ $S_0nl \cdot S_1ncl$ $S_0ncl \cdot S_1ncl$, $S_0nc \cdot Q_0c$ $S_0nl \cdot Q_0c$ $S_1nc \cdot Q_0c$ $S_1nl \cdot Q_0c$, $S_0nc \cdot S_1nc \cdot Q_0c$ $S_0nc \cdot S_1nc \cdot Q_0c \cdot Q_1c$**	All
	start$(S_0w) \cdot$ start(S_1w) start$(S_0w) \cdot$ end(S_1w), indict$(S_1wS_0w) \cdot$ len(S_1wS_0w) indict$(S_1wS_0w, S_0t) \cdot$ len(S_1wS_0w)	REDUCE-SUBWORD
String features	$t_{-1} \cdot t_0$ $t_{-2} \cdot t_{-1}t_0$ $w_{-1} \cdot t_0$ $c_0 \cdot t_0$ start$(w_{-1}) \cdot t_0$ $c_{-1} \cdot c_0 \cdot t_{-1} \cdot t_0$,	SHIFT-SEPARATE
	w_{-1} $w_{-2} \cdot w_{-1}$ w_{-1}, where len$(w_{-1}) = 1$ end$(w_{-1}) \cdot c_0$, start$(w_{-1}) \cdot$ len(w_{-1}) end$(w_{-1}) \cdot$ len(w_{-1}) start$(w_{-1}) \cdot$ end(w_{-1}), $w_{-1} \cdot c_0$ end$(w_{-2}) \cdot w_{-1}$ start$(w_{-1}) \cdot c_0$ end$(w_{-2}) \cdot$ end(w_{-1}), $w_{-1} \cdot$ len(w_{-2}) $w_{-2} \cdot$ len(w_{-1}) $w_{-1} \cdot t_{-1}$ $w_{-1} \cdot t_{-2}$ $w_{-1} \cdot t_{-1} \cdot c_0$, $w_{-1} \cdot t_{-1} \cdot$ end(w_{-2}) $c_{-2} \cdot c_{-1} \cdot c_0 \cdot t_{-1}$, where len$(w_{-1}) = 1$ end$(w_{-1}) \cdot t_{-1}$, $c \cdot t_{-1} \cdot$ end(w_{-1}), where $c \in w_{-1}$ *and* $c \neq$ end(w_{-1})	REDUCE-WORD
	$c_0 \cdot t_{-1}$ $c_{-1} \cdot c_0$ start$(w_{-1}) \cdot c_0t_{-1}$ $c_{-1} \cdot c_0 \cdot t_{-1}$	SHIFT-APPEND

Figure 1.3: 10.8 Joint feature template

while the pseudo intra-word dependencies used in the above models are similar to those of Hatori's work (Incremental joint approach to word segmentation, pos tagging, and dependency parsing in chinese).

11.10 We design some actions of system, which is character-level system that has state stack, words buffer and input character set: (a) SHIFT-NER, which removes the front character from input set, assigning the NER label NER to the character and pushing it onto the buffer. (b) SEP-S, which removes the all characters from buffer, assigning the sentiment label S to character span and pushing it onto the stack.

11.11 - S-SHIFT, which copies 3 an item from the front of B and pushes it on S.
- S-REDUCE pops an item from S.
- S-RIGHT (ℓ) creates a syntactic dependency. Let u be the element at the top of S and v be the element at the front of B. The new dependency has u as head, v as dependent, and label ℓ. u is popped off S, and the resulting structure, rooted at u, is pushed on S. Finally, v is copied to the top of S.
- S-LEFT (ℓ) creates a syntactic dependency with label ℓ in the reverse direction as S-RIGHT. The top of S, u, is popped. The front of B, v, is replaced by the new structure, rooted at v.
- M-SHIFT removes an item from the front of B and pushes it on M.
- M-REDUCE pops an item from M.
- M-RIGHT (r) creates a semantic dependency. Let u be the element at the top of M and v, the front of B. The new dependency has u as head, v as dependent, and label $r.u$ is popped off M, and the resulting structure, rooted at u, is pushed on M.
- M-LEFT (r) creates a semantic dependency with label r in the reverse direction as MRIGHT. The buffer front, v, is replaced by the new v-rooted structure. M remains unchanged.
- M-SWAP swaps the top two items on M, to allow for crossing semantic arcs.
- M-PRED(p) marks the item at the front of B as a semantic predicate with the sense p, and replaces it with the disambiguated predicate.
- M-SELF (r) adds a dependency, with label r between the item at the front of B and itself. The result replaces the item at the front of B.

11.12 A state can be (α, β, A), where α represents a stack of words being processed with mention labels, β represents a buffer of next incoming words and A represents the set of relations between words. The set of transition actions include:
- SHIFT, which removes the front word from the buffer, pushing it onto the stack, and assigning mention labels;
- POP, which removes the front word from the buffer, which is not a mention;
- RELATION-X, which constructs a relation with label X between the top two mentions on the stack;

- SWAP, which removes the second top word from the stack, putting it back to the buffer front.

11.13 As long as least one gold-standard sequence is in the beam, we don't update the parameters.

Chapter 12 Reference Answers

12.1

$$P(x_1,x_2,x_3) = P(x_1)P(x_3 \mid x_1)P(x_2 \mid x_1,x_3)$$
$$\frac{P(x_1,x_2,x_3)}{P(x_1)} = P(x_2,x_3 \mid x_1)$$
We know $x_2 \perp x_3 \mid x_1$
$$P(x_2,x_3 \mid x_1) = P(x_2 \mid x_1)P(x_3 \mid x_1)$$
Therefore, $P(x_2 \mid x_1,x_3) = P(x_2 \mid x_1)$

12.2

12.3 (a) Yes, because they are parent of x_5.
(b) x_4 and x_5.
(c) Seven parameters are needed.
(d)

$$\begin{aligned} P(x_1,x_2,x_3,x_4,x_5) &= P(x_1)P(x_2)P(x_3 \mid x_1,x_2)P(x_4 \mid x_2)P(x_5 \mid x_3,x_4) \\ &= 0.5 \times 0.5 \times 0.9 \times 0.8 \times 0.06 \\ &= 0.0108 \end{aligned}$$

(e) We keep other varible and only consider x_2 and x_5
(f)

12.4

12.5 Yes. We can use MAP and Gipps sampling to choose the model parameters.

12.6 It is a multinomial distribution so that we add Dirichlet prior. Naive Bayes calculates result of dividing the number of a parameter by the total number of parameters, which is same as the formation of add-alpha smooth.

12.7

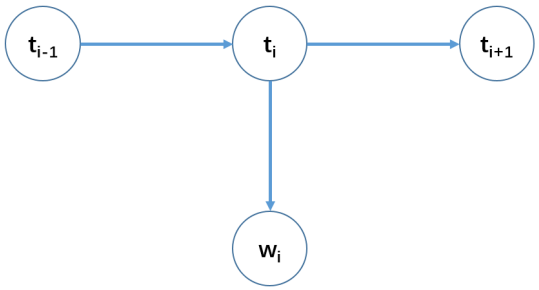


Figure 1.4: relation sequence of 12.8

12.8

$$t_i P(t_i|t_{i-1})P(t_{i+1}|t_i)P(w_i|t_i)$$

When sampling we only have to consider the Markov blanket of a variable.

Chapter 13 Reference Answers

13.1

$$\begin{aligned}\text{sigmoid} &= \frac{1}{1+e^{-x}} = \sigma(x)\\ \sigma(x) &= \frac{1}{(1+e^{-x})'}\\ &= \frac{-1xe^{-x}}{(1+e^{-x})^2}\\ &= \frac{1+e^{-x}-1}{(1+e^{-x})^2}\end{aligned}$$

$$\sigma(x) = \frac{1}{(1+e^{-x})}\left(\frac{1+e^{-x}-1}{1+e^{-x}}\right) = \frac{1}{(1+e^{-x})}\left(1-\frac{1}{1(+e^{-x}}\right) = \sigma(x)(1-\sigma(x))$$

$$\begin{aligned}\sigma(x)'' &= (\sigma(x)(1-\sigma(x)))'\\ &= (\sigma(x)-\sigma(x))' = \sigma(x)' - 2\sigma(x)\cdot\sigma(x)'\\ &= \sigma(x)(1-\sigma(x)-2\sigma(x)(\sigma(x)(1-\sigma(x))\\ &= \sigma(x)(1-\sigma(x))(1-2\sigma(x))\end{aligned}$$

$$\begin{aligned}&\sigma(x) = \sigma(x_0) + \sigma'^{(x_0)(x-x_0)} + \frac{\sigma''^{(x_0)}}{2!}(x-x_0)^2 + O(x)\\ &= \sigma(x_0) + \sigma(x_0)(1-\sigma(x_0))(x-x_0) + \frac{\sigma(x_0)(1-\sigma(x_0))(1-2\sigma(x_0)}{2!}(x-x_0)^2\end{aligned}$$

Therefore, it achieves pair-wise and triple-wise combinations of automatic features. However, single-layer can not combine input vector's multi-dimension.

13.2

$$\begin{aligned}\text{swish}(x) &= x\cdot\text{sigmoid}(\beta x)\\ \frac{\partial swish(x)}{\partial x} &= \frac{\partial(x\cdot\text{sigmoid}(\beta x))}{\partial x}\\ &= \sigma(\beta x)\cdot + x\cdot\sigma'(\beta x)\\ &= \sigma(\beta x) + x\cdot\beta\cdot\sigma(\beta x)(1-\sigma(\beta x))\\ &= \sigma(\beta x)(1+\beta x(1-\sigma(\beta x)))\end{aligned}$$

(b) The gradients of ReLU, LeakyReLU and ELU is fixed when its value greater than 0. When its value less or equal to 0, the gradients of RelU and LeakyReLU are fixed while ELU's is αe^x.

13.3

$$
\begin{aligned}
y_1 &= \sigma\left(w_{11}^y x_1 + w_{12}^y x_2\right) \\
y_2 &= \sigma\left(w_{21}^y x_1 + w_{12}^y x_2\right) \\
o &= \sigma\left(u_1 y_1 + u_2 y_2\right)
\end{aligned}
$$

$$
\frac{\partial L}{\partial y_1} = -u_1(1-o)
$$

$$
\frac{\partial L}{\partial y_2} = -u_2(1-o)
$$

$$
\frac{\partial L}{\partial x_1} = -(1-0)\left(u_1 w_{11}^y \sigma(w_{11}^y x_1 + w_{12}^y x_2)\left(1-\sigma\left(w_{11}x_1 + w_{12}x_2\right) + u_2 w_{21}\sigma\left(w_2 y_1 + w_{22}^y x_2\right)\left(1-\sigma\left(w_{21}^y x\right.\right.\right.\right.
$$

$$
\frac{\partial L}{\partial x_2} = -(1-0)\left(u_1 w_{12}^y \sigma(w_{11}^y x_1 + w_{12}^y x_2)\left(1-\sigma\left(w_{11}x_1 + w_{12}x_2\right) + u_2 w_{22}\sigma\left(w_2 y_1 + w_{22}^y x_2\right)\left(1-\sigma\left(w_{21}^y x\right.\right.\right.\right.
$$

$$
\frac{\partial L}{\partial u_1} = -y_1(1-o) + 2\lambda u_1 \tag{1.1}
$$

$$
\frac{\partial L}{\partial u_2} = -y_2(1-o) + 2\lambda u_2 \tag{1.2}
$$

$$
\frac{\partial L}{\partial_{w_{11}}^y} = -u_1\sigma\left(\omega_{11}^y x_1 + \omega_{12}^y x_2\right)x_1(1-o) + 2\lambda w_{11}^y \tag{1.3}
$$

$$
\frac{\partial L}{\partial_{w_{12}}^y} = -u_1\sigma\left(\omega_{11}^y x_1 + \omega_{12}^y x_2\right)x_2(1-o) + 2\lambda w_{12}^y \tag{1.4}
$$

$$
\frac{\partial L}{\partial_{w_{21}}^y} = -u_2\sigma\left(\omega_{21}^y x_1 + \omega_{22}^y x_2\right)x_1(1-o) + 2\lambda w_{21}^y \tag{1.5}
$$

$$
\frac{\partial L}{\partial_{w_{22}}^y} = -u_2\sigma\left(\omega_{21}^y x_1 + \omega_{22}^y x_2\right)x_2(1-o) + 2\lambda w_{22}^y \tag{1.6}
$$

13.4 (1)

$$\begin{aligned}
&\frac{\partial L(\vec{x}_i, c_i, \theta)}{\partial \vec{u}} = \left(\frac{\partial \langle (\vec{x}_i, c_i i\theta)}{\partial \vec{u}_1}, \frac{\partial L(\bar{x}, c_i, \theta)}{\partial u_2} \right) \\
&= \left(\frac{\partial L(\overrightarrow{x_1}, C_i, \theta)}{\gamma_0} \frac{\partial_0}{\partial \vec{u}_1}, \frac{\partial L(\vec{x}_i, c_i, \theta)}{\partial_0} \frac{\partial_0}{\partial \overrightarrow{u_2}} \right) \\
&= \frac{\partial L}{\partial_0} \cdot \left\langle \frac{\partial_0}{\partial \vec{u}_1} \cdot \frac{\partial_0}{\partial \overrightarrow{u_2}} \right) | \\
&= \frac{\partial L}{\partial 0} \left(\frac{\partial \left(u_1, u_2 \right)^\top \left(y_1, y_2 \right\rangle \right)}{\partial u_1}, \frac{\partial \left(\left\langle u_1, u_2 \right)^\top \left(y, y_2 \right) \right)}{\partial u_2} \right) \\
&= \frac{\partial L}{\partial \mathrm{o}} \left(y_1 \cdot o(1-o), y_2 \cdot o \cdot (1-\mathrm{o}) \right\rangle \\
&= \frac{\partial L}{\partial \mathrm{o}} \cdot \mathrm{o}(1-\mathrm{o}) \vec{y}
\end{aligned}$$

(2)

$$\begin{aligned}
&\frac{\partial L^0}{\partial \vec{w}^y} = \frac{\partial L^0}{\partial \vec{y}} \cdot \frac{\partial \vec{y}}{\partial \vec{w}^y} \\
&= \frac{\partial L^\circ}{\partial \vec{y}} \cdot \begin{pmatrix} \frac{\partial \vec{y}}{\partial w_{11}^y} & , \frac{\partial \vec{y}}{\partial w_{12}^y} \\ \frac{\partial \vec{y}}{\partial w_{21} y}, & \frac{\partial y}{\partial w_{22}^y} \end{pmatrix} \\
&= \frac{\partial L^\circ}{\partial \vec{y}} \cdot \begin{pmatrix} 2x_1 w_{11}^y x_1, & 2x_1 w_{12}^y x_1 \\ 2x_2 w_{21}^y x_2, & 2x_2 w_{22}^y x_2 \end{pmatrix} \\
&= \frac{\partial \mathrm{L}}{\partial y} \theta \left(2 \cdot \vec{w}^y \cdot \vec{x} \right) x^\top
\end{aligned}$$

13.5

$$\begin{aligned}
&\frac{\partial L^\circ}{\partial \vec{w}^y} = \frac{\partial L^\circ}{\partial \vec{y}} \frac{\partial y}{\partial \vec{w}^y} \\
&= \left(\frac{-\log 0}{\partial y_1}, \frac{-\log 0}{\partial y_2} \right) \otimes \begin{pmatrix} 2x_1 w_{11}^y x_1 & 2x_2 w_{12}^y x_2 \\ 2x_1 w_{21}^y x_1, & 2x_2 \omega_2 y_{x_2} \end{pmatrix} \\
&= -(1-0) \otimes \begin{pmatrix} 2x_1 {w_{11}}^y x_1 \cdot & , 2x_2 {w_{12}}^y x_2 \\ 2x_1 {w_{21}}^y x_1, & , 2x_2 {w_{22}}^y x_2 \end{pmatrix}
\end{aligned}$$

Thus, it is equivalent to Eq.13.10

13.6

Because $\Delta W_{ij} \times \dfrac{\partial o}{\partial w[i][j]}$ is the approximation of the change, in which ΔW_{ij} is smaller.

13.7 We assum that we take the derivative with respect to x.
$\frac{\partial t_y}{\partial x} = w^y \ \frac{\partial y}{\partial x} = \frac{\partial y}{\partial t_y} \cdot \frac{\partial t_y}{\partial x} = 2t_y \cdot \omega^y \ \frac{\partial t_0}{\partial x} = \frac{\partial t_0}{\partial y} \cdot \frac{\partial y}{\partial x} = 2t_y \cdot \omega^y \cdot \vec{u}$
The advantage is it can use chain rule of derivatives.

13.8 $H_{1:n-k+1} = P(\text{CNN}(x_1 : N, k, d_0)$
If using max pooling, the gradient at maximum is $\frac{\partial H}{\partial x}$, others are 0.
If using average pooling, the gradient is $\frac{1}{n}\frac{\partial H}{\partial x}$

13.9

$$\frac{\partial H_{1:n-k+1}}{\partial x_i}$$
$$= \sum_{i=low}^{hign} w_i$$
$$\text{low} = \begin{cases} i-k+1 & i-k+1 \geq 1 \\ 1 & \text{else} \end{cases}$$
$$\text{high} = \begin{cases} i+k-1 & \text{if } i+k-1 \leq n-k+1 \\ n+k-1 & \text{else} \end{cases}$$

13.10

For $L = -\sum_{i=1}^{n} \log \vec{P}$

Every value can be non-zero.

For $L = \sum_{i=1}^{N} \max\left(0, -P_i + \max_{c \neq i} p_i\right)$, only the maximum is not zero.

13.11 (a) assume that filter size is k, then the padding size is k-1
(b) n-gram model
(c) add a pooling layer on it

13.12 H has definite meaning when it is in PLSA

13.13 (1) Take a batch to calculate mean and variance
(2)

$$x \in \mathbb{R}^{m \times p \times q}$$
$$\gamma \in \mathbb{R}^m$$
$$\beta \in \mathbb{R}^m$$
$$BN = \gamma \frac{x - E(x)}{V(x)} + \beta$$

13.14

$$\frac{\partial v}{\partial z} = \frac{\partial (z-u)}{\sigma}\alpha + \beta$$
$$= \alpha \cdot \left(\frac{-\frac{1}{\sqrt{2\alpha}}}{b^2} - \frac{\frac{\alpha}{d} - u \cdot \frac{i}{\sqrt{2\alpha}}}{\alpha^2} \right)$$

Chapter 14 Reference Answers

14.1 Suppose that we know $\frac{\partial L}{\partial y_t}, \frac{\partial L}{\partial c_{t+1}}, \frac{\partial L}{\partial o_{t+1}}, \frac{\partial L}{\partial f_{t+1}}, \frac{\partial L}{\partial i_{t+1}}, \frac{\partial L}{\partial g_{t+1}}$. To find the gradient of a node, first find the output node of the node, then calculate the gradient of all output nodes times the gradient of the output node to the node, and finally add them together to get the gradient of the node. We get

$$\frac{\partial L}{\partial h_t} = W_{yh}^T \cdot \frac{\partial L}{\partial y_t} + W_{oh}^T \cdot \frac{\partial L}{\partial o_{t+1}} + W_{fh}^T \cdot \frac{\partial L}{\partial f_{t+1}} + W_{ih}^T \cdot \frac{\partial L}{\partial i_{t+1}} + W_{gh}^T \cdot \frac{\partial L}{\partial g_{t+1}}$$

Compare to the gradient of RNN, there is no factor of the form $((W^h)^T)^t$ in the gradient calculation of LSTM, which leads to vanishing gradients. In addition, f_t, i_t, o_t and c_t are learned by model itself, which means neural network learns to change the gated values to determine when to forget the gradient and when to retain it.

14.2 According to the equation 14.10, we can define bi-directional GRUs(BiGRU). Formally,

$$\begin{aligned}
\overrightarrow{\mathbf{H}} &= \overrightarrow{\mathrm{GRU}}(\mathbf{X}) = \left[\overrightarrow{\mathbf{h}}_1; \overrightarrow{\mathbf{h}}_2; \ldots; \overrightarrow{\mathbf{h}}_n\right] \\
\overleftarrow{\mathbf{H}} &= \overleftarrow{\mathrm{GRU}}(\mathbf{X}) = \left[\overleftarrow{\mathbf{h}}_1; \overleftarrow{\mathbf{h}}_2; \ldots; \overleftarrow{\mathbf{h}}_n\right] \\
BiGRU(\mathbf{X}) &= \overrightarrow{\mathbf{H}} \oplus \overleftarrow{\mathbf{H}} = \left[\overrightarrow{\mathbf{h}}_1 \oplus \overleftarrow{\mathbf{h}}_1; \overrightarrow{\mathbf{h}}_2 \oplus \overleftarrow{\mathbf{h}}_2; \ldots; \overrightarrow{\mathbf{h}}_n \oplus \overleftarrow{\mathbf{h}}_n\right]
\end{aligned}$$

Then we can stack multiple layer BiGRU as stacked GRUs. Formally,

$$\begin{aligned}
&\overrightarrow{\mathbf{H}}^0 = \mathbf{X}, \overleftarrow{\mathbf{H}}^0 = \mathbf{X} \\
&\overrightarrow{\mathbf{H}}^1 = \overrightarrow{GRU}_1\left(\overrightarrow{\mathbf{H}}^0\right), \overrightarrow{\mathbf{H}}^2 = \overrightarrow{GRU}_2\left(\overrightarrow{\mathbf{H}}^1\right), \ldots, \overrightarrow{\mathbf{H}}^l = \overrightarrow{\mathrm{GRU}}_l\left(\overrightarrow{\mathbf{H}}^{l-1}\right) \\
&\overleftarrow{\mathbf{H}}^1 = \overleftarrow{GRU}_1\left(\overleftarrow{\mathbf{H}}^0\right), \overleftarrow{\mathbf{H}}^2 = \overleftarrow{\mathrm{GRU}}_2\left(\overleftarrow{\mathbf{H}}^1\right), \ldots, \overleftarrow{\mathbf{H}}^l = \overleftarrow{\mathrm{GRU}}_l\left(\overleftarrow{\mathbf{H}}^{l-1}\right) \\
&\mathbf{H} = \overrightarrow{\mathbf{H}}^l \oplus \overleftarrow{\mathbf{H}}^l = \left[\overrightarrow{\mathbf{h}}_1^l \oplus \overleftarrow{\mathbf{h}}_1^l; \overrightarrow{\mathbf{h}}_2^l \oplus \overleftarrow{\mathbf{h}}_2^l; \ldots; \overrightarrow{\mathbf{h}}_n^l \oplus \overleftarrow{\mathbf{h}}_n^l\right].
\end{aligned}$$

14.3 First of all, the former method only concatenate forward and backward features at the last layer, which ignore the interaction of them. Although the latter method can make forward and backward features interact more fully, the difference between features is less obvious than the first method.

14.4 (a) Define documents as $D = d_1, d_2, ..., D_{n_D}$, sentences of each document d_i in D as $S_i = s_{i,1}, s_{i,2}, ..., s_{i,n_{S_i}}$ and tokens in each sentences as $X_{i,j,k} = x_{i,j,1}, x_{i,j,2}, x_{i,j,n_{i,j}}$. We separate hierarchical LSTMs into token-level and sentence-level LSTM. Formally, for sentence $s_{i,j}$ of document d_i at token-level

$$\mathbf{h}_1^{token}, \mathbf{c}_1^{token} = \text{LSTM\_STEP}\left(\mathbf{x}_1, \mathbf{h}_0^{token}, \mathbf{c}_0^{token}\right)$$
$$\mathbf{h}_2^{token}, \mathbf{c}_2^{token} = \text{LSTM\_STEP}\left(\mathbf{x}_2, \mathbf{h}_1^{token}, \mathbf{c}_1^{token}\right)$$
$$\mathbf{h}_{n_{i,j}}^{token}, \mathbf{c}_{n_{i,j}}^{token} = \text{LSTM\_STEP}\left(\mathbf{x}_{n_{i,j}}, \mathbf{h}_{n_{i,j}-1}^{token}, \mathbf{c}_{n_{i,j}-1}^{token}\right)$$

We choose the last token's hidden state as the sentence representation. Similar to token-level LSTM, at sentence-level

$$\mathbf{h}_1^{sent}, \mathbf{c}_1^{sent} = \text{LSTM\_STEP}\left(\mathbf{x}_1, \mathbf{h}_0^{sent}, \mathbf{c}_0^{sent}\right)$$
$$\mathbf{h}_2^{sent}, \mathbf{c}_2^{sent} = \text{LSTM\_STEP}\left(\mathbf{x}_2, \mathbf{h}_1^{sent}, \mathbf{c}_1^{sent}\right)$$
$$\mathbf{h}_{n_{S_i}}^{sent}, \mathbf{c}_{n_{S_i}}^{sent} = \text{LSTM\_STEP}\left(\mathbf{x}_{n_{S_i}}, \mathbf{h}_{n_{S_i}-1}^{sent}, \mathbf{c}_{n_{S_i}-1}^{sent}\right)$$

Then the last hidden state of sentence-level LSTM can be the representation of document.
(b) Similar to using the attention mechanism on LSTM, we can use different kinds of attention mechanism on token-level and sentence-level LSTM layer separately.

14.5 Let $y = \text{HIGHWAY}(x)$

$$\frac{\partial L}{\partial b^T} = \frac{\partial L}{\partial y} \otimes (g - x) \otimes \frac{\partial \sigma\left(W^T x + b^T\right)}{\partial\left(w^T x + b^T\right)} \cdot x^T$$
$$\frac{\partial L}{\partial b^H} = \frac{\partial L}{\partial y} \otimes t \otimes \frac{\partial \sigma\left(W^H x + b^H\right)}{\partial\left(w^H x + b^H\right)} \cdot x^T$$
$$\frac{\partial L}{\partial b^T} = \frac{\partial L}{\partial y} \otimes (g - x) \otimes \frac{\partial \sigma\left(W^T x + b^T\right)}{\partial\left(w^T x + b^T\right)}$$
$$\frac{\partial L}{\partial b^H} = \frac{\partial L}{\partial y} \otimes t \otimes \frac{\partial \sigma\left(W^H x + b^H\right)}{\partial\left(w^H x + b^H\right)}$$
$$\frac{\partial L}{\partial x} = \left(w^T\right)^T \cdot \left((g - x) \otimes \frac{\partial \sigma\left(W^T x + b^T\right)}{\partial w^T x + b^T}\right) \oplus \left(w^H\right)^T \cdot \left(t \otimes \frac{\partial \sigma\left(w^H x + b^H\right)}{\partial w^H + b^H}\right) \oplus (1 - t)$$

They both carry information from the input to the output directly. However ResNets don't have the extra parameters compared with highway and it tries to create the ideal mapping by using x directly while highway uses a 'gate' mechanism.

14.6 Empirical results have led many to believe that noise added to recurrent layers (connections between RNN units) will be amplified for long sequences, and drown the signal. However, variational dropout in the word-based model corresponds then to randomly dropping word types in the sentence, and might be interpreted as forcing the model not to rely on single words for its task.

14.7 Let x_t as input.

$$\begin{aligned}
\mathbf{Q} &= \text{ LayerNorm } \left(\mathbf{W^Q x_t}; \boldsymbol{\alpha}_1, \boldsymbol{\beta}_1\right) \\
\mathbf{K} &= \text{ LayerNorm } \left(\mathbf{W^K x_t}; \boldsymbol{\alpha}_2, \boldsymbol{\beta}_2\right) \\
\mathbf{V} &= \text{ LayerNorm } \left(\mathbf{W^V x_t}; \boldsymbol{\alpha}_3, \boldsymbol{\beta}_3\right) \\
\mathbf{H} &= \text{attention}\left(\mathbf{Q}, \mathbf{K}, \mathbf{V}\right)
\end{aligned}$$

14.8 dot-product attention

$$\begin{aligned}
\frac{\partial \text{ score } (q,h)}{\partial q} &= h \\
\frac{\partial \text{ score } (q,h)}{\partial h} &= q
\end{aligned}$$

scaled dot-product attention

$$\begin{aligned}
\frac{\partial \text{ score } (q,h)}{\partial q} &= \frac{h}{\sqrt{d}} \\
\frac{\partial \text{ score } (q,h)}{\partial h} &= \frac{q}{\sqrt{d}}
\end{aligned}$$

general attention

$$\begin{aligned}
\frac{\partial \text{ score } (q,h)}{\partial q} &= Wh \\
\frac{\partial \text{ score } (q,h)}{\partial h} &= W^T q \\
\frac{\partial \text{ score } (q,h)}{\partial W} &= qh
\end{aligned}$$

additive attention

$$\begin{aligned}
\frac{\partial \text{score}(q,h)}{\partial v} &= \tanh(W(q \oplus h) + b) \\
\frac{\partial \text{ score } (q,h)}{\partial b} &= v \otimes \left(1 - \tanh(w(q \oplus h) + b)^2\right) \\
\frac{\partial \text{ score } (q,h)}{\partial W} &= (q \oplus h)^T \cdot \left(v \otimes \left(1 - \tanh(w(q \oplus h) + b)^2\right)\right) \\
\frac{\partial \text{ score } (q,h)}{\partial W} &= W^T \cdot \left(v \otimes \left(1 - \tanh(w(q \oplus h) + b)^2\right)\right)
\end{aligned}$$

14.9 We can use three dimension matrix $W\prime$, then $s_i = qW\prime h_i$, which has interaction between each element in vector q and h_i.

14.10 (a)First, we can hard code leaf node by $c = h = emb(x)$. We can also treat leaf node as branch node.

$$\mathbf{h}_1, \mathbf{c}_1 = \text{LSTM}(\mathbf{x}_1, \mathbf{h}_0, \mathbf{c}_0)$$

$\mathbf{h}_0$, $\mathbf{c}_0$ are the initial state and cell vectors, which are model parameters.
(b) Dependency tree is a structure that contain the dependent relation between each word. Child-sum tree LSTM is a model that can be used to represent arbitrary tree, not only dependency tree. The most suitable value for each node to input is the representation of word corresponding to each node.
(c) For unary-branching nodes, we don't change the expression of original binary tree LSTM. Let the only node to be the left node and assign zero values to right node.
(d) Similar to the solution of (c), we creat a N-ary tree LSTM, in which vectors of non-existent nodes are made $\vec{0}$
(e) For each nodes, we only consider its incoming edge and calculate them as same as child-tree LSTM. Because DAG has no circle in graph, we can sequential calculate the values of the nodes and use the values of the child nodes to compute the values of the parent nodes.

14.11 (a) Sure. Replace LSTM by GRU to generate representation of words and remain the other part of GCN model unchanged.
(b) Sure. In the process of operation, node information and edge information are equally considered.
(c) Multi-layer SANs are GATs that each node is connect to other nodes. Multi-layer CNNs have different kernel size in each layer, therefore, we can treat nodes in threshold as neighbor nodes of the node generated by pooling. (d) Add the node itself to the calculation of attention mechanism, and weighted sum neighbour nodes and itself when calculating its hidden state.
(e) At sentence-level, we treat each sentence as a node and define the causal logic, emotion causal and topic similarity between sentences as an edge. For each word in the sentence, we make each word as a node and use word categories, entity labels, referential relationships, and entity relationships as the definition of edges.

14.12 Different kinds of Tree-LSTM can represent constituent trees and dependency trees. For AMR graphs, we can use GNN to generate the representation of them. In order to get a representation that jointly learns all the structure and the original sequence, pooling layer is a good choice. Tree structure depends on original sequence . Empirically, the latter one is better.

14.13 In terms of complexity, AdaMax can be simpler than Adam due to without tracking the quadratic history gradients. In addition, AdaMax does not need to consider the bias correction term. For optimisation effectiveness, AdaMax can sometimes be more effective than Adam for certain applications. But it is difficult

to theoretically compare which one is more effective. In practice, the computation overload of quadratic history gradients can be neglected, and Adam is a much more popular choice than AdaMax.

Chapter 15 Reference Answers

15.1 They differ on output layer. A CRF layer is used to replace a local softmax output layer. Because a CRF output layer captures Markov dependencies between consecutive labels in a sequence, it out-performs a local classification output layer. However, taking dependencies between consecutive labels into consideration corresponds to cost more computation resource.

15.2 The above function can't interact element of h_i and h_j, while biaffine model the interaction between the corresponding elements of h_i and h_j. Also, we can calculate score by $s_{i,j} = \mathbf{v_1}^{\mathbf{T}}\mathbf{h}_i^T\mathbf{W}\mathbf{h}_j + \mathbf{h}_i^T\mathbf{U}\mathbf{h}_j + \mathbf{v_2}^{\mathbf{T}}(\mathbf{h}_i \oplus \mathbf{h}_j)$, where W is three-dimensional matrix. The advantage is each element of h_i and h_j can interact with each other. The disadvantage is more computing resources are required.

15.3 The most useful algorithm is Chu-Liu-Edmond algorithm. The runtime of Prim algorithm and Kruskal algorithm are $O(n\log n + m\log n)$ and the Chu-Liu-Edmond algorithm's is $O(n\log n + m)$, where m is the number of edges and n is the number of nodes. The choice of decoding algorithm affect the training of the parser because different algorithm will lead to different maximum spanning tree.

15.4 Yes, we should adapt word representation into span representation, and we get a sequence of pan-level hidden state. Then the task can be defined as finding the most likely span s_i for each span s_j in the span sequence. The POS tag and dependency syntax are useful for relation extraction. We can use graph neural network to integrate dependency syntax as input feature in the task, where span is node and dependency syntax is edge.

15.5

15.6

15.7 Model 2 is missing arc label feature. I think arc label feature and coreference feature can be added to model 2. Take arc label feature for an example, it can be served as edge in GNN to let two words interact with each other.

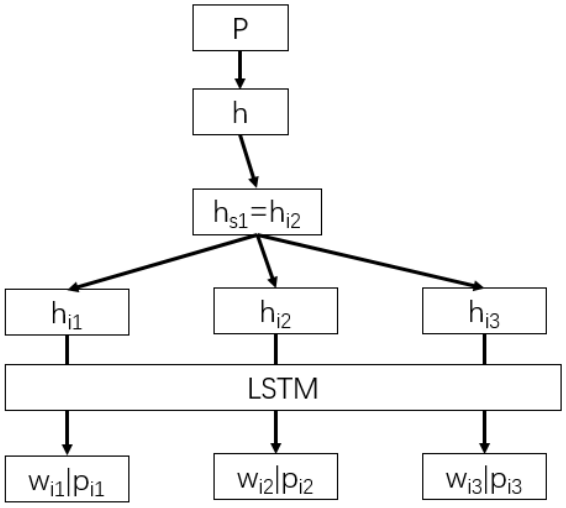


Figure 1.5: Back-propagation of question 15.8

15.8 h integrates hidden representation h_{i_3}, h_{i_2}, h_{i_1} and h_{j_1} for representing s. i_1, i_2, i_3 and j_1 are static. Model 2 use three independent feature to represent stack information and BiLSTM makes each hidden state contain sentence information.

15.9 There are t^2+1 parameter instances in b. We can extend parameter instances of b to $|D|(t^2+1)$, where D is the word number of dictionary.

15.10 Finish with Ch8

15.11

$$\frac{\partial L(W_i,T_i,\Theta)}{\partial \mathbf{b}(\ell,\ell')} = -\sum_{j=1}^{|W_i|}\left(\mathbf{h}_j^i \delta\left(t_j^i=\ell, t_{j-1}^i=\ell'\right) - \mathbb{E}_{T'\sim P(T'|W_i)}\mathbf{h}_j^i \delta\left(t_j^i=\ell, t_{j-1}^i=\ell'\right)\right)$$

15.12

$$\begin{aligned}\frac{\partial L}{\partial \vec{\gamma}(r)} &= -\left(\sum_{r\in T_i}\vec{\tau}(r)\mathbf{W}^{f^T} - \sum_{r'\in \mathrm{GeNR}(W_i)} P\left(r' \mid W_i\right)\vec{\tau}(r')\mathbf{W}^{f^T}\right)\\ &= -\left(\sum_{r\in T_i}\vec{\tau}(r)\mathbf{W}^{f^T} - \mathbb{E}_{r'\sim P(r'|W_i)}\vec{\tau}(r')\mathbf{W}^{f^T}\right)\end{aligned}$$

15.13

$$\begin{aligned}\frac{\partial {p_i}^{arc}}{\partial h_j} &= \frac{\partial {p_i}^{arc}}{\partial {o_i}^{arc}} \cdot \frac{\partial {o_i}^{arc}}{\partial h_j} \\ &= [\text{diag}\,({p_i}^{arc}) - p_i^{arc} \cdot p_i^{arc}]\, vector(\delta(s_{i,j} = s_{i,j'}))\left(U^\top h_i + v\right)\end{aligned}$$

,where $\delta(s_{i,j} = s_{i,j'})$ returns 1 if $s_{i,j} = s_{i,j'}$ is true and 0 otherwise.

$$\begin{aligned}\frac{\partial {p_i}^{arc}}{\partial h_i} &= \frac{\partial {p_i}^{arc}}{\partial {o_i}^{arc}} \cdot \frac{\partial {o_i}^{arc}}{\partial h_i} \\ &= [\text{diag}\,({p_i}^{arc}) - p_i^{arc} \cdot p_i^{arc}]\, vector(\delta(s_{i,j} = s_{i.j'}))\,(Uh_j + v)\end{aligned}$$

15.14 The original vector was A, and then it was A'. So if you use concatenation, then your fully connected network can be represented as a block matrix $[B,C]$, resulting in $BA + CA'$; If you add, then your fully connected network can be represented as a matrix D, resulting in $D(A + A')$. As can be seen from the above formula, the concatenation operation must be able to represent what the addition operation can represent. As long as the neural network learns $B = C = D$, the concatenation operation can get the same result as the addition operation. Conversely, this conclusion is not true. It can work just because the difference between the two embedding is small, they can be mapped into one space.

15.15 We can add up all the word embedding in the span as span representation. In addition, pooling function and attention functions are also available. Empirically, attention functions can give the similar performance because it consider the information between each word in span. Named entity recognition, relation extraction, coreference resolution and event extraction can benefit from span representation.

15.16

Chapter 16 Reference Answers

16.1

$$\lambda = \sigma\left(\mathbf{W}^{\lambda h}\mathbf{h}_{i-1}^{dec} + \mathbf{W}^{\lambda y} emb'(y_{i-1}) + b^{\lambda}\right)$$

$$P(y_i \mid X_{1:n}, Y_{1:i-1}) = \frac{\text{score}_{\ g}(y_i = v_i) + \ \text{score}_{\ c}(y_i = u_k)}{\sum_{v \in V} \ \text{score}_{\ g}(v) + \sum_{u \in U} \text{score}_c(u)}, v_i \in V and u_k \in U$$

, where v_i denotes the ith vocabulary word in V and u_k denotes the kth vocabulary word in U.

16.2 It will be very useful in semantic analysis, because under different context the same sentence has different meaning. Conditional encoding can use attention at every encode step by calculating the context vector and adding is as part of input.

16.3 Yes, multi-layer enables more complex representation of our time-series data, capturing information at different scales and adds levels of abstraction of input observations over time. Each h_i of BiLSTM depends on h_{i-1}. In contrast, for SAN, the attention function for each h_i is independent, which allows strong parallelisation in computation. Therefore, multi-layer BiLSTM will cost more time.

16.4 (a)Yes, We can view target labels as a sequence of label that need to be predicted by sequence-to-sequence model. The advantage is each label is predicted given the previous $T_{1:i-1}$ and word sequence $W_{1:n}$ which is a global answer. The disadvantage is sequence-to-sequence model have to predict all the label given the last hidden state of encoder, which may lost some important information when the sequence of label is long.
(b)Yes. They both fuse many information into a vector to predict structure. Models in Chapter 15 only use some of certain position hidden state to predict and don't use the fusion vector as the input of next step prediction, while sequence-to-sequence model fuses all the sequence information to predict answer and at each step the previous output is used as input.

16.5 (a)The model can not remember long term utterances, thus lose much information. Yes, chitchat and task-oriented dialogue don't require model to use much

history information, which may lead to some noise.
(b)Stacked LSTM and BERT. The former one first get sentence-level representation then fuse history dialogue. we can concatenate dialogue history into a long string and use BERT to get the representation of dialogue history.
(c)Similar to the formal attention mechanism conduct on context words, we view knowledge as context words. Each entity and relation in knowledge base has its own representation. Then we can get the attention distribution and context vectot over knowledge to generat the next word.
(d)Yes. We can view each m_i in memory network as context word representation and use attention mechanism on it. The advantage of doing this is adding extra information that related to the query can improve the performance.

16.6 A sequence-to-sequence neural network encodes the source sentence and generates the target nodes and edges with decoder. This model predicts each node or edge given the previous prediction and source sentence information, which is a greedy local predition. Compared with a greedy local structure, its prediction is generated under more information. Compared with a transaction-based model, it without global normalisation and don't need to define features.

16.7 We define cross entropy as the training objective funtion.

$$\frac{\partial L}{\partial h^1} = (P_{match}(h^1, h^2) - y)\frac{h^1}{|h^1||h^2|}$$
$$\frac{\partial L}{\partial h^2} = (P_{match}(h^1, h^2) - y)\frac{h^2}{|h^1||h^2|}$$

16.8 Average of hidden states contains more complete information of source sentence, therefore, this kind of representation is more precise. We can average representation and use this representation to calculate attention distribution on h_i, which can be used to weigh sum the h_i. The advantage of the aggregation layer is that it contains more important information closed to the whole sentence.

16.9 Co-attention matching used one scoring matrix X for calculating both α_1 and α_2, which makes the attention scores in both direction closely coupled.

16.10 This method contains different level information, which includes words level, weighted sum of h_j^2 and the weighted sum of most important words in H^i with respect to the other h^j. This makes the representation are contains more information.

16.11 Bilinear attention has interaction between corresponding element between ${h_i}^1$ and ${h_j}^2$. Dot-product attention doesn't need extra parameters. Additive attention adds non-linear function, which can model more complex relationship.

16.12

Chapter 17 Reference Answers

17.1 Yes. RNNs can be used for n-gram neural language modelling(describe the reason). (using the character of recurrent neural language model to answer)

17.2 hierarchical softmax discussed in Section 17.1.3

17.3 The parent node and children nodes can be defined as the context words.

17.4 The advantage is that the context words with their relative position have more information and more accurate. The disadvantage is enlarging the table of the context embedding, which will increasing parameter size. In addition, in test phrase, it will be more OOV words occurred.

17.5 (a) Word embedding is the representation of a word, therefore, dot product can be performed between two word embeddings to calculate the similarity of two words.
(b) In the space of word embedding, words of the same category are close to each other. Therefore, we can calculate the distance between a group of word embeddings to find the outlier.

17.6

$$\mu_a = E[a] = \frac{0.55+0.65+0.8+0.2+0.99+0.1}{6} = 0.548$$
$$\mu_b = E[b] = \frac{0.52+0.43+0.75+0.1+0.875+0}{6} = 0.446$$
$$\mu_c = E[c] = \frac{0.6+0.7+2.3+0.1+0.99+0.22}{6} = 0.485$$
$$\sigma_a = \sqrt{E[a^2] - E[a]^2} = 0.315$$
$$\sigma_b = 0.316$$
$$\sigma_c = 0.307$$
$$\rho(a,b) = \frac{E[ab] - E[a]E[b]}{\sigma_a \sigma_b} = 0.98$$
$$\rho(a,c) = 0.78$$

17.7 No. Because sentences cannot be enumerated, which leads to severe OOV. In addition, each sentence appears only a few times in the corpus. Therefore, our model has difficulty building lookup table from the data. We can concatenate the result of pooling function over a set of vectors with the last hidden state of sequence model to represent a sentence.

17.8 Yes. After pre-training each word is well represented, however the performance of those model for directly calculating word similarities and detecting analogy empirically bad, because those contextualised word embeddings is used under the condition that has context, not a single word.

17.9 We can freeze the parameter of those model, and using the contextualised embeddings on dependency parsing, semantic role labelling and word sense disambiguation. Compare the performance of those model with Glove and current baseline of this tasks.

17.10 We can stack the encode of transformers as the model encoder. For the local output layers we can use the output layer of local transition-based Model 2 in chapter 15 to combine feature vector and make prediction.

17.11 Naive multi-task learning shares a set of common model parameter, which means such a method doesn't has specific model parameter for different tasks. However, parameter generation network can generate task-specific parameter without losing the mutual information of different tasks.

17.12 TO BE DISCUSSED

17.13 See Algorithm 15

Algorithm 15: Dynamic masking in 17.13

Inputs: $D = \{d_i\}|_{i=1}^{m}$ denotes training instances, $d^i = \{s_j\}|_{j=1}^{N_i}$;
for $d_i \in D$ **do**
  for $s_j \in d_i$ **do**
    RANDOMMASK(s_j)
for $i' \in [1, \ldots, m]$ **do**
  PRETRAIN(d_i')

Chapter 18 Reference Answers

18.1 (a)

$$\log P(Y \mid X) = \log \Sigma_Z e^{W^\top f(X,Y,Z;\theta)} - \log \sum_{Y'} \sum_{Z'} e^{W^\top f(X,Y',Z';\theta)}$$

$$\begin{aligned}
\frac{\partial \log P}{\partial \theta} &= \frac{\sum_Z \frac{\partial e^{W^\top f(X,Y,Z;\theta)}}{\partial \theta}}{\sum_Z e^{W^\top f(X,Y,Z;\theta)}} - \frac{\sum_{Y'} \sum_{Z'} \frac{\partial e^{W^\top f(X,Y',Z';\theta)}}{\partial \theta}}{\sum_{Y'} \sum_{Z'} e^{W^\top f(X,Y',Z';\theta)}} \\
&= \frac{\sum_Z e^{W^\top f(X,Y,Z;\theta)} W^\top \frac{\partial f}{\partial \theta}}{\sum_Z e^{W^\top f(X,Y,Z;\theta)}} - \frac{\sum_{Y'} \sum_{Z'} e^{W^\top} f(X,Y',Z';\theta) \frac{\partial f}{\partial \theta}}{\sum_{Y'} \sum_{Z'} e^{W^\top f(X,Y',Z';\theta)}}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial \log W}{\partial W} &= \frac{\sum_Z \frac{\partial e^{W^\top f(X,Y,Z;\theta)}}{\partial}}{\sum_Z e^{W^\top f(X,Y,Z;\theta)}} - \frac{\sum_{Y'} \sum_{Z'} \frac{\partial e^{W^\top f(X,Y',Z';\theta)}}{\partial W}}{\sum_{Y'} \sum_{Z'} e^{W^\top f(X,Y',Z';\theta)}} \\
&= \frac{\sum_Z e^{W^\top f(X,Y,Z;\theta)} f(X,Y,Z;\theta)}{\sum_Z e^{W^\top f(X,Y,Z;\theta)}} - \frac{\sum_{Y'} \sum_{Z'} e^{W^\top} f(X,Y',Z';\theta) f(X,Y',Z';\theta)}{\sum_{Y'} \sum_{Z'} e^{W^\top f(X,Y',Z';\theta)}}
\end{aligned}$$

(b) Forward-backward algorithm gives the marginal probability of each state, and Viterbi algorithm gives the probability of the most likely sequence of states.
(c) We define the label sequence score $F(y \mid x, \Theta)$ as the maximum score among its latent sequence:

$$F(y \mid x, \Theta) = \max_{h:\text{Proj}(h)=y} F(h \mid x, \Theta),$$

where $\text{Proj}(h)$ is the projection from a latent sequence h to a label sequence y :

$$\text{Proj}(h) = y \Longleftrightarrow h_j \in H_{y_j} \text{ for } j = 1, \ldots, m,$$

and $F(h \mid x, \Theta)$ is the score of a latent sequence:

$$F(h \mid x, \Theta) = \Theta \cdot f(h,x) = \sum_k \Theta_k \cdot f_k(h,x),$$

where f_k denotes the k-th value of the global feature vector.

18.2 Probability function is represented by the softmax function. The parameters of HMM are represented by neural network parameters. We use two neural modules to represent transition model and emission model respectively. Then calculate gradient descent algorithm and use softmax function to update model parameters.

18.3

18.4 We can represent each subsequent using the output of BiLSTM in each step, which contains global information from two directions. The advantage is that it contains more information instead of information from previous steps.

18.5

18.6 First, we can use embedding matrix to map each words in span $X_{i:j}$ to vectors. Then we have several methods to integrate information in vector to one vector such as pooling function ,average sum and weight sum with attention mechanism.

18.7