Exam 2018-08-31

Marco Kuhlmann

This exam consists of three parts:

1. **Part A** consists of 5 items, each worth 3 points. These items test your understanding of the basic algorithms that are covered in the course. They require only compact answers, such as a short text, calculation, or diagram.

   Collected wildcards are valid for this part of the exam. The numbering of the questions corresponds to the numbering of the wildcards.

2. **Part B** consists of 3 items, each worth 6 points. These items test your understanding of the more advanced algorithms that are covered in the course. They require detailed and coherent answers with correct terminology.

3. **Part C** consists of 1 item worth 9 points. This item tests your understanding of algorithms that have not been explicitly covered in the course. This item requires a detailed and coherent answer with correct terminology.

Grade requirements: For grade 3, you need at least 12 points in Part A (some of which may come from wildcards). For grade 4, you additionally need at least 14 points in Part B. For grade 5, you additionally need at least 7 points in Part C.

Note that surplus points in one part do not raise your score in another part.

Good luck!

Part A

01 **Text classification** **(3 points)**

a) Complete the decision rule for the Naive Bayes classifier under the assumption that the classifier uses log probabilities. Explain your notation.

$$\hat{c} = \arg\max_{c \in C} \ldots$$

b) Complete the learning algorithm for the multi-class perceptron:

for each class c **do**
  $\boldsymbol{w}_c \leftarrow \mathbf{0}$
for each epoch **do**
  for each training example $(\boldsymbol{x}, y)$ **do**
    ...

c) You train two multi-class perceptrons by running the above learning algorithm for 100 epochs on the two data sets shown below. Which model will have perfect accuracy when evaluated on its training set, and why? Answer with a short text.

data set A:

$\boldsymbol{x}$	y
(0, 1)	pos
(1, 0)	neg
(1, 2)	pos
(2, 1)	neg

data set B:

$\boldsymbol{x}$	y
(0, 1)	pos
(1, 0)	neg
(1, 2)	neg
(2, 1)	pos

02 **Language modelling** **(3 points)**

The Corpus of Contemporary American English (COCA) is the largest freely-available corpus of English, containing approximately 560 million tokens. In this corpus we have the following counts of unigrams and bigrams:

snow	*white*	*white snow*	*purple*	*purple snow*
38,186	256,091	122	11,218	0

a) Estimate the following probabilities using maximum likelihood estimation without smoothing. Answer with fractions containing concrete numbers. You do not have to simplify the fractions.

i. $P(\textit{snow})$

ii. $P(\textit{snow} \mid \textit{purple})$

b) Estimate the following probabilities using absolute discounting with $d = 0.2$. Assume that the vocabulary consists of 1,254,193 unique words, that each word from the vocabulary is observed at least once, and that the number of unique words observed after the word *purple* is 2,462. Answer with fractions containing concrete numbers. You do not have to simplify the fractions.

i. $P(\textit{snow})$

ii. $P(\textit{snow} \mid \textit{purple})$

c) We use maximum likelihood estimation with add-k smoothing to train n-gram models on the COCA corpus, with $n \in \{1, \ldots, 5\}$ and $k \in \{0, 0.1, 1\}$. The following table shows the entropy of each trained model on the training data. Which row corresponds to which k-value, and why? Answer with a short text.

	$n = 1$	$n = 2$	$n = 3$	$n = 4$	$n = 5$
$k = a$	7.3285	5.9834	6.7332	6.9556	7.0555
$k = b$	7.3376	3.4269	1.4290	0.5436	0.4171
$k = c$	7.2732	7.3837	8.4573	8.6577	8.7350

03 **Part-of-speech tagging** **(3 points)**

The following matrices specify a hidden Markov model in terms of costs (negative log probabilities). The marked cell gives the transition cost from BOS to AB.

	AB	PN	PP	VB	EOS
BOS	11	10	12	11	25
AB	11	11	11	10	14
PN	11	12	12	10	16
PP	13	11	12	14	18
VB	11	10	10	13	15

	she	got	up
AB	25	25	14
PN	13	25	25
PP	25	25	13
VB	25	14	19

When using the Viterbi algorithm to calculate the least expensive (most probable) tag sequence for the sentence 'she got up' according to this model, we get the following matrix. Note that the matrix is missing three values (marked cells).

		she	got	up
BOS	0			
AB		36	59	72
PN		23	60	82
PP		A	60	70
VB		36	47	B
EOS				C

a) Calculate the missing values.

b) State the runtime complexity of the Viterbi algorithm as a function of the number of tags in the model, m, and the number of words in the sentence, n. Use Big O notation. Provide a short explanation of your analysis.

c) Which of the following features go beyond what can be expressed in a hidden Markov model? Answer with a short text.

i. current word

ii. word to the left of the current word

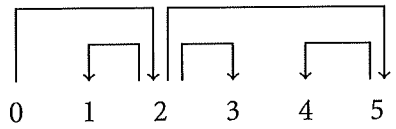
iii. word to the right of the current word

iv. part-of-speech tag of the word to the left of the current word

04 Syntactic analysis (3 points)

a) Draw all projective dependency trees for the sentence '1 2 3' in which the artificial root vertex 0 has exactly one child.

b) State a sequence of transitions that make an arc-standard parser produce the following dependency tree:



c) To train a transition-based dependency parser, we need a *training oracle* that tells us the gold-standard transition sequence for a tree in the training data. State the condition under which this oracle chooses the transition 'right arc'.

05 Semantic analysis (3 points)

a) Consider the following co-occurrence matrix over the six-word vocabulary *coffee*, *tea*, *beans*, *bubble*, *cocoa*, *taxi*. Rows correspond to target words, columns correspond to context words. All counts that are not explicitly given are zero.

	beans	bubble	cocoa	taxi
coffee	35	0	11	0
tea	1	17	4	0

List all target word–context word pairs (w, c) whose positive pointwise mutual information is (strictly) greater than zero.

b) If we can represent words as vectors, then we can measure word similarity as the distance between the vectors. One way to do this is to compute the dot product of the two vectors. What potential problems does this metric have?

c) In a certain vector space model, the two nearest neighbours of the word *fast* are *quick* and *slow*. Explain why this is an expected result. Answer with a short text.

Part B

06 **Levenshtein distance** **(6 points)**

The following matrix shows the values computed by the Wagner–Fischer algorithm for finding the Levenshtein distance between the two words *student* and *teacher*. Note that the matrix is missing a value (marked cell).

t	7	6	5	5	5	6	6	7
n	6	5	4	4	5	5	6	6
e	5	4	3	4	4	5	5	6
d	4	3	3		4	5	6	7
u	3	2	2	3	4	5	6	7
t	2	1	2	3	4	5	6	7
s	1	1	2	3	4	5	6	7
#	0	1	2	3	4	5	6	7
	#	t	e	a	c	h	e	r

a) Define the concept of the Levenshtein distance between two words. The definition should be understandable even to readers who have not taken this course.

b) Calculate the value for the marked cell. Explain your calculation in detail. Show that you have understood the Wagner–Fischer algorithm.

c) It is much more likely for a user to mistype the word *student* as *stusent* than as *stulent*; this is because the keys for the letters *d* and *s* are much closer to each other on the keyboard than the keys for the letters *d* and *l*. Explain how the Wagner–Fischer algorithm could be adapted to take this information into account.

07 **Eisner algorithm** **(6 points)**

Here is incomplete pseudocode for the Eisner algorithm:

for i **in** $[0, \ldots, n]$:
    // two lines missing
for k **in** $[1, \ldots, n]$:
    for i **in** $[k-1, \ldots, 0]$:
        $T_4[i][k] = \max_{i \leq j < k} \left(T_2[i][j] + T_1[j+1][k] + A[i][k]\right)$
        // three lines missing

The table A holds the arc-specific scores. The tables T_t hold values from the set $\mathbb{R} \cup \{-\infty\}$ and correspond to the four different types of subproblems:

type 1 type 2 type 3 type 4

These tables are initialised with the value $-\infty$.

a) Complete the missing lines.

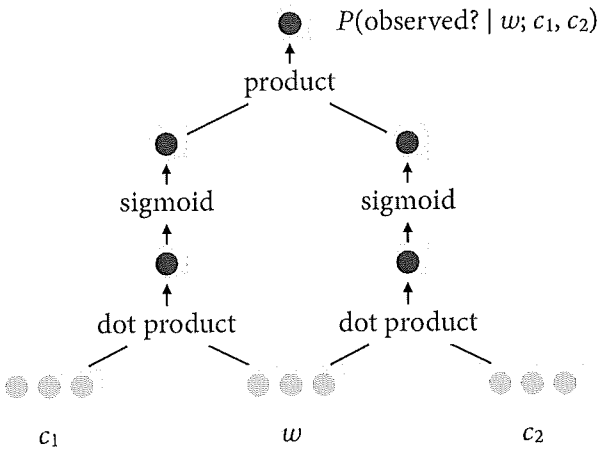
b) The Eisner algorithm can be viewed as a more effective version of the Collins algorithm. Give pseudocode for the Collins algorithm. State the runtime complexity of the Collins algorithm and the Eisner algorithm as a function of the number of words in the sentence. Use Big O notation.

c) The Eisner algorithm is typically used for arc-factored parsing. Under this model, the score of a candidate dependency tree y for a (part-of-speech-tagged) sentence x is defined as the sum of the scores of the arcs of the tree, and the score of an arc $h \rightarrow d$ is defined as the dot product of a feature vector $\phi(x, h \rightarrow d)$ and a weight vector $\boldsymbol{w}$. Give pseudocode for the structured perceptron algorithm that is used to learn this weight vector. Give some examples of features that may be useful in this context.

08 **word2vec** **(6 points)**

Google's word2vec implements two separate algorithms for training word embeddings: *continuous bag-of-words* and *skip-gram*. Both algorithms obtain word embeddings as 'side products' of a binary prediction task.

a) Explain the architecture of the skip-gram model in your own words. Refer to the relevant slide from the lecture, reproduced below.



b) To train a binary classifier, one needs both positive and negative instances. Explain how this data is obtained for the skip-gram model.

c) Levy and Goldberg (2014) showed a close connection between the skip-gram model and the method of obtaining word embeddings from a co-occurrence matrix that was covered in the course. Explain this connection in your own words. Why would you prefer the skip-gram model over the matrix approach?

Part C

09 **Odds and ends of dependency parsing** **(9 points)**

For a) and b), consider dependency trees without an artificial root vertex, that is, trees where the root vertex may take any position between 1 and n.

a) Every projective dependency tree can be constructed by some sequence of transitions in the arc-standard system, but this sequence is not necessarily unique – one and the same tree may be constructed in several different ways.

i. Provide an example that illustrates this point.

ii. The number of projective dependency trees on n vertices is given by the integer sequence 1, 2, 7, 30, 143, ... How many different transition sequences are there for projective dependency trees on n vertices, for $1 \leq n \leq 5$?

b) The *arc-hybrid* system has the same configurations and the same initialisation and termination conditions as the arc-standard system, but uses a different LEFT transition. Let us denote a configuration as $c = (\sigma, \beta, A)$, where σ is the stack, β is the buffer, and A is the set of already constructed dependency arcs. Then the three transitions of the arc-hybrid system can be defined as follows:

$$(\sigma, b|\beta, A) \rightarrow (\sigma|b, \beta, A) \qquad \text{SHIFT}$$
$$(\sigma|s_1|s_0, \beta, A) \rightarrow (\sigma|s_1, \beta, A \cup \{(s_1, s_0)\}) \qquad \text{RIGHT}$$
$$(\sigma|s, b|\beta, A) \rightarrow (\sigma, b|\beta, A \cup \{(b, s)\}) \qquad \text{LEFT}$$

i. State a sequence of transitions that make an arc-hybrid parser produce the dependency tree from item 04 b).

ii. How many transitions does an arc-hybrid parser make when processing a sentence with n words? State your answer as a function of n.

c) The Eisner algorithm can be simplified by replacing subproblems of type 3 and subproblems of type 4 with one new type of subproblem, which we may refer to as type 5. To compute the score of subproblems of this type, we combine two 'triangles' without charging the score for an arc:

$$T_5[i][k] = \max_{i \leq j < k} \left(T_2[i][j] + T_1[j+1][k] \right)$$

How do you have to modify the other rules of the Eisner algorithm such that the modified algorithm becomes equivalent to the original one? Argue for the correctness of the modified algorithm.