

Information page for written examinations at Linköping University



Examination date	2019-06-12
Room (1)	<u>TER2(6)</u>
Time	14-18
Edu. code	TDDE09
Module	TEN1
Edu. code name Module name	Natural Language Processing (Språkteknologi) Written examination (Skriftlig tentamen)
Department	IDA
Number of questions in the examination	8
Teacher responsible/contact person during the exam time	Marco Kuhlmann
Contact number during the exam time	013-284644
Visit to the examination room approximately	endast telefonjour
Name and contact details to the course administrator (name + phone nr + mail)	Veronica Kindeland Gunnarsson, 013-285634, veronica.kindeland.gunnarsson@liu.se
Equipment permitted	inga
Other important information	
Number of exams in the bag	

Exam 2019-06-12

Marco Kuhlmann

This exam consists of two parts:

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.

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.

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

Grade requirements:

- Grade 3: at least 12 points in Part A
- Grade 4: at least 12 points in Part A and at least 7 points in Part B
- Grade 5: at least 12 points in Part A and at least 14 points in Part B

Good luck!

Part A

01 Text classification

(3 points)

- a) The evaluation of a text classifier produced the following confusion matrix. The marked cell gives the number of times the system classified a document as class A whereas the gold-standard class for the document was B.

	A	B	C
A	58	6	1
B	5	11	2
C	0	7	43

Set up fractions for the following values:

- i. precision with respect to class A
 - ii. recall with respect to class B
- b) Here is the learning algorithm for the un-averaged multi-class perceptron. For the sake of simplicity, we ignore the bias term.

```
1  for each class  $c$  do
2     $w_c \leftarrow 0$ 
3  for each epoch  $e$  do
4    for each training example  $(x, y)$  do
5       $p \leftarrow \arg \max_c xw_c$ 
6      if  $p \neq y$  do
7         $w_p \leftarrow w_p - x$ 
8         $w_y \leftarrow w_y + x$ 
```

State the changes that you have to make to this algorithm to get the learning algorithm for the *averaged* multi-class perceptron.

Part c) on the next page

- c) Here is the model at the core of a logistic regression classifier:

$$\hat{y} = \text{softmax}(xW + b)$$

Suppose that the classifier has been trained on a sentiment analysis data set. The data set consists of 8,544 annotated sentences extracted from movie reviews; each sentence has been annotated as either 'positive', 'negative', or 'neutral' towards the movie at hand. The classifier's vocabulary consists of 16,361 unique words, including a special 'word' for out-of-vocabulary items.

State the following:

- i. number of columns of x
- ii. number of columns of W

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:

<i>snow</i>	<i>white</i>	<i>white snow</i>	<i>purple</i>	<i>purple snow</i>
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.
 - i. $P(\textit{white})$
 - ii. $P(\textit{snow} \mid \textit{white})$

- b) Estimate the following probabilities using absolute discounting with $d = 0.01$. Assume that the vocabulary consists of 1,254,100 unique words, that 99% of these words are 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.
 - i. $P(\textit{snow})$
 - ii. $P(\textit{snow} \mid \textit{purple})$

Part c) on the next page

- c) We use maximum likelihood estimation with add- k smoothing to train n -gram models on the COCA corpus, with $n \in \{1, \dots, 5\}$ 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.

	$n = 1$	$n = 2$	$n = 3$	$n = 4$	$n = 5$
row 1	7.3376	5.9834	6.7332	6.9556	7.0555
row 2	7.3376	3.4269	1.4290	0.5436	0.4171
row 3	7.3376	7.3837	8.4573	8.6577	8.7350

03 Part-of-speech tagging

(3 points)

Here is a Hidden Markov model (HMM) specified in terms of costs (negative log probabilities). The marked cell gives the transition cost from BOS to PL.

	PL	PN	PP	VB	EOS
BOS	11	2	3	4	19
PL	17	3	2	5	7
PN	5	4	3	1	8
PP	12	4	6	7	9
VB	3	2	3	3	7

	they	freak	out
PL	17	17	4
PN	3	19	19
PP	19	19	3
VB	19	8	19

- a) Compute the cost (negative log probability) of the following tagged sentence:

they freak out
 PN VB PL

Parts b) and c) on the next page

- b) When using the Viterbi algorithm to calculate the least expensive (most probable) tag sequence for the same sentence, one gets the following matrix. Calculate the missing values (marked cells).

		they	freak	out	
BOS	o				
PL		28	27	A	
PN		5	28	35	
PP		22	27	B	
VB		23	14	36	
EOS					C

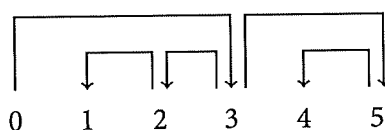
- c) The following table shows a comparison between hidden Markov models and multiclass perceptrons. Complete the three missing cells in this comparison.

Hidden Markov model	Multiclass Perceptron
A	uses weights (arbitrary real numbers)
exhaustive search for the best sequence	B
features: current word, previous word's tag	C

04 Syntactic analysis

(3 points)

- a) Draw all dependency trees for the sentence '1 2 3' in which the artificial root vertex 0 has exactly one child. Mark the trees that are non-projective.
- b) State the following asymptotic complexities for the Collins algorithm, measured in terms of the number of words in the sentence, n .
 - i. runtime requirement
 - ii. memory requirement
- c) State a sequence of transitions that makes the transition-based dependency parser produce the following dependency tree:



05 Semantic analysis

(3 points)

- a) Consider the following co-occurrence matrix. Rows correspond to target words, columns correspond to context words.

	butter	cake	school	cow	deer
cheese	12	2	0	1	0
bread	5	5	0	0	0
goat	0	0	0	6	1
sheep	0	0	0	7	5

Set up fractions for the following PPMI values:

- i. $PPMI(\text{cheese}, \text{cow})$
 - ii. $PPMI(\text{goat}, \text{cow})$
- b) The cosine similarity between two word vectors is based on the dot product of these vectors. Explain why one does not use the dot product directly. Answer with a short text.
 - c) In a certain set of word embeddings, the cosine similarity between *black* and *sheep* is greater than that between *white* and *sheep*. Explain why this may have happened. Answer with a short text.

Part B

06

Levenshtein distance

(6 points)

- 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) Compute the Levenshtein distance between the two words *game* and *lake* using the Wagner–Fischer algorithm. Your answer should contain both the distance itself and the complete matrix.
- c) Jurafsky and Martin (2018) explain how to augment the Wagner–Fischer algorithm to store backpointers in each cell. Add these backpointers to your matrix. Explain why cells may have several backpointers.

07

word2vec

(6 points)

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

- a) Explain the skip-gram model in your own words. What prediction task does it target? How does solving this task yield word embeddings?
- 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) show 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?

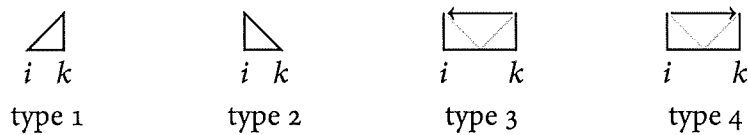
Here is incomplete pseudocode for the Eisner algorithm:

```

for  $i$  in  $[0, \dots, n]$ :
    // two lines missing
for  $k$  in  $[1, \dots, n]$ :
    for  $i$  in  $[k - 1, \dots, 0]$ :
         $T_3[i][k] = \max_{i \leq j < k} (T_2[i][j] + T_1[j + 1][k] + A[k][i])$ 
        // 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:



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

- a) Complete the missing lines.
- b) Provide pseudocode for the structured perceptron algorithm for learning a dependency parser under an arc-factored model. Explain your notation. Make it clear how the Eisner algorithm is used in structured perceptron training.
- 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:

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

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