Försättsblad till skriftlig tentamen vid Linköpings universitet

Datum för tentamen	2019-08-30
Sal (1)	TER3(4)
Tid	14-18
Utb. kod	TDDE09
Modul	TEN1
Utb. kodnamn/benämning **Modulnamn/benämning**	Språkteknologi Skriftlig tentamen
Institution	IDA
Antal uppgifter som ingår i tentamen	8
Jour/Kursansvarig Ange vem som besöker salen	Marco Kuhlmann
Telefon under skrivtiden	013-284644
Besöker salen ca klockan	endast telefonjour
Kursadministratör/kontaktperson (namn + tfnr + mailaddress)	Veronica Kindeland Gunnarsson, 013-285634, veronica.kindeland.gunnarsson@liu.se
Tillåtna hjälpmedel	inga
Övrigt	
Antal exemplar i påsen	

Exam 2019-08-30

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 B

ii. recall with respect to class C

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 ← 0
3  for each epoch e do
4    for each training example (x, y) do
5      p ← arg max_c x w_c
6      if p ≠ y do
7        w_p ← w_p − x
8        w_y ← 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) A logistic regression 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 with a rating from 1 (least positive towards the movie at hand) to 5 (most positive). The classifier's vocabulary consists of 16,361 unique words, including a special 'word' for out-of-vocabulary items. Here is the model at the core of the classifier:

$$\hat{y} = \text{softmax}(\boldsymbol{x}\boldsymbol{W} + \boldsymbol{b})$$

State the following:

i. number of columns of $\boldsymbol{x}$

ii. number of columns of $\boldsymbol{W}$

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:

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.

i. $P(snow)$ ii. $P(snow \mid purple)$

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. $P(snow)$ ii. $P(snow \mid purple)$

Part c) on the next page

c) We use maximum likelihood estimation with add-k smoothing to train two trigram models on the COCA corpus: model A with $k = 0.1$, model B with $k = 0.01$. We compute the entropy of both models on the training data. Which model has the higher entropy, and why? Anwer with a short text.

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 costs (negative log probabilities) of the following two candidate tag sequences for the input sentence 'they freak out'.

PN VB PP 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	0			
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	scores candidate solutions using weights (real numbers)
B	linear-time greedy search with locally optimal decisions
features: current word, previous word's tag	C

04 Syntactic analysis (3 points)

a) 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

b) Draw the dependency tree generated by a transition-based dependency parser after executing the following sequence of transitions:

SH SH SH SH RA SH SH RA RA LA RA

c) The initial configuration of a transition-based dependency parser has an empty stack, a buffer containing all n words of the input sentence, and an empty set of dependency arcs. How many transitions does the parser make to reach a final configuration? State your answer as a function of n.

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(sheep, butter) ii. PPMI(goat, cow)

b) Explain why the cosine similarity of two word vectors read off from a PPMI matrix is never negative. Answer with a short text.

c) In a certain set of word embeddings, the cosine similarity between *black* and *sheep* is higher 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?

08 **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** $[k-1, \ldots, 0]$:
        $T_3[i][k] = \max_{i \leq j < k} \left(T_2[i][j] + T_1[j+1][k] + A[k][i]\right)$
        // 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) 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:

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

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.