Assignment 4

Information Retrieval and Text Mining 20/21

Publication: 2021-01-12
Submission Deadline: 2021-01-21
Discussion Video Online: 2021-01-28
Live Q&A Session: 2021-01-28

Roman Klinger
Valentino Sabbatino, Tordis Daum, Tobias Schmid

- **Groups:** Working in groups of up to three people is encouraged, up to four people is allowed. More people are not allowed. Copying results from one group to another (or from elsewhere) is not allowed. Changing groups during the term is allowed.
- **Grading:** Passing the assignments is a requirement for participation in the exam in all modules IRTM can be part of. Altogether 80 points need to be reached. There are five assignments with 20 pen & paper points and 10 programming points each. That means, altogether, 150 points can be reached.
- **Submission:** First make a group in Ilias, then submit the PDF. Write all group members on the first page of the PDF. Only submit *one* PDF file. If you are technically not able to make a group (it seems that happens on Ilias from time to time), do not submit a PDF multiple times by multiple people – only submit it once. Submission for the programming tasks should also be in the same PDF.
- **Make it understandable:** Do the best you can such that we can understand what you mean. Explain your solutions, comment your code. Print the code in a readable format, write your solutions in a way we can read them.
- **Handwriting:** We received some submissions which were handwritten. That is fine, but they were sometimes barely readable. If you submit handwritten solutions, make sure that they are well organized, easy to read and understand, and that there is not doubt about the interpretation of letters. If you think that this might be hard, please typeset the solutions. We might reduce points if it's really tough for us.

Task 1 (Naïve Bayes) 7 points

Train a Naïve Bayes (the version of the model which we discussed in class) given the following documents annotated with classes c_1 and c_2. Use Add-One-Smoothing. Provide all parameters for a full model specification.

c_1 "happy new year"

c_1 "happy holiday"

c_1 "new year"

c_2 "term starts"

c_2 "work starts"

Given the document

- "happy new year celebrations"

Which class is assigned by the model?

Task 2 (Maximum Entropy Classification) 8 points

Given the following features (without making a difference between upper and lower case) and documents:

weight	feature
$\lambda_1 = 0.2$	$f_1(y,x) = 1$ if "$" in x and y = SPAM
$\lambda_2 = -0.1$	$f_2(y,x) = 1$ if "$" in x and y = HAM
$\lambda_3 = 0.5$	$f_3(y,x) = 1$ if "Nigerian" in x and y = SPAM
$\lambda_4 = -0.2$	$f_4(y,x) = 1$ if "Nigerian" in x and y = HAM
$\lambda_5 = -0.1$	$f_5(y,x) = 1$ if "you" in x and y = SPAM
$\lambda_6 = 0.4$	$f_6(y,x) = 1$ if "you" in x and y = HAM
$\lambda_7 = 0.1$	$f_7(y,x) = 1$ if y = SPAM
$\lambda_8 = 0.0$	$f_7(y,x) = 1$ if y = HAM

Class y	document
SPAM	x_1 = $1 million from Nigerian defense minister
SPAM	x_2 = Please contact Nigerian finance minister
SPAM	x_3 = You won $30,000!
SPAM	x_4 = Buy these Ginsu knifes now.
HAM	x_5 = You should send the Nigerian wildlife report.
HAM	x_6 = Thanks for great dinner. I owe you $20.

Subtask 2.1, 4 points

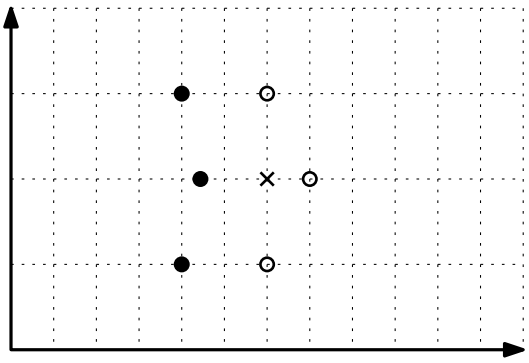
Calculate $p(\text{SPAM}|x_1)$ with this given configuration of a maximum entropy classifier with the specified features and weights?

Subtask 2.2, 4 points

Calculate the partial derivative of the log-likelihood of all documents with respect to λ_6!

Task 3 (kNN Classification) 5 points

Given the following instances in a vector space:



For which $k \in \{1, 3, 5\}$ in a kNN classifier do you obtain the lowest classification confidence for the cross? You can interpret the probability of the assigned class as a confidence value of the classifier's decision. Explain your solution and provide the probabilities for the prediction for the cross for the three different values of k.

Programming Task 4 (10 points)

The assignment data contains two files `games-train.csv` and `games-test.csv`. These are German app reviews for games (a subset of the data described in `http://www.lrec-conf.org/proceedings/lrec2016/pdf/59_Paper.pdf`).

The files are formatted as follows:

- Column 1: Title of game
- Column 2: Class of review (good or bad)
- Column 3: Title of review
- Column 4: Review text

Title and review texts can be empty.

Subtask 1, 10 points

Implement a perceptron learning algorithm to obtain a linear classifier (from scratch, but you can use existing code from the other assignments or libraries for tokenization and preprocessing) which predicts the class (good, bad) stated in column 2. You can use all information from the training file to build your classifier. You are free in chosing the meta-parameters (smoothing, stop-word deletion, stemming, preprocessing) or optimizing those on validation data/via cross validation.

You could implement this as follows (or differently, whatever you prefer):

- Create a Hashmap (Java) or Dictionary (Python) which takes as a key a term and stores the weight of the perceptron weight vector as a value. That means, the weight vector would be stored in a sparse representation (term to weight mapping, instead of listing all weights in a particular order).
- Each instance is represented as an (unsorted) list of features (e.g., words) that hold in an instance. I propose you always put one fixed "word", which is unlikely to occur in the data (e.g., THETA_FEATURE), into this feature list. The associated weight corresponds to θ as we discussed it in class and you don't need to handle θ additionally to the other parameters .
- Calculating the score of the perceptron means: receive all features that represent an instance and sum up the weights for these features as they are available in the Hashmap/Dictionary mentioned above.
- Adapting the weights in the perceptron means: adapting those weights in the hashmap/dictionary (adding 1 or -1) which are listed in the instance currently under consideration.

As usual, submit your code, well-commented and with an explanation. Which terms have the highest importance, according to the model? List the 100 terms with highest weight in each class together with the term weights.

Bonus: 10 points

Implement an evaluation system to Subtask 1 and apply it on `games-test.csv` What is your precision, recall, and F to predict the class good and what is your precision, recall, and F to predict the class bad? Also report the numbers of TP, FP, FN. Discuss your results. Is your result a good result?

Look at some wrongly classified instances and try to understand why they have been wrongly classified (and show them in the submission). Could you come up with ideas how to improve your model, based on these issues?