Advanced Methods in Text Mining
Summer Semester 2023
Assignment 1 – NLP Basics

Instructor: Dr. Rafet Sifa, Tobias Deußer
University of Bonn

Introduction

In this assignment, we will explore some basic concepts of Natural Language Processing (NLP). This assignment will have 3 main parts totalling 100 points (pts.). The coding exercises will be implemented using the programming language python 3.9.

1. Preprocessing

In the lecture, we talked about various ways of preprocessing textual data. In here, we discuss different concepts and how they help the performance in later NLP tasks.

1.1 Concepts (10 Pts.)

Explain what the following text preprocessing concepts are, how they function, and why they are useful.

1. Sentence tokenization
2. Word tokenization
3. Part-of-speech (POS) tagging
4. Lemmatization
5. Stop word removal

1.2 Implementation (10 Pts.)

Build your own NLP preprocessing routine (a function named `process_paragraph(paragraph)`) that takes a paragraph as input, and splits the paragraph into sentences, applies word tokenization and lemmatization on all words. Furthermore, it should remove stop words from the input sentence. The function should return a list containing the processed sentences. You can use any open source package you like for this task, as long as each step is well documented.

2. Logistic Regression

Suppose we have 5,000 E-Mails in our dataset, of which 1,000 are spam E-Mails and 4,000 are non-spam E-Mails. Furthermore, you have two additional binary features for each E-Mail. Table 1 shows the number of occurrences for the two labels and the three possible features a E-Mail can have in this setup, i.e. `Colleague` for an E-Mail sent by a colleague, `Contacted` for an E-Mail from someone outside the company but contacted previously, and `Neither` if the sender of the E-Mail is neither a colleague nor someone previously contacted.

Spam	Colleague	Contacted	Neither	Total
True	15	59	926	1000
False	1029	1418	1553	4000
Total	1044	1477	2479	5000

Table 1: Frequency Table of Spam E-Mails.

2.1 Odds Ratio (5 Pts.)

Find the odds ratio for an E-Mail being Spam for `Neither` vs. `Contacted` (ignoring `Colleague`). Interpret this number.

2.2 Model Formulation (5 Pts.)

Write down the logistic regression model formulation in detail for predicting `Spam` from the variables described in Table 1. Specifically make sure you have defined and described the coefficients and variables in the model. Use `Neither` as the base level.

Hint: Look at page 16 on the practise session slides *ML Basics* for how the formula for a logistic regression is defined.

2.3 Maximum likelihood (10 Pts.)

Derive the maximum likelihood estimates for the parameters of the model, using `Neither` as the base level.

2.4 Implementation (10 Pts. + 5 bonus pts.)

Implement this logistic regression with the data provided in Table 1 using Python 3.9 and `sklearn`. What is the difference between the estimated coefficients and the coefficients calculated above? Why does this difference occur?

Bonus for +5 pts.: Implement the gradient descent-based method introduced in the lecture. What values of the learning rate seem to work the best? How do the results compare to the ones obtained from `sklearn`.
Hint: A good starting point is probably the documentation of `sklearn`:
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

2.5 Trustworthiness (10 Pts.)

Given the model and the data as described in Table 1, would you trust such a model to classify E-Mails correctly into Spam and Not-Spam? Justify your answer and point out strengths and weaknesses as well as possible remedies of this approach.

2.6 Tf-idf (10 Pts.)

Research what the term tf-idf stands for and describe how it is used to vectorize text (cite your sources!). Do you think an approach using a tf-idf encoder is superior to the one described above?

3. SMS Spam Detection using textual features (30 Pts.)

Given the SMS spam dataset[1] introduced by Almeida et al. 2011, build a full spam prediction pipeline with the following parts:

1. Reading in the raw data.
2. Randomly splitting the dataset into training (80% of the data) and test (20% of the data).

[1] `https://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection`

3. Applying preprocessing. You are encouraged to use the function you wrote in 1.2, but can also use functions from any Python package you like.
4. Encoding your textual data using a tf-idf encoder.
5. Training a logistic regression model on predicting whether the input is `spam` or `ham` using only your training set.
6. Predicting on your hold-out test set and reporting your achieved accuracy.
7. Compare your results to a k-nearest neighbour search with at least $k = 3$ neighbours.

A note on the performance of your model: You will **not** be graded on the accuracy of your model, only on the soundness of your approach.

References

Almeida, Tiago A, José María G Hidalgo, and Akebo Yamakami (2011). "Contributions to the study of SMS spam filtering: new collection and results". In: *Proc. DocEng*, pp. 259–262.