

# TDT4171 Artificial Intelligence Methods

## Exercise 5

March 30, 2020

- **Delivery deadline: April 15, 2020 by 23:59 sharp. The deadline is hard.** No late delivery will be graded! Deadline extensions will only be considered for extraordinary situations such as family or health-related circumstances. However, these circumstances **must be documented**, e.g., with a doctor's note ("legeattest"). Having a lot of work in other classes is not a legitimate excuse for late delivery. Also, happy events such as birthdays, weddings, and vacations are not valid excuses as they are typically known well in advance, so you will need to plan your work (and delivery) schedule around them.
- Read the information provided on the assignment page on Blackboard **carefully**.
- Required reading for this assignment: Chapter 18.7, Paper on Deep Learning.
- Deliver your solution on *Blackboard*. Please upload your report as a **PDF file**. For the programming part, deliver the source code alongside the pdf file. Please **do not** put anything into an archive. Just upload the code alongside the pdf file. In other words, upload, **PDF file + source code**.
- Students can NOT work in groups. Each student can only submit solution individually.
- This homework counts for 4% of the final grade.
- Copying("koking") from other students is not accepted, and if detected, will lead to immediate failure of the course. The consequence will apply to both the source and the one cribbing.
- For help and questions related to the assignment, **ask the student assistants during the assignment hours**. The assignment hours can be found on the assignment page on Blackboard. If the student assistants are not able to help you or you have other types of inquiries, an email can be sent to Yan(yanzhe.bekkemoen@ntnu.no).

## Introduction

In this exercise you will try out several learning algorithms on a dataset containing text-based reviews of restaurants. The original dataset is available from <https://yelp.com/dataset/download>, but we will work on preprocessed versions available on Blackboard. The goal of the assignment is to be able to learn the meaning of each review: Is it positive or negative? Several learning algorithms will be employed, but you will not need to do the detailed implementations yourself. Instead we will use two Python packages: `sklearn` and `tensorflow.keras`.

## Implementation Part 1 – Using sklearn

In this part we will use scikit learn (install on your computer using `pip install -U scikit-learn`). scikit learn is a collection of machine-learning algorithms, and it also supplies a number of helpful preprocessing techniques. The documentation is available from <https://scikit-learn.org/stable/documentation.html>.

### About the data

Use the data in the file `sklearn-data.pickle` in this part. It is a python pickle file containing a single python dictionary. To load the data, use the command `data = pickle.load(open("sklearn-data.pickle", "rb"))`.

This gives you a dictionary with 4 keys:

- **x\_train**: A list of strings. Each element in the string is one review. You can take a look to get a feel for what the reviews are like. This is the data we will use for training the classifiers.
- **y\_train**: A list of integers. This is the target of the learning, with 1 representing a positive review and 0 a negative one. The ordering of elements fit that of **x\_train**.
- **x\_test**: A list of strings, where again each element in the string represents a review. This is the **test-set**, and should not be used during training.
- **y\_test**: A list of integers. These are the class-labels for **x\_test** and are used at test-time.

The machine learning algorithms we will use in this part are “feature-based”. This means that they expect an input where each observation (review) is represented by a fixed length vector, and where the interpretation of a given feature (e.g., the first one), is the same for all observations. The way we will ensure this is to recode the input. The features are to be defined as presence

or absence of specific words, so if the first feature represents the word “like”, both reviews “I like pizza” and “I don’t like pizza” would have one in the first position, while the review “I love pizza” would have a zero. Note also that the ordering of words is lost in this representation: We only know if a word is *present* in the review, we do not know *where* it was positioned.

The re-coding can be done fairly easily using a dictionary, but in this assignment you may rather use `HashingVectorizer`, a utility preprocessing function that can be imported from `sklearn.feature_extraction.text`. Check the documentation to use the optimal settings; of highest importance are `stop_words`, `binary`, and `n_features`. The call to `HashingVectorizer` creates an object that you should use to transform both `x_train` and `x_test`.

## Learning classifiers

Scikit-learn has a quit simple interface to the long list of supported classifiers. First instantiate the classifier object with hyper-parameters, e.g. `classifier=BernoulliNB(<parameters>)`.

Now, `classifier.fit(X=trainng.observations, y=related.classes)` can be used to train the classifier.

Finally, `classifier.predict(test_set)` can be used to make predictions for a test-set. To measure the quality of the predictions we compare it to the correct classes using `sklearn.metrics.accuracy_score`.

Implement this pipeline for both the Naive Bayes classifier and the decision tree classifier. If you do it correctly, you should expect around 70% – 75% accuracy.

## Implementation Part 2 – Using keras

Tensorflow (documentation at <https://tensorflow.org>) is a library for deep learning. While flexible and powerful, it can be seen as having a steep learning curve. We therefore use Keras (<https://keras.io>) as a simplifying abstraction layer in this exercise. Keras is part of Tensorflow (install on your computer using `pip install tensorflow`), and can be accessed using `tensorflow.keras`. We will also use `keras.preprocessing`, which should be installed as one of Tensorflow’s requirements (if not, `pip install` it).

### About the data

Use the data in the file `keras-data.pickle` in this part. As in Part I the data is stored in a pickle-file containing a dictionary. The dictionary has the same keys `x_train`, `y_train`, `x_test`, and `y_test` as used in Part 1.

Additionally, `vocab_size` holds information about the number of different words in the vocabulary and `max_length` gives the longest review in the dataset (in terms of the number of words used).

The data in this part are encoded as lists of word-id's, so the first element starts with `[117, 143, 307, 468, 38, 117, 411, ...]`. In this part we will build a *recurrent neural network* (more precisely an LSTM), and in Keras this is accomplished by first making each sequence (that is, each document) have the same length. Use `sequence.pad_sequences` from `keras.preprocessing` to accomplish this. Both the training-data and the test-data must be padded to the same length; play around with the `maxlen` parameter to trade off speed and accuracy.

## Learning the classifier

To build the recurrent neural network we will use Keras' `Sequential` API. After defining a model by `model = tf.keras.Sequential()`, we are able to build up the model by simply adding layers one by one. Your model needs to start with an `Embedding`, that embeds the word-vectors into a high-dimensional space (`model.add(tf.keras.layers.Embedding(<parameters>))` will do that for you; take care when choosing the parameters). You also need an `LSTM` layer and a `Dense` layer at the end. Take care when you choose the parameters also for these layers.

When the model is done, you can use `model.fit()` to train the model. The dataset is quite large, so if you run this on your own computer you may not want to do more than three epochs (number of epochs can be chosen by setting the `epochs: int` argument in the call to `model.fit`).

Finally, we evaluate the model with `model.evaluate(x_test, y_test)`. If you do this correctly and choose reasonable parameters you should expect at least 90% accuracy from the LSTM.

## Report

Supply a report together with your code where you briefly ...

- Thoroughly document the parameters you have used when defining all three models, and their respective obtained accuracies.
- What is the reason for the large improvement in accuracy from the Naive Bayes/Decision Tree models to the LSTM? Give the most important reasons.
- Make sure the code is appropriately commented and runnable before delivery.