Final Assignment
Python for Text Analysis

2017-2018

1 Introduction

This final assignment is the last part of the Python for Text Analysis course. Now that you have learned the basics of the Python programming language, it's time to put your skills into practice and work on your own code project.

This is a group assignment in which you will work together with one other student. You can form your own team.

For this assignment, you will choose a classification task and a corresponding dataset (see below) for which you are asked to:

1. download/obtain the data;
2. split the dataset into train/test sets;
3. read and process the files in your dataset;
4. extract relevant statistics from those files;
5. store the computed statistics in a useful format (e.g. CSV/TSV);
6. present the statistics to the user by means of visualization;
7. use the computed statistics as features for the classification task;
8. save the predictions of your model on the test data in a separate file;
9. BONUS: evaluate your system's accuracy on the test set.

Each of these steps are explained in more detail below.

2 Classification Tasks and Datasets

You can choose from the following tasks with corresponding datasets:

Task	Dataset	Type of data	Language(s)
Authorship Attribution	Reuter_50_50	newswire	English
	The Enron Email Dataset	corporate e-mails	English
	Project Gutenberg	books	English
	DARIAH Dataset	plays/tragedies	French
Author Profiling	Twitter User Gender Classification	tweets: male/female/brand	English
	PAN 2017 Author Profiling	tweets: gender & language variety	English, Spanish, Portugese, Arabic
	PAN 2016 Author Profiling	tweets: gender & age	English, Spanish, Dutch
	TwiSty Corpus	tweets: gender and personality (MBTI)	Spanish, Portuguese, French, Dutch, Italian, German
	CLiPS Stylometry Investigation (CSI) Corpus	student texts: gender, age, sexual orientation, region of origin, personality (MBTI)	Dutch
Genre Classification	SignalMedia	blogs vs. news	mainly English
Source Classification	SignalMedia	blogs & news from different sources	mainly English
Binary Sentiment Classification	Large Movie Review Dataset	IMDB movie reviews	English

If you have an alternative idea for a task or dataset, you can discuss it with us. We will decide whether it's suitable for the final assignment.

3 The Assignment: Step-by-step

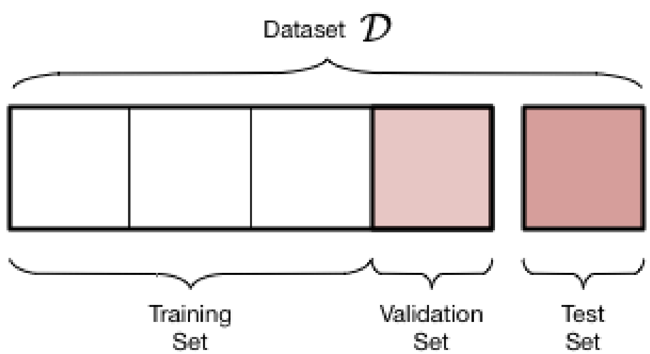
In the following, we explain step-by-step what you need to do to finish the final assignment. However, keep in mind that this assignment isn't something you should do in a linear fashion. It is a good practice to first go through all of the steps and have a minimal version of the assignment. For example, you could first extract one feature in step 4, and use only this feature in steps 5-8. Once you have this up and running, you can go back to step 4 and extract another feature. This way you will get a feeling for how much work the assignment will be, and what format the features should have for you to easily store and visualize your results.

Step 1: Obtain/download the data

You can use the links above to obtain the data. Some of them require you to fill in a request form (and optionally a license agreement). Others require a password (PAN17) or provide additional download software that is necessary to be able to use the data (PAN16). Discuss your choice with us and we will help you understanding the procedure.

Step 2: Split your data into train/test sets

In this assignment, we are working towards creating a model for an automatic classification task. It is common practice to split your data into subsets: training data and testing data. Sometimes there's a a third subset: the validation (or development) set, but we will not take that into account for this assignment. By splitting your data, you can fit your model on the train set in order to make predictions on the test set.



For some of the datasets that we'll use in this final assignment, the data is already split into train/test sets. If that's not the case for your data, your first task is to **split the data using a 80/20 ratio**, which is usually a fair split or at least a good starting point.

How to split your data, depends on how it is structured. Sometimes you can do it manually (if the data is structured in separate files), but you likely need to write some Python code to do it for you. Make sure that the classes are at least somewhat distributed over both your train and test sets, i.e. don't assign one class to the train set and the other to the test set. After all, you cannot expect a model to recognize a class in the test set if it has never seen it before.

Step 3: Read and process the files

Now that you have split your data into a train and a test set, you should read the texts in your data. This may mean simply opening your files, or you should extract the relevant part from the CSV or XML structure. Once you have the text available, you should do **at least** the following Natural Language Processing tasks:

1. Sentence splitting
2. Tokenization
3. POS-tagging
4. Lemmatization

You will need these in order to compute the statistics from the texts (see next step). We suggest you use either NLTK or spaCy.

In addition, depending on the features you want to extract, you may need to do one or more of the following:

5. Named Entity Recognition (e.g. with spaCy)
6. Sentiment Analysis (e.g. with NLTK)

If you are comfortable with using other tools or modules (e.g. Stanford CoreNLP), feel free to use them. Be careful you don't spend too much time on this though!

Step 4: Extract statistics (features) from files

The next step is to write functions to extract some relevant statistics or *features*. By features, we mean properties or characteristics of a text that are useful inputs for a machine to make predictions about that text. For example, stylometric features like punctuation are known to be effective for predicting the author or their gender/age.

Here are some suggestions for features that you can extract for each text (they should be numerical, i.e. counting something):

- Character-based features:
  - number of characters
  - number of letters
  - number of uppercase letters
  - number of lowercase letters
  - number of numeric characters
  - number of whitespace (tab/space/newline) characters
  - number of special characters
- Punctuation-based features:
  - number of commas
  - number of dots
  - number of exclamation marks
  - number of question marks
  - number of colons

- number of semicolons
- number of hyphens

- Word-based features:
  - number of words
  - average word length (characters)
  - number of long/short words
  - number of stopwords
  - number of emoticons
  - number of spelling errors
  - vocabulary richness: type-token ratio (TTR)
  - vocabulary richness: hapax legomena/token ratio (HTR)
  - frequency of specific words (i.e. most-frequent words in corpus)
- Sentence-based features:
  - number of sentences
  - average sentence length (characters, words, clauses)
  - standard deviation of sentence length
- Paragraph-based features:
  - number of paragraphs
  - average paragraph length (characters, words, clauses, sentences)
- Syntactic features:
  - number of part-of-speech tags
  - number of function words
  - number of content words
  - average length of noun/verb phrases
- Semantic features:
  - overall sentiment score
  - number of positive/negative words
  - number of named entities
- ...your own suggestion(s)

You don't have to extract all of these features; pick those that you think are most relevant for your task/data. Try to think of the requirements each of them has: what do you need to do in order to compute these features? However, you shouldn't write a separate function for each feature! Instead, group features that are naturally related to each other so that you minimize repeating yourself.

Some of these features sound very basic, but don't let their basic nature fool you! This piece on punctuation in novels shows you how informative punctuation can be. And here is an excellent quote by Gary Provost (from 100 Ways To Improve Your Writing) that shows the power of sentence length to change the character of a text:

VARY SENTENCE LENGTH

> This sentence has five words. Here are five more words. Five-word sentences are fine. But several together become monotonous. Listen to what is happening. The writing is getting boring. The sound of it drones. It's like a stuck record. The ear demands some variety.
>
> Now listen. I vary the sentence length, and I create music. Music. The writing sings. It has a pleasant rhythm, a lilt, a harmony. I use short sentences. And I use sentences of medium length. And sometimes when I am certain the reader is rested, I will engage him with a sentence of considerable length, a sentence that burns with energy and builds with all the impetus of a crescendo, the roll of the drums, the crash of the cymbals--sounds that say listen to this, it is important.
>
> So write with a combination of short, medium, and long sentences. Create a sound that pleases the reader's ear. Don't just write words. Write music.

We recommend that you read A survey of modern authorship attribution methods by Efstathios Stamatatos. This paper provides a nice overview of features that are commonly used in authorship attribution and can be applied to other tasks as well.

Step 5: Store the computed statistics in a useful format (e.g. CSV or TSV)

After computing the statistics, you can store them on two levels:

1. **Statistics for single training example**: for each example case, store your computed features and the gold class (e.g. author for author identification)
2. **Aggregate per prediction class, e.g. per author**: each row represents a class, while the columns represent aggregated statistics/features per class

For example, let's say we are dealing with the author identification task and we want to distinguish between the authors Jane Austen, Charlotte Bronte, and George Eliot. Let's consider that we have 2 books from each author that we compute the statistics on, and that we compute two statistics: average number of tokens per paragraph and average sentence length.

1. For a **single training example**, we would store 6 rows, each representing a book. For each row, we would have two numbers, one for each of the statistics, and the gold class (which of the three authors this book belongs to).
2. For the **aggregated statistics**, we would store 3 rows, each representing an author. For each row, we would store two numbers, one for each of the statistics.

For predicting classes (step 7), you will use the aggregated statistics. For visualization, however, you can also use the individual statistics.

Step 6: Present the statistics to the user by means of visualization

For this purpose, create at least three visualizations of the statistics you inferred on the training set. A simple example of a visualization would be a graph that shows the number of tokens per chapter for each of the books. Try to use the visualizations to help yourself and the reader of your project understand the data and/or your approach.

Step 7: Use the computed statistics as features for an automatic classification task

By computing a set of statistics, you are already halfway towards building a classifier. Let's look at the other half. It is common in NLP to combine computed statistics in a machine learning decision making system. Since we are not teaching machine learning in this course, we suggest a different method for automatic classification.

Namely, given the values for N features, we want to compute the best match. This is achieved in two steps:

1. Create a model of the classes you have encountered in the training set (you created this already in the steps 4 and 5 of this assignment). This model tells us what is the representative number of tokens, paragraph length, type-token ratio, etc. for each class.

2. For each test case, you compute the values for the same N features, and you compare their value to each of the known classes to determine the best match. So, in our example, a test case would have the features $f_{t,1}$ and $f_{t,2}$. We can check how different is this test case from each known author, by computing its feature distance from the features of each of the three authors. The distance to, e.g. JaneAusten's features $f_{j,1}$, and $f_{j,2}$ is computed as:

$$D(t,j) = ||f_{j,1} - f_{t,1}|| * w_1 + ||f_{j,2} - f_{t,2}|| * w_2$$

   w_1 and w_2 are weights that you can use to favor certain features over others. For example, if w_1=0.8 and w_2=0.2, then we give 4 times more importance to w_1 than w_2. You can decide on your own weights; feel free to leave them out (i.e. have equal weights for all features).

   For a general case of N features (instead of 2), the formula can be written like this:

$$D(t,j) = \sum_{i=1}^{N} ||f_{j,i} - f_{t,i}|| * w_i$$

   In the same way we can also compute the distance to the typical model for the other two authors ($D(t,b)$ for Bronte and $D(t,e)$ for Eliot). Finally, we can find the best match by finding the minimum of all three distances for this test case:

$$D_{BEST} = min(D(t,j), D(t,b), D(t,e))$$

Step 8: Save the predictions of your model on the test data in a separate file

For each of the test cases, store two values: your system prediction and the gold class. Optionally, you can also store more columns, such as the features you computed for that test case. For example, let's say we want to use our author identification model to classify 5 books (`b1.txt` to `b5.txt`). Then this is an example output file:

file_id	gold	prediction
b1.txt	JaneAusten	JaneAusten
b2.txt	JaneAusten	GeorgeEliot
b3.txt	GeorgeEliot	GeorgeEliot
b4.txt	JaneAusten	CharlotteBronte
b5.txt	CharlotteBronte	CharlotteBronte

BONUS: Evaluate your system's accuracy on the test set

You can compute this by comparing your predictions to the gold class and counting the percentage of correct classifications. For the example case, this is 3/5=60% accuracy.

4 General Information

Important dates

When?	What?
Friday 22 December 2017 (13:00)	Decision about the task & dataset (please inform us by e-mail)
Monday 29 January 2018 (15:30-17:15)	5-minute presentation
Sunday 4 February 2018 (23:59)	Deadline submission final assignment

5-minute presentation

You'll also have to present your work in the last Monday lecture of the course. This is useful for several reasons.

- First, we want you to reflect on the way you've handled your project.
- Second, it's useful to see how other people tackle similar problems.
- Third, your fellow students' work may be useful to you in the future as well. We'd like to encourage you to check out what your classmates did.

Also note that the presentation is six days before the final project is due. This means that there is still time to incorporate the feedback from your peers into the project.

Preparation for the presentation Here are some questions to help you reflect on your project. You don't need to address all of these points in your presentation (5 minutes is really short!). Just highlight the points that are most important for you.

- What did you do?
- Why is it useful?
- How did you do it?
- What modules did you use? What are they useful for?
- What kind of data did you use? How did you get it?
- How did you manage your project? What did your workflow look like?
- How can others use or build on your project?
- What was the greatest challenge for you?
- What took the largest amount of time? (try to keep track of this)
- What would you do differently next time?

Note that these questions also serve to guide you through your project. Keep them in mind, and take notes as you progress. We also encourage you to discuss these points with your fellow students, even before the final presentation. In terms of format, you can choose any kind of presentation that you like (powerpoint, notebook, web demo, ...).

Grading

To score the final assignment, we will weigh our judgment criteria as follows:

	Weight
Code Accuracy	20
Code Structure	20
Content & Features	35
Visualizations	10
Documentation	10
Presentation	5
BONUS	5