Stylometry intro

Mateusz Żółtak

1 Introduction

When we are using statistical methods we should adhere to statistical inference rules.

Summarizing them in few points:

- Exploratory methods (most of the ones we will be using today) give us only some hints.
  - This hints come from correlations and do not proof cause-effect relations
    - See e.g. funny examples (http://www.dailymail.co.uk/sciencetech/article-2640550/Does-sour-cream-cause-bike-accidents-No-looks-like-does-Graphs-reveal-statistics-produce-false-connections.html) and serious wikipedia article (https://en.wikipedia.org/wiki/Correlation_does_not_imply_causation)
  - Thus we should be very careful when we are drawing conclusions
- Although it is technically possible, it is a very bad practice to use statistical methods as a “black box”
  - You always make assumptions regarding what given method does.
  - But you are typically wrong
  - It is highly possible that this will affect your conclusions
    - (surprisingly it is always possible that it won’t - funny feature of the implication is that false premises can also lead to true conclusions)
- Without a convincing story explaining how numbers we put into and obtain from statistical analysis refer to the real world statistics is only playing with numbers.

Because of that we will now spend some time on talking about statistical methods used in stylometry (to be precise in the stylo package).

2 Stylometry analysis

In stylometry we are investigating (dis)similarities between texts using (mainly) frequency analysis. This process can be divided into steps:

1. Data gathering
2. Choosing and extracting features
3. Performing statistical analysis
4. Investigating results

2.1 Data gathering

The stylo package we will be using assumes that all texts are grouped by primary and secondary attributes, e.g. author and text title or magazine title and year, etc. It also assumes that texts are simple text files and these attributes can be obtained by splitting file name on the first occurrence of an underscore.

Because on one hand we want to be able to analyze our texts in a flexible and on the other we would prefer to avoid renaming files by hand a special database and small R package were provided.

2.1.1 Available texts

- 5 Abacus texts with information on author, title and date (~1 MB);
- 10 “death fliers” with information on COMPLETE (~50 kB);
- British fiction - corpora of 27 British novels by 11 authors with information on author, title and date (~35 MB);
- Harper Lee - corpora of 29 novels by 9 authors with information on author, title and date (~17 MB);
- samples from the AMC with information on the station/magazine/newspaper name, date and author/speaker (if provided in the AMC):
    - tv: MWVOLL from years 2003, 2007 and 2013
    - magazines:
        - FALTER from years 2001 and 2013 (~27 MB)
        - PROFIL from years 1994, 2001 and 2013 (~50 MB)
        - SPORTZTG from years 1996, 2001 and 2013 (~12 MB)
        - WOMAN from years 2003, 2007 and 2013 (~17 MB)
    - newspapers:
        - HEUTE from years 2007 and 2013 (~25 MB)
        - KRONE from years 1994, 2001 and 2013 (~566 MB)
        - PRESSE from years 1994, 2001 and 2013 (~300 MB)
        - STANDARD from years 1994, 2001 and 2013 (~270 MB)

2.1.2 Gathering texts

See the “Important commands” document.

2.2 Features

Choosing and extracting features is quite a wide topic. It can be divided into:

1. Choosing features
2. Extracting features from texts (we will skip it as it is purely technical)
3. Computing features frequencies
4. Tuning up features frequency table

2.2.1 Choosing features

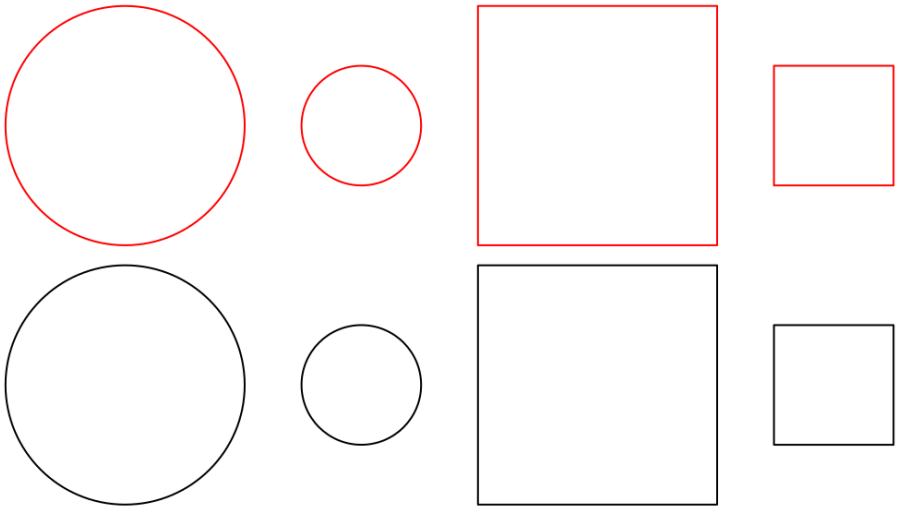
Feature is a primary unit of our analysis. Typically features are:

- words
- or characters
- or word/character n-grams
    - word/character n-gram means n following words/characters
    - “The quick brown fox” has 3 word 2-grams: “The quick”, “quick brown” and “brown fox”

but to be honest you can choose any feature you want (and able to extract from text), e.g. “noun - adjective pairs” (if you have POS tagged corpus) or “sentence length”.

When you are choosing a feature, you decide what defines “the text style” for you.

Lets think about it for a while - please group figures below by a common style (there are no bad answers!):



- Depending on the feature we choose (shape, size or color) we will achieve different grouping.
- It is also perfectly possible that **depending on the feature you choose** (and a way you will analyze them) **you will get different results** of your stylometry analysis.
    - It won't mean that "something gone wrong" but that **"the text style can be defined in many ways"**.
    - It is one of the most difficult part of statistical analysis (not only stylometry ones) to give a **meaningful descriptions** to the results.
        - E.g. to give a meaningful description of *"relative frequencies of word 2-grams limited to 100 most frequent ones, with prepositions omitted and a Classic Delta distance metric used"*.

2.2.1.1 Technical hints

Taking apart philosophical deliberations according to the stylo package how-to:

- word 2-grams typically perform good;
- if you have many typos in your text (e.g. you have OCR results without manual review), character n-grams perform much better then words.

2.2.2 Computing features frequencies

When we extracted features lists from our texts, we can compute their frequencies. We can do it:

- In absolute terms (e.g. "word A appeared 15 times in text X")
    - What is really our feature if we apply absolute frequencies to texts which differ a lot in length?
        - We can deal with this problem by drawing equal size samples of features from our texts before computing frequencies.
- Relatively to text length (e.g. "word A is 0.45% of all words in text X")
    - Relative frequencies are resistant to differences in texts length.

As a result we end up with a table of frequencies of each feature in each text.

It will look like (for sample 7 words and 3 texts):

```
##        the   to  and    a   of   he was
## text1 4.35 2.22 1.24 2.87 1.27 1.88 1.7
## text2 4.89 1.86 1.50 2.92 1.60 2.00 1.3
## text3 3.81 2.32 1.29 2.80 1.25 2.03 1.4
```

2.2.3 Tuning up features frequency table

We might have many concerns regarding the frequency table of all features:

- this tabe is typically very large (due to high number of different features found in texts) which makes it impractical to analyse;
- many features have extremely low frequencies;
    - if it is a problem for us or not depends on our definition of “the text style” but typically in stylometry people focus on analyzing most frequent features;
    - special attention should be paid to very rare features present in single texts only - their impact on analysis may be much stronger then we want;
- we can think some features are irrelevant to “a text style”, e.g. pronouns, articles, etc.

According to all these concerns we can tune up the features frequency table, by e.g.:

- remove irrelevant features (e.g. pronouns);
- remove features which appear only in some texts
    - this is called “culling”
    - we can decide on the proportion of texts in which a feature has to appear
- leave only a part of the table
    - typically N most frequent words
    - but we can also choose N less frequent or take N most frequent and then skip M most frequent of them.

Resulting table will be a basis for statistical analysis.

2.3 Statistical analysis

Most stylometry analysis base on:

- computing distances between texts (on the basis of the features frequency table)
- grouping/comparing resulting distances

so at first we should understand what “a distance between texts” means.

2.3.1 Computing distances

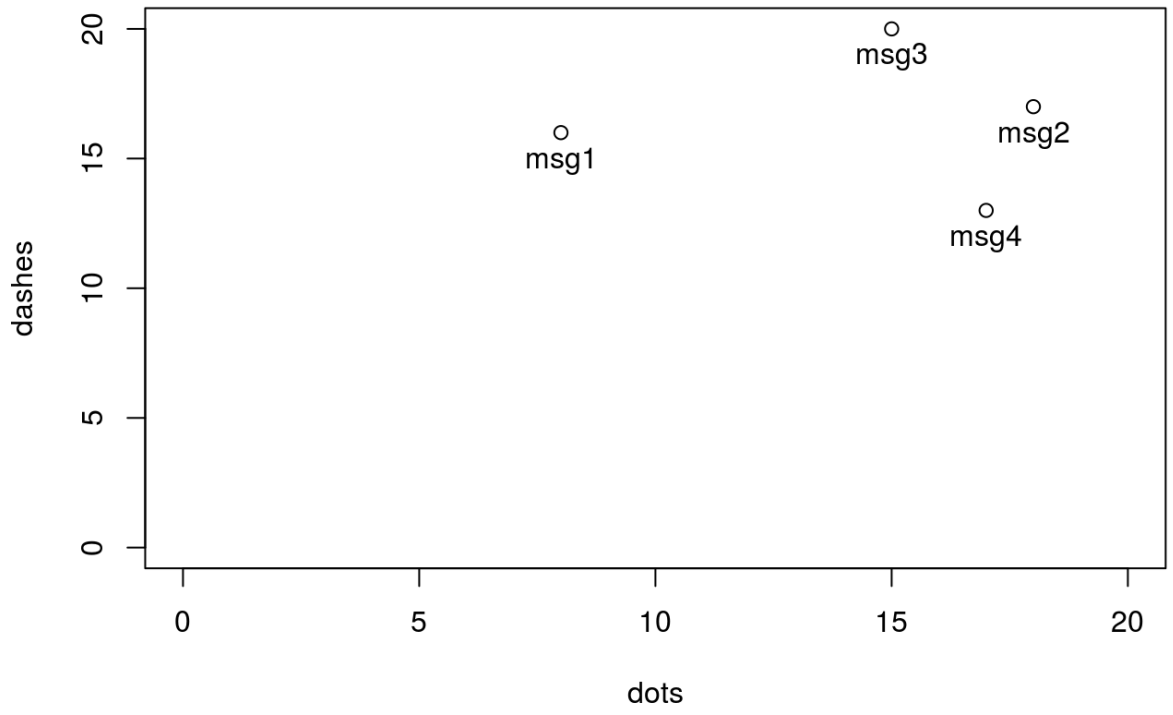
The most important think to understand is that *every feature will create a separate dimension* in “the text style space”.

To make it easier to understand lets consider a 2-dimensional case:

- Lets assume we have corpora of 4 Morse code encoded messages, we choose a single Morse code character as our feature and we are using absolute frequencies
- This means we have only 2 features (“dot” and “dash”)
- Every text will be characterized by “dots” and “dashes” counts which we can easily put on a graph

A sample data:

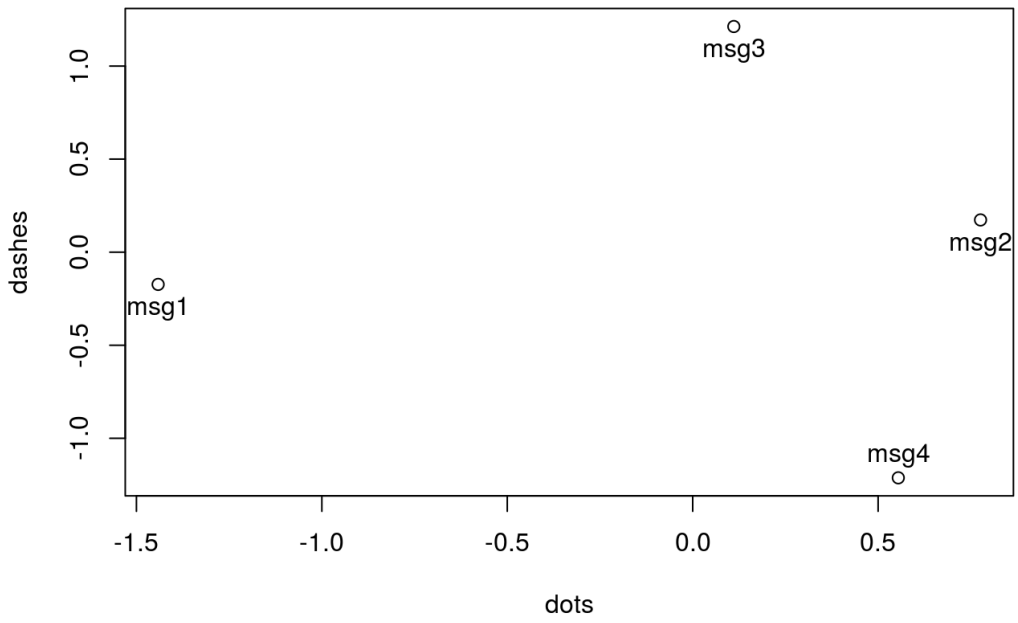
```
##      dots dashes
## msg1    8     16
## msg2   18     17
## msg3   15     20
## msg4   17     13
```



As we can see **we didn't put all frequencies on one dimension but we have separate dimension for each feature.**

Now the question is how to compute distances between texts?

- the simplest method is "an Euclidean distance"
  - we will simply draw a straight line between each pair of points and take its length
  - it can be computed using the Pythagorean theorem:
    $distance = sqrt((dots_a - dots_b)^2 + (dashes_a - dashes_b)^2)$
- another method can be distance in so called Manhattan (or taxi or city) metric
  - Manhattan distance is defined as a sum of distances on each dimension - in our case it will be simply: $distance = |dots_a - dots_b| + |dashes_a - dashes_b|$
- in stylometry Classic Delta metric is a very popular distance measure
  - it is defined as (in our two-dimensional case):
    $distance = 0.5(|\frac{dots_a - dots_b}{dots_{sd}}| + |\frac{dashes_a - dashes_b}{dashes_{sd}}|)$
  - as we can see, it standardizes values on each dimension, so the mean is equal to 0 and standard deviation is equal to 1



- there are many others (and you can invent your own too)

Lets compare differences:

```
##           msg2-msg1 msg3-msg1 msg4-msg1 msg3-msg2 msg4-msg2 msg4-msg3
## Euclidian    10.050     8.062     9.487     4.243     4.123     7.280
## Manhattan    11.000    11.000    12.000     6.000     5.000     9.000
## Cl. Delta     1.282     1.469     1.518     0.852     0.804     1.434
```

The same values as a relation to the shortest distance (separately for each method):

```
##           msg2-msg1 msg3-msg1 msg4-msg1 msg3-msg2 msg4-msg2 msg4-msg3
## Euclidian     2.437     1.955     2.301     1.029         1     1.766
## Manhattan     2.200     2.200     2.400     1.200         1     1.800
## Cl. Delta     1.595     1.828     1.888     1.060         1     1.784
```

What we can see:

- distance values vary a lot among methods
- message 2 is always closest to message 4
- distance between message 1 and messages 2 and 3 varies a lot depending on the metric:
  - in the Manhattan metric distances between message 1 and messages 2 and 3 are equal
  - in the Euclidian metric message 1 is closer to message 3 then to message 2
  - in the Classic Delta metric message 1 is closer to message 2 then to message 3

What is the right way to measure distance then?

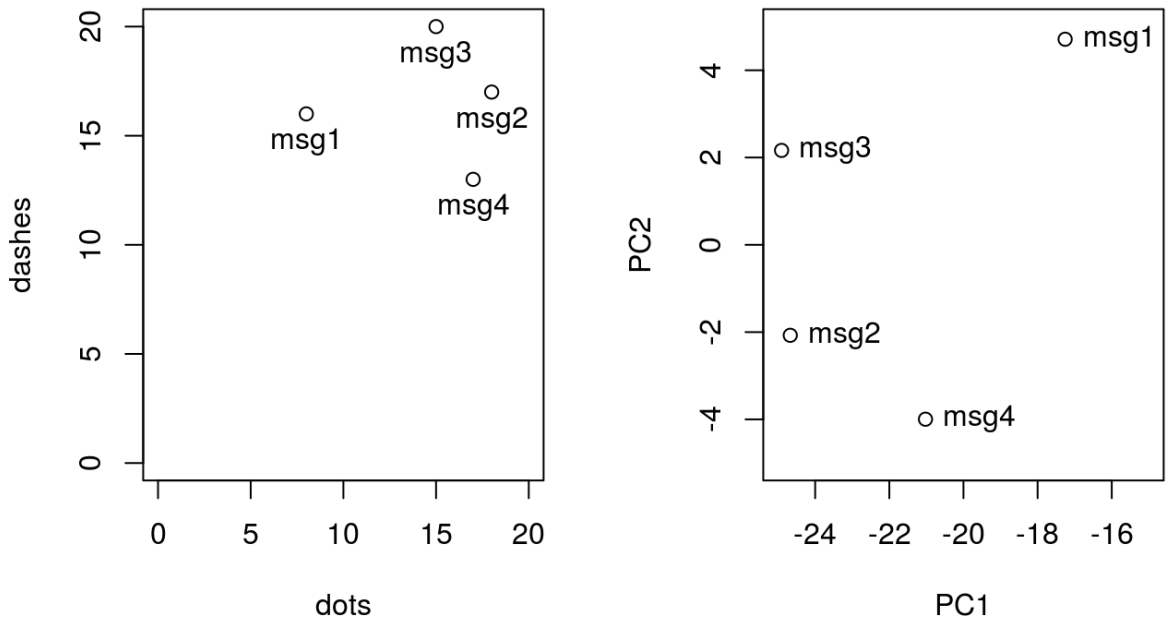
- **There is no *only one correct way* for measuring distances.**
- **By choosing distance metric you decide what does "a difference in text style" mean for you.**
- To make a correct choice you should be aware about each metrics characteristic, e.g.:
  - Euclidian and (even more) Manhattan are sensitive to differences in a scale used on each dimension while Classic Delta is not.
  - Manhattan and Classic Delta are insensitive to correlation between dimensions while Euclidean is not (correlation between dimensions will lower distance in Euclidian metric).
  - Classic Delta performs standardization - resulting value scale differs from dimensions scale.
- We should be also aware of some general mathematical characteristics of *metric spaces*:
  - You shouldn't be attached to exact values. Any *linear transformation* (multiplication by a constant and/or adding a constant) of all values which give equivalent results.
- Again, the most difficult part is to find a meaningful "real world" descriptions to these mathematical characteristics.
  Some (but stil quite low-level) examples can be:
  - Euclidian and Manhattan metrics are strongly influenced by dimensions with "a long scales" (scales with very high maximal observed values). If our feature were words this would mean they are strongly influenced by the most frequent words (and even more if we use features counts). Classic Delta metric is not because it standardizes values on each dimension (transforms them to more or less common scale) before computing distance.
  - In Classic Delta and Manhattan metrics we skip information on co-occurrences of features (e.g. in the German language the fact that the word "der" will correlate with masculine nouns) while in Euclidean metric such correlation will shorten distance a little.

2.4 Analysis

2.4.1 Principal Component Analysis

- In PCA we are trying to find “a common dimension” for our space of features. We are calling it Principal Component.
  - We can try to interpret it as a “most important factor of a text style according to a chosen feature”.
  - It is very useful if we are able to give it a descriptive interpretation (although it is typically very hard).
    - e.g. in educational assessments we would interpret it as a “overall intelligence”
- Than we can try to find the second “common dimension” which will be perpendicular (in algebraic sense) to previous ones. We will call it Secondary Component.
  - We can repeat this procedure to obtain higher order components.
  - Computing higher order components can be used to asses how many “meaningful dimensions”" are in our features frequency table, e.g.:
    - If some features are highly correlated, PCA will “merge them” in one dimension.
    - Each component describes some amount of variance in our feature frequency table. In most cases this amount will decrease very fast with higher component order, so we can easily divide components into important ones (describing a lot of variance) and irrelevant.
- All remarks regarding *metric spaces* apply here, especially resulting components can be *linearly transformed* in regard to our original scale.
- In contrary to other methods in PCA we do not choose the method of computing distances.

Employing PCA on our test data set will give us:



```
## Importance of components:
##                             PC1     PC2
## Standard deviation     25.6181 3.96381
## Proportion of Variance  0.9766 0.02338
## Cumulative Proportion   0.9766 1.00000
```

And now kingdom for a meaningful name for Principal and Secondary Component :-)

2.4.2 Cluster analysis

Having computed distances between each pair of documents we can try to build up a tree (dendrogram) in which similar texts will be close to each other.

Again there are many different methods which may result in different structure of the tree and there is no clear rule which one is the best.

Good summary is available here (http://arxiv.org/pdf/1105.0121.pdf)

2.4.3 Multidimensional scaling

Our features frequency table typically has much more dimensions that we can imagine (I am completely unable to imagine more then 4 dimensions and typically we have hundreds of them). MDS method tries to deal with that problem by:

- reducing dimensionality of the data (in the stylo package always to 2 dimensions, in general to any number)
- while trying not to change distances between points.
    - Typically it is impossible to keep all distances but the transformation is done in such a way, that a sum of differences between original and resulting distances is as small as possible.
    - The lower number of target dimensions, the higher differences.

Reducing dimensionality of our features frequency table to 2 makes it possible to draw a nice plot. Looking on the plot we can draw some conclusions about similarities of examined texts but we should remember this is extremely weak evidence.

2.5 Classification

In classification methods we try to guess some text trait (e.g. authorship) by comparing it to texts with known trait values.

We call the set of texts with a known trait value *a training set* and the set of unknown texts *a test test*.

- Delta
  - Match unknown text with the closest one according to chosen distance metric.
- Nearest Shrunken Centroid
  similar to Delta:
  - At first we group all texts from the training set according to trait value and compute avarege position on every dimension for all such groups; this average position is called *centroid* and might be seen as just another text in our features frequency table.
  - Then we compute Euclidean distance between unknown text and every centroid.
    - In general we could use any metric, but the stylo package does not support that
  - Finally we match unknown document with the closest centroid.
- k-Nearest Neighbors
  - Euclidean distance is computed between unknown text and every text from the training set;
    - In general we could use any metric, but the stylo package does not support that.
  - Then we take *k* (which is algorithm parameter) closest texts (neighbors) and count how many of them point to each trait.
  - We choose trait pointed by majority of chosen neighbors. In case of a tie we choose at random.
- Naive Bayes
  - we don't have enough time to talk about Bayesian methods
- SVM
  - To be honest - I don't know how it works