Text Classification

What does it do? Text classification algorithms assign *labels* to texts from a pre-cultivated list.

Typically you take a small labeled set of data, and extrapolate those labels to a larger set that you don't know the classes for.

What's so great? It can help you find a needle in a haystack, theoretically; classify texts as protestant sermons and find more protestant sermons out there.

Authorship attribution is one subset of this that has sometimes been a particularly active area of research. In this case, the labels are *authors*. You can find out whether an author wrote a text by seeing how confident the classifier is in attributing that author to the text.

What decisions do you have to make? What are the "features" the algorithm will use? Usually, it's words; but you can throw other elements into the mix. And often you have to reduce the number of words.

What are the classes you want to test? It's often easier if it's a simple yes/no decision.

What implementation should you use? There are two choices to make here: the algorithm, and the software.

In terms of algorithms: if you want great performance, talk to a computer scientist or statistician. Unlike many other things digital humanists do, this directly maps to their areas of research. If you can't find one, there are several sensible approaches to try out. "K-nearest-neighbor" approaches find the most similar documents. Logistic regression uses words to adjust probabilities of being in class or another.

In terms of software; you will have to use a computer language to do this well, because the first run won't work. Python or R might both work: for certain goals like authorship attribution, the `stylo` package in R already includes a number of these algorithms specifically for text. Mallet, widely used for topic modeling, also implements logistic regression.