ž— layout: default title: Initial Data Analysis parent: Lesson nav_order: 3 —

1. Initial Data Analysis

Initial data analysis (IDA) is performed early in the textual analysis workflow in order to better understand the reliability and completeness of the data. It may be done with a number of tools, such as text editors and visualization software like Voyant. In the lesson that follows, we will use Microsoft Word to perform initial data analysis given its available functionality and ubiquity; Apache OpenOffice is an open-source alternative if you do not have access to MS Word.

Initial data analysis gives us a sense of the errors that exist within the corpus, but also informs the design of your pre-processing workflow: in what order should the error correction steps take place? What potential new errors may be introduced by an error correction strategy that is too broad (e.g. replacing "m" with "in" because "in" is sometimes misinterpreted as "m")? What tasks can be streamlined? You may, for example, decide that you should group similar documents together to efficiently pre-process them.

For example, if you notice that errors in the document tend to be consistent and few in number (i.e. 95-99% accuracy) then using OpenRefine may be a feasible approach to automate some of the tasks while retaining greater control over the corrections that occur. A higher number of unpredictable errors may require using a machine learning system that you train yourself.

Initial Data Analysis (IDA) and Exploratory Data Analysis (EDA)

Initial data analysis informs the pre-processing stage while exploratory data analysis concerns testing hypotheses and identifying salient features of the data to communicate in the output stage.

The two are not neatly separated, however; you are likely to notice some features and characteristics of the data as you are performing your initial data analysis (and in other stages of the workflow). But generally, exploratory data analysis will largely occur towards the end of the workflow when we are performing natural language processing tasks such as named entity recognition and topic modelling.

The following steps use the spell-check function within Microsoft Word and a macro to perform inital data analysis. If you prefer, you can alternative use Python to create a list of misspelled words.

Performing Initial Data Analysis (IDA) with Microsoft Word

Jump to step >

1.1. Remove extraneous text

1.2. Run spellcheck in MS Word

1.3. Create a macro to export an OCR error list

1.4. Review the OCR error list to inform the design of your pre-processing stage

1.1. Remove extraneous text

Working with sample corpus A, Zwick.txt, open the .txt file in MS Word.

Before any other step, we may first want to delete the preamble at the beginning of the document as it is not relevant to our text analysis. There are also figures and appendices at the end of the document that can also be deleted; anything after section 10.0 ("References") or page 39 in the document is similarly "noisy."

Consider doing the same with any other dataset you plan to analyze: remove blocks of extraneous text in the IDA stage.

1.2. Run spellcheck in MS Word

We are using Word because it will highlight errors for us and we can export a list of misspelled words from it using a macro. The software, however, may not recognize errors automatically when working with a .txt file; if you are running MS Word on either a Windows or Mac operating system, refer to the Microsoft documentation ("Turn the automatic spelling and grammar checker on or off").

Once you have made spelling errors visible in your document, take a bit of time to review them - they are now easy to find! Although we are about to export a list of misspelled words, it is helpful to see the errors in context as it can at times be difficult to infer the correct spelling of the error without knowing how they are being used.

1.3. Create a macro to export an OCR error list

Although seeing the errors in context is helpful, it is of course also useful to isolate the errors. We can create a list of OCR errors using a macro in MS Word.

In MS Word, use the shortcut keys `Alt` + `F11` to open the Visual Basic Editor. Once the Visual Basic Editor is open, the shortcut key `F5` will bring up the "Run Macro" dialog box, which is where we will create the macro.

Name your macro "GetSpellingErrors" and select "Normal" from "Macros In:" which will make your macro available outside of your current document for future use. Create the macro.

Copy the macro code below - created by Allan Wyatt - and paste into the code area of macro, overwriting any existing text (by default, the editor will "Sub GetSpellingErrors()" and "End Sub" code).

NB: I don't know Allan Wyatt personally or anything about his political views but I appreciate him sharing the code below freely.

```
'Macro code by Allan Wyatt: https://word.tips.net/T001465_Pulling_Out_Spelling_Errors.html'
Sub GetSpellingErrors()
    Dim DocThis As Document
    Dim iErrorCnt As Integer
    Dim J As Integer

    Set DocThis = ActiveDocument
    Documents.Add

    iErrorCnt = DocThis.SpellingErrors.Count
    For J = 1 To iErrorCnt
        Selection.TypeText Text:=DocThis.SpellingErrors(J)
        Selection.TypeParagraph
    Next J
End Sub
```

Typing the "Run Macro" shortcut key (`F5`) again will run the Macro - you should now have a second MS Word document open that contains the errors words only.

If you are using OpenOffice, you should be able to similarly create a macro using the same code.

1.4. Review the OCR error list to inform the design of your pre-processing stage

Now that you have the errors separated out from the text, review your data closely. You can toggle back and forth between the error list and the full document, Zwick.txt, or the source document if the correct spelling of some misspelled words is difficult to guess. You may also wish to employ a text analysis tool like Voyant Tools.

You will notice that most of the OCR errors fall under one or a combination of the following: * misinterpreted characters * omitted characters * extra characters * split words (two words that are meant to be one, or where one of the characters has been misinterpreted as a space)

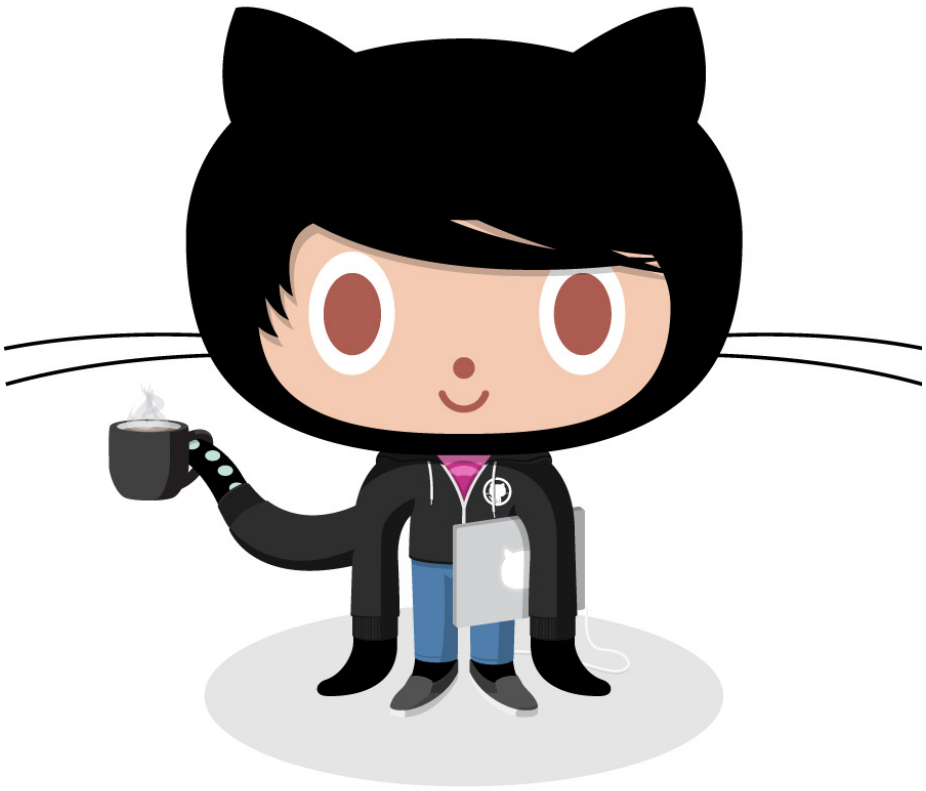
Error words are likely to duplicated - in fact, we are depending on the same words being misspelled in the same ways for the error correction techniques we will be using shortly. If you would like to streamline the list and remove multiples, you can bring the error list into OpenRefine and use *facets* to group repeated values together (see "Strategies for OCR Error Correction in OpenRefine").

In your initial data analysis, try to identify patterns within the errors: * is one character frequently the source of errors (such as “m” in the sample text)? * is one or a set of characters regularly substituted for another (“oim” for “oun” in the sample text)? * does an error often occur in relation to another character or characters (like “m” for “in” when following “f” or “tiy” for “tly” in the sample text)? * etc.

While it may be tempting to create a table containing the misspelled word and its corrected version for use with a find-and-replace technique, a manual approach will be unsustainable for all but the smallest of datasets. We will instead focus our efforts on misspelled groups of letters repeated across numerous words and not let perfect be the enemy of good.

One exception to the above is any words that are particularly significant to your analysis; you can safely ignore misspellings of “the,” “this,” and other commonly used words but - to take an example from our Zwick dataset - you would probably want to correct “Mimicipal.”

Document your observations as completely as possible to help to make your error correction tasks more efficient and less likely to introduce new errors.



stupac a	stupac b
———	———
redak 1	redak 2