Lab 4: Text Analysis

Week 7

Walid Magdy

March 2026

1 Introduction

In this lab, we will explore how to use automated methods to obtain insights about different corpora at scale. We will experiment with different options for text pre-processing and their impact on visualizing term distributions in a text. We will also look beyond just observing differences in term distributions to statistically quantifying these differences and exploring the variation in content of the two corpora.

1.1 KNIME pre-requisites

We will need an extra KNIME extension to complete this Lab. In the top right, click "Menu" → "Install Extensions" and search for **KNIME Textprocessing**. Go though the prompts to install.

1.2 Text Pre-Processing and Zipf's Law

Goal: In your lectures, you should have seen that the frequency of term occurrence in a corpus follows **Zipf's law** (the n-th term frequency is inversely proportional to n). In this section of the lab, we will explore this law by plotting the term frequency of sections from the Bible and Quran (translated to English). We will also investigate the impact of different types of text processing, specifically:

1. **Tokenization** – convert text into tokens by removing punctuation and numbers
2. **Case folding** - convert all characters to lowercase
3. **Stopping** – remove English stop words
4. **Normalization** – stem the words

Q: Why is text pre-processing necessary?

Try to think about the impact of each of these steps and the tradeoffs of each choice before implementing them in KNIME. Can you think of specific

applications or domains where certain types of pre-processing would be preferred over others? Consider different domains (e.g. social media vs. academic texts), different downstream use-cases (e.g. understanding broad topics vs. grabbing exact hashtags).

1.2.1 Data

We will be comparing two corpora – the Bible and Quran (translated to English). Download the following files in .txt format.

- Bible: download from here
- Quran: download from here

1.2.2 Basic KNIME Workflow for Visualizing Term Distributions

1. Make sure you have installed the KNIME Textprocessing extension (see Section 1.1 on pre-requisites)
2. Add two CSV Reader Nodes and for each, read in the Bible or Quran text. As these files are .txt, we need to configure the CSV Reader node to allow them to be read in. In the node's configurations, change the column delimiter from "," to "\n" and uncheck the box "has column header". Please now execute the node.
3. Add a String Cleaner node, configure it to remove punctuation (**tokenization)**, remove numbers and lowercase the text (**case folding**). This node takes care of the first three pre-processing steps.
4. Add a String to Document node to convert the text into a data format that can be used by downstream nodes.
5. Add a Stop Words Filter node (**stopping)** and select the default built-in English stop words list to remove. Execute the node. **Q:** What are some of your observations about the resulting final documents? Is anything surprising? Can you think of pros/cons of stop word filtering for retaining the meaning of the text?
6. Add a Stemmer node (**normalization)**. You can start first with just a Porter Stemmer and, if interested, later explore other types of stemmers such as the Snowball Stemmer). In configuring the node, make sure you uncheck "Replace column" and instead create a new column with a name like "Preprocessed Document Stems" so you can clearly observe what the Stemmer does to the input text.
7. Add a Bag of Words Creator node, and set the Document column to whatever you named the output of the Stemmer from the previous step (e.g. "Preprocessed Document Stems"). This node will create a new row for term in each document (one row per term per document). At this point

we want to clean up our table (so the subsequent nodes work correctly) so make sure that in you configure the note to exclude all columns except "Preprocessed Document Stems" in the output table.

8. Add a TF (Term Frequency) node. Open dialog and in "Document Col" make sure the stemmed document data column is selected. In TF options uncheck "Relative frequency" – we want total counts, not percentages. The resulting "TF abs" column will represent how often the term for that row appears in the document. You will notice that all of the terms are scaled by a factor of 2. This is due to a quirk of KNIME (for the really curious, see here). We don't want this so simply add an extra node to divide everything by 2.

9. Add a Math Formula node and configure it to perform `$[‘‘TF abs"]/2`, choose to "Append" and output column that you call something like "tf_true".

10. Add a GroupBy node, set the Group to the Term column (Groups Tab) and then change to Manual Aggregation and make sure you select tf_true column with Sum.

11. Add a Rank node and sort the new Sum(tf_true) column in **descending** order. Are the most frequent terms as you expect? Is something surprising? Can you already see some of the downstream effects of the pre-processing strategy that was used?

12. Add a Scatter plot and visualize the Rank against Sum(tf_true). This represents the relationship between a term's rank and its frequency in the corpus. Is the pattern what you expect based on Zipf's law?

13. Let's try to plot the relationship in log space. To do this we attach the Rank node to a "Math Formula" node, which we configure to output two new columns. Configure two new expressions (`log10($["Sum(tf_true)"])` and `log10($["Sum(Rank)"])`) and create columns called log(freq) and log(rank).

14. Now add another Scatter Plot node and plot log(rank) against log(freq). What is the relationship between the variables in log space?

15. To find the best fitting slope between the variables in log space, add a Linear Regression Learner node where the target is log(freq) and the values used to learn include only log(rank). Execute the node and look at the Coefficients and Statistics tab. How does the coefficient tie back to the Scatter Plot in log space?

Now you have a basic workflow ready for the corpus you chose! Repeat the steps for the other text. Can you observe anything interesting that differs in the text patterns of the Bible compared to the Quran?

1.2.3 Exploring the Impact of Pre-Processing

What do you think will happen to the text distribution if the pre-processing steps are changed? Revisit your workflow and play around with the following:

- In Step 3, we added a String Cleaner node, which allowed us to remove punctuation, remove numbers and lowercase the text. Try removing combinations of these techniques and observe what happens. What can you conclude about the importance of these steps?
- In Step 5, we added a Stop Words Filter node. Try removing it and re-run the pipeline. You should see new terms become the most frequent ones. Inspect the KNIME English stop-words list. For our corpora, is this list appropriate? Can you think of any modifications that might be useful?
- in Step 6, we added a Stemmer node and chose the Porter Stemmer. Try a different Stemmer (e.g. Snowball) and observe the results. For some background on stemming and different algorithms, you can take a look at this IBM page.

1.3 Comparing Corpora

We can only draw limited insights from observing the term frequency distributions. In this section we will explore using two different methods (Chi-square metric and Topic Modeling / Latent Dirichlet Allocation) to analyze how the Bible and Quran compare in subject matter.

1.3.1 Chi-Square KNIME Workflow

In class we have covered the **Chi-square (χ^2) metric** and how it can be used to find terms that are **maximally informative** in separating two corpora. Let's implement it in KNIME with the following steps:

1. Replicate the text processing pipeline from section 1.2.2 (CSV Reader node $\rightarrow$ ... $\rightarrow$ Group By node) for both the Bible and the Quran. In KNIME you can select a rectangular area consisting of all the nodes you want to duplicate and simply copy + paste. Then simply change the source document in the CSV Reader nodes and set the last node to execute - all the nodes will execute in the required order.
2. To each of your two pipelines, add a Constant Value Column Appender node. Append a new Column of type String called "Corpus" and set it to Bible or Quran, respectively. We will use this column to differentiate where the data is coming from, as in the next step we will be concatenating the tables.
3. Combine the tables with a Concatenate node. Select the Union of all columns and create a new RowID.

4. Add a Pivot node to get the counts of the union of all term from each corpora. In the pivot node the group column is the "Term", pivot by "Corpus", and aggregate by the Sum of the Sum(tf_true) column.

5. Add a Missing Value node (we are taking the union of all terms in both corpora, so whenever a term appears only in one, there will be a missing value for the other) and add a "Fix Value" of 0 to the Bible+Sum(Sum(tf_true)) and Quran+Sum(Sum(tf_true)) columns.

6. To make life simpler we will rename the columns to something more interpretable. Add a Column Renamer node and rename the Bible+Sum(Sum(tf_true)) node to Bible-freq and Quran+Sum(Sum(tf_true)) to Quran-freq.

7. We need to get the total count of all terms in each corpora for the Chi-square formula. Add a GroupBy node, leave Groups empty, and manually aggregate to get the Sum of both Bible-freq and Quran-freq columns. Record these values as Bible-N and Quran-N.

8. Go back to the Column Renamer node and connect it to the Math Formula one. This is where we will have to manually write the Chi-square formula. For each term, we get its frequency in the Bible and Quran and calculate the difference between how often it appears in each vs how often we expect it to appear.

```
(((Bible-N + Quran-N) * (($Bible-freq$ * (Quran-N - $Quran-freq$)) -
    ($Quran-freq$ * (Bible-N - $Bible-freq$)))^2)
/
(($Bible-freq$ + $Quran-freq$) * ((Bible-N - $Bible-freq$) +
    (Quran-N - $Quran-freq$)) * Bible-N * Quran-N))
```

   Replace Bible-N and Quran-N with the numbers you recorded from the previous step. Append a new column for the result and call it chi-2. Execute the node and sort the terms by the new chi-2 column values.

Which terms receive the highest chi-2 score? Does this correlate with what you expect to be the most informative terms for differentiating the Bible from the Quran? How could have pre-processing affected the results?

1.3.2 Topic Modeling in KNIME

One way to explore the themes present in the text data is to use Topic Modeling. This method surfaces key topics discussed throughout the text. One such topic modeling algorithm is Latent Dirichlet Allocation (LDA) and you can apply this to your data using the LDA node in KNIME.

Let's implement it:

1. You will again have two parallel workflows for the two different corpora. This time we don't need the Bag of Words and TF nodes. Remember that you can copy and paste nodes in KNIME to preserve the settings. You will need the following pipeline: CSV Reader → String Cleaner → Strings to Document → Stop Word Filter → Porter Stemmer → Constant Value Column Appender. Revisit instructions from previous sections on each of these nodes.

2. Combine the Bible and Quran workflows with a Concatenate Node, similar to before.

3. Add a Topic Extractor (Parallel LDA) Node. Choose the right Document Column ("Preprocessed Document Stems") and set the node to execute. It will use a statistical algorithm to discover latent topics based on word co-occurrence.

After the LDA node finishes executing, explore the resulting Topic Terms for each of the learned topics. Do the topics make sense? **LDA is very sensitive to the chosen hyperparameter settings.** Re-run the node and play around with the **number of topics** and **number of words** per topic settings. How are the results affected?

In the current workflow we are lumping the two corpora together to learn joint topics. Try to get separate topic distributions by instead running LDA on each corpora in isolation. What are the pros and cons of doing topic modeling on the Bible and Quran together vs. separately?

1.3.3 Stretch Task: Using a Larger Corpora

Try to run the pipelines we have just covered on a larger corpora. We will use substantially larger dataset of Wikipedia abstracts: download it from here. You can insert the Row Sampler node between the CSV Reader and String Cleaner in order to sample a smaller subset of the greater abstracts set (set Relative Size to 10 to randomly sample 10% of the data). Then follow all the same steps from the previous sections with this new dataset. How does using a larger dataset affect the observed results?

1.3.4 Takeaways

What different insights can you get from using chi-2 vs topic modeling to compare two different corpora? How does pre-processing affect the anlaysis?