Lecture 2: Introduction to Text Mining

Ben Shepherd, Principal.

Ben@Developing-Trade.Com

Key Takeaways

1. In principle, text is just as much data as numbers are.

2. But working with it requires particular tools. Some are very intuitive (word counts, frequency measures), others require a little more analytical work to understand (TF-IDF), and some are sophisticated (topic mining using LDA).

3. R contains a range of tools that make it (relatively) easy to work with text, ranging from importing data, to cleaning, to analysis.

4. Text as data applications are still rare in international trade, so there is huge scope to add to the policy literature.

5. Understanding how to work with text is a pre-requisite to using it for other purposes, such as predictive modeling or classification.

Outline

1. What is Text Mining?
2. Basic Tools and Workflow of Text Mining.
3. Demonstration in R: Free Trade Agreements

1. What is Text Mining?

- As economists, we are used to using quantitative data as the raw material for empirical work: we look at numbers as data.

- Text mining takes a different approach: written text is the raw material for empirical work. We treat text as data.

- We've all done some simple text mining:
  - Counting instances of particular words or combinations in a document.
  - Classification of groups of documents by main topic.
  - Analysis of the sentiment contained in groups of words.
  - Linking words to observed quantitative data.

- Our challenge now is to formalize all of this, and structure a workflow that allows us to work with text documents (nearly) as easily as we are used to working with quantitative datasets.

1. What is Text Mining?

- How is text mining linked to ML?
- Remember that ML's leading use is as a prediction tool.
- So what if we could take a text input, treat it as data, and use it to predict something else?
  - Example 1 (frequently done): Analyze tweets or breaking news for sentiment regarding particular stocks, then predict short-term performance based on those sentiments.
  - Example 2 (in progress): Analyze the text of regional trade agreements and use it to predict their trade creation effects, derived from a quantitative model.
  - Example 3 (in progress): Take short texts from experts describing NTMs and use them to classify the measures by category, based on previous associations between text and classification. Even better: use full domestic laws!
- So just as we use quantitative data to predict and classify, so too can we theoretically use text for that purpose.
- But first: we need to get used to working with text, and develop some basic tools!

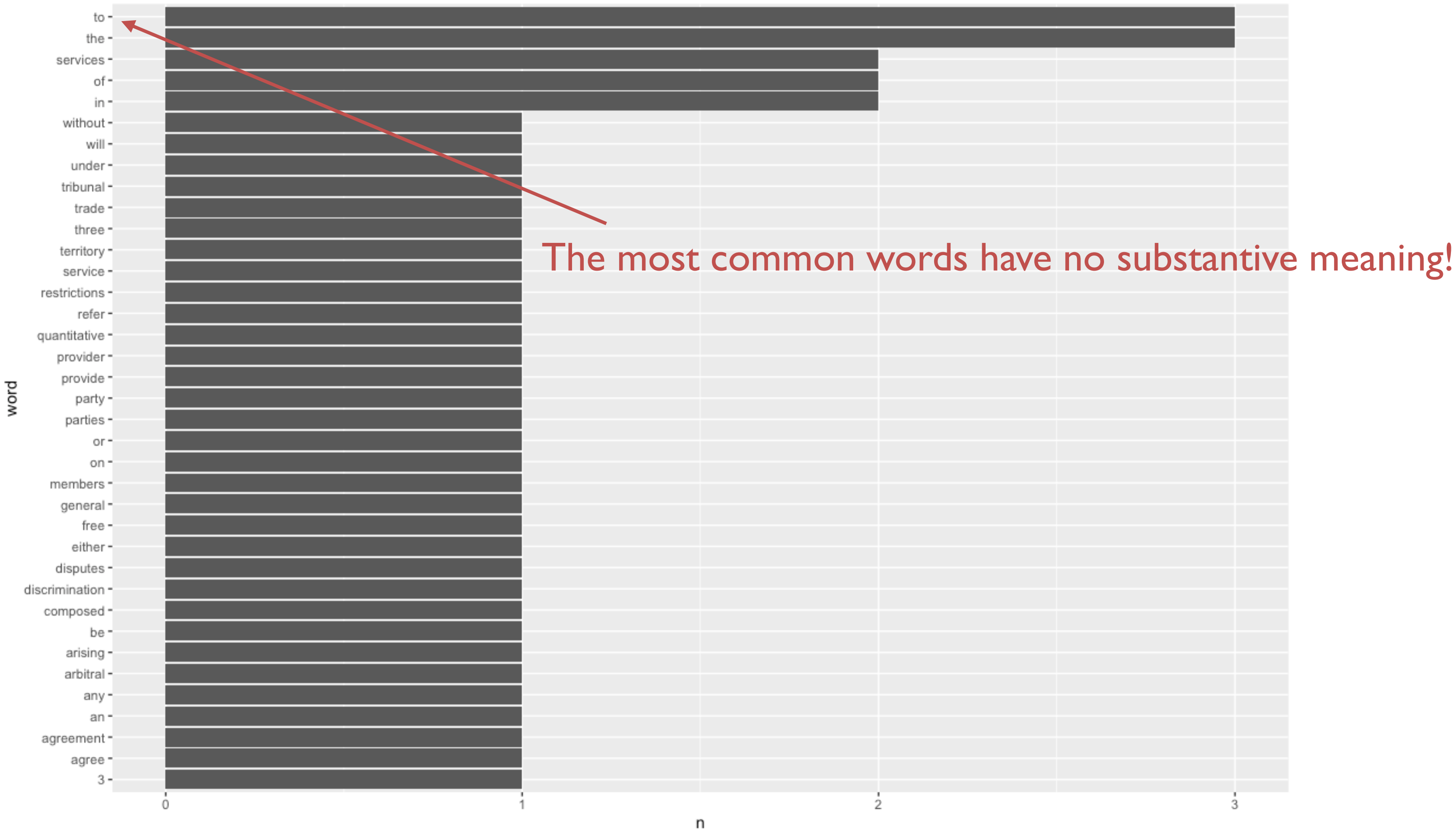
2. Basic Tools and Workflow of Text Mining

- First, some terminology:
  - Token: a meaningful unit of text, typically a word.
  - N-gram: a group of n words occurring together.
    - ”Asia” is a word.
    - “Asia and the Pacific” is an n-gram (4-gram).
  - Stop words: words that add little meaning, such as “a”, “the”, etc.
  - Lemma: the meaningful part of a word, apart from grammatical function (e.g., “running” and “run” both have the lemma “run”.)
  - Sentence: typically a collection of tokens.
  - Document: a collection of tokens (or sentences).
  - Corpus: a collection of documents.
  - Bag of Words Model: assume word order does not matter.

2. Basic Tools and Workflow of Text Mining

- Imagine we have two documents, that could be articles of a trade agreement for example:
  - "The parties agree to refer disputes arising in the territory of either party to an arbitral tribunal composed of three (3) members."
  - "Any service provider under the General Agreement on Trade in Services will be free to provide services without quantitative restrictions or discrimination."
- Let's combine these into a single corpus, and analyze word counts.
- To do this:
  - Combine the documents into a single data unit.
  - Tokenize the corpus, so that we have (basically) a vector of words.

2. Basic Tools and Workflow of Text Mining

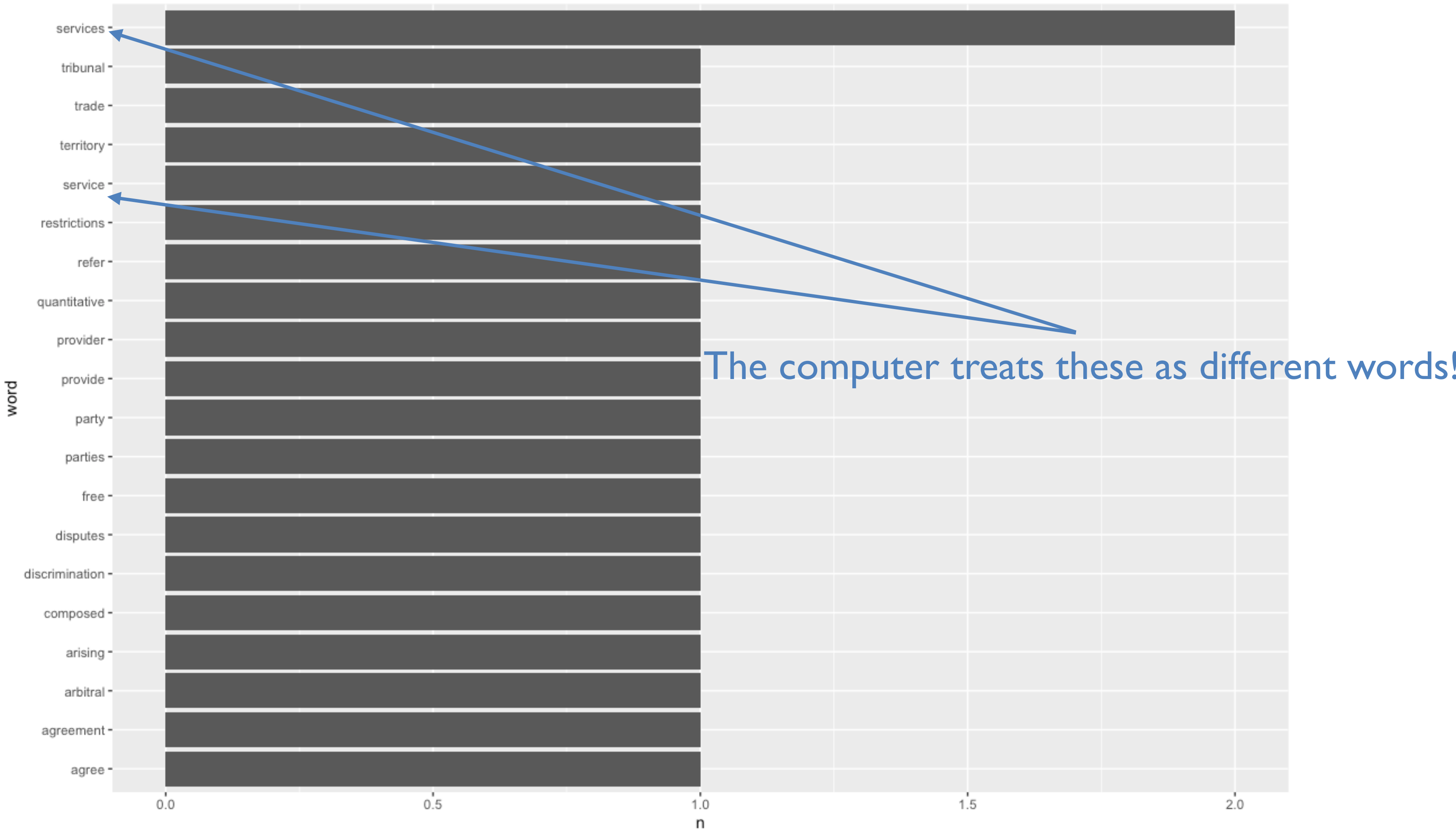


2. Basic Tools and Workflow of Text Mining

- Conceptually, removing stop words and numbers is relatively straightforward.
  - A nice person compiles a dictionary of stop words (or we tell the computer to identify numbers).
  - We simply merge the corpus with the dictionary, and delete words that appear in both (or delete words that are in fact numbers).

- To implement it practically, the answer is what it usually is: "there's a package for that!".

2. Basic Tools and Workflow of Text Mining



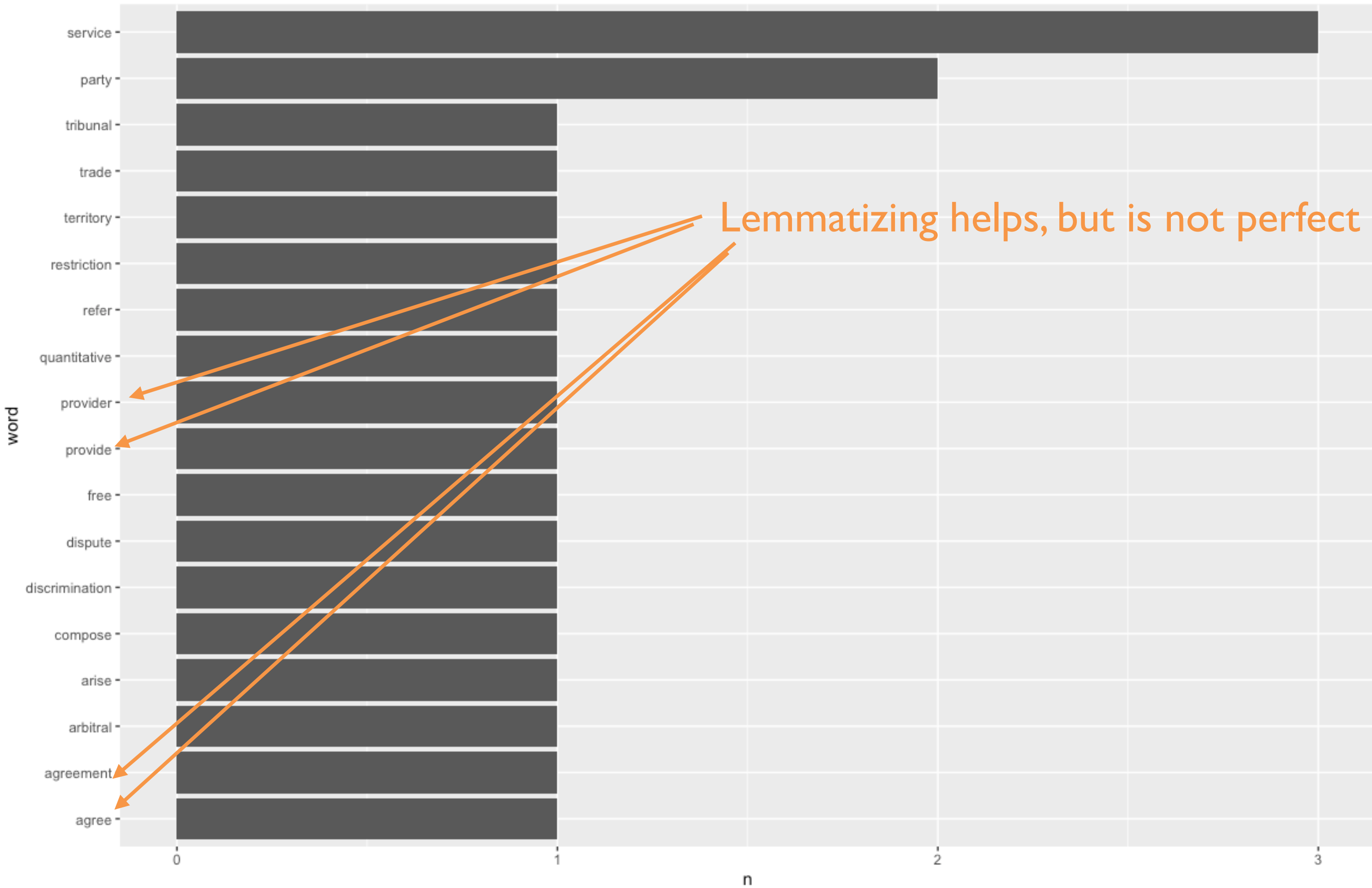
2. Basic Tools and Workflow of Text Mining

- To count "party" and "parties" or "services" and "service" as single words, we need to lemmatize the text.

- Lemmatizing is much harder than removing stop words, because it requires detailed linguistic analysis to link words to their stem of meaning.

- Again, "there's a package for that!". But don't expect it to work perfectly; may need to experiment with different packages to get the result we're looking for.

2. Basic Tools and Workflow of Text Mining



2. Basic Tools and Workflow of Text Mining

- The simplest metric in text mining is simply frequency: how many times do different tokens occur in a document?
  - To compare across documents, we typically need to adjust for document length: count(token)/count(all tokens). This gives a true frequency measure.
  - Once we have frequencies, we can get a very basic sense of how different documents pay attention to different factors.
  - Simple, transparent, intuitive measure.

- What about extending this approach to develop a simple classification metric? Can we use some version of a frequency measure to identify documents in a corpus that are more or less "about" particular words?
  - Frequency gets part of the way there, but we need to adjust it to provide a clearer measure of focus in a document.
  - Frequent words, even if many are extracted by brute force as stop words, may be uninformative as to topic. In a trade agreement, think of "article" or "party".

2. Basic Tools and Workflow of Text Mining

- This is where the TF-IDF metric comes in = Term Frequency – Inverse Document Frequency.
  - TF: Count the number of times a token appears in a document and divide by the total number of tokens in a document.
  - IDF: Divide the number of documents in the corpus by the number of documents where a token appears. (Sometimes scaled by taking the logarithm.) Also known as "specificity".
  - Calculate TF * IDF.

- Intuitively, consider a token that appears commonly in document 1 and is included in most documents in the corpus: TF for document 1 is high (and TF for other documents is high) and log(IDF) is close to zero, so TF-IDF is small.

- Now consider a token that appears commonly in document 2 and is included in very few documents in the corpus: TF for document 2 is high (and TF for other documents is low), and log(IDF) is large, so TF-IDF is large.

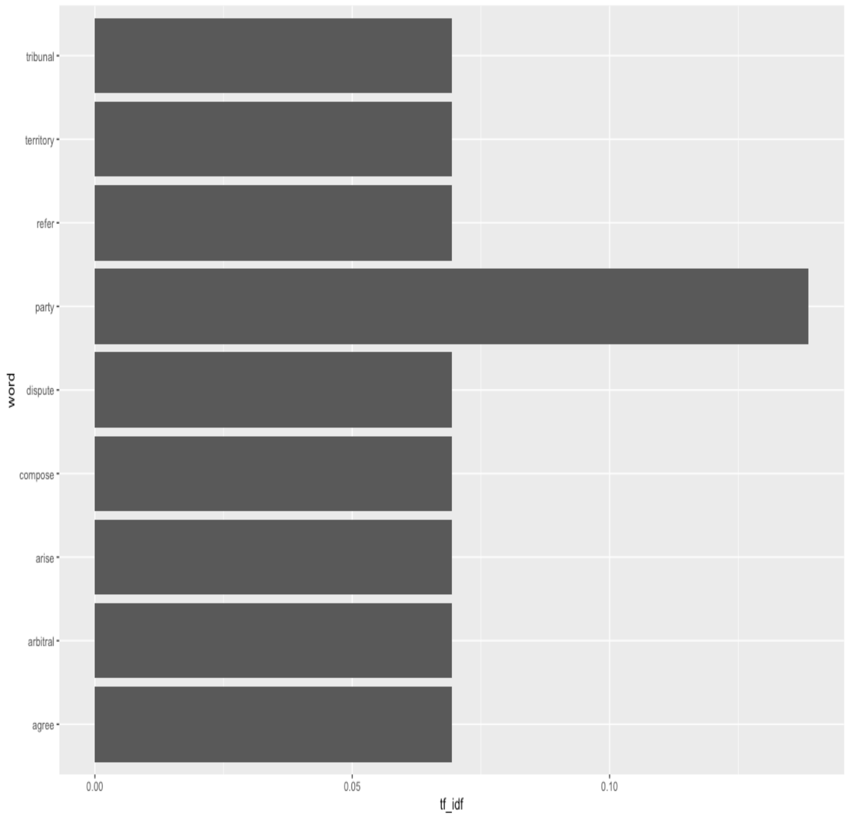
- Looking at TF-IDF across tokens, we can see that it is doing more or less what we want: pulling out documents that are plausibly "about" a term, in that they use it frequently compared to both other terms and other documents.

2. Basic Tools and Workflow of Text Mining

Document 1

- "The parties agree to refer disputes arising in the territory of either party to an arbitral tribunal composed of three (3) members."

TF-IDF

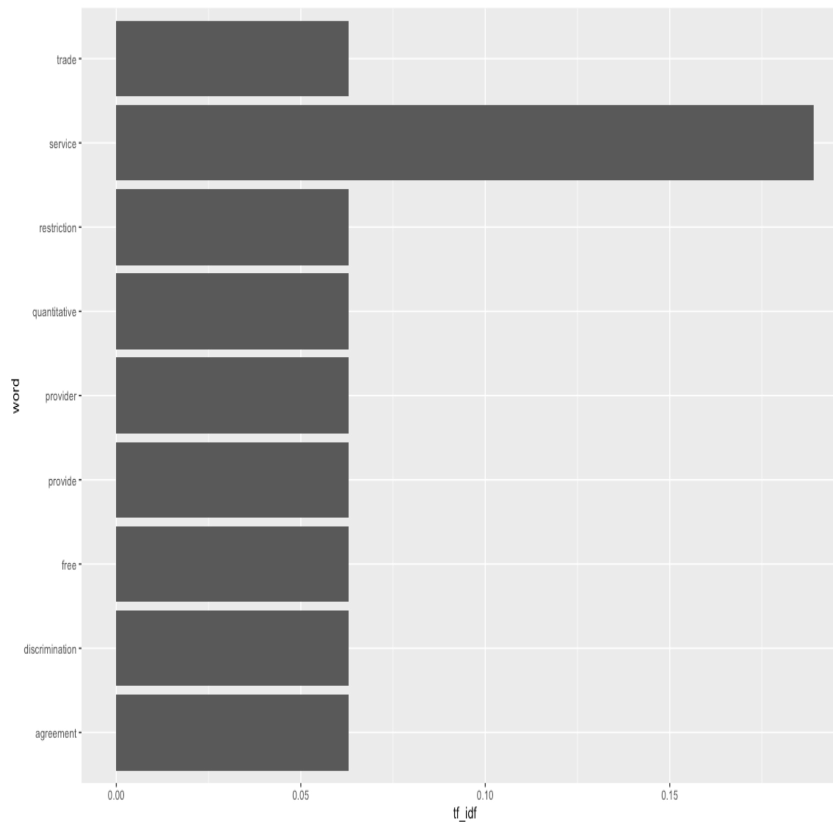


2. Basic Tools and Workflow of Text Mining

Document 2

- "Any service provider under the General Agreement on Trade in Services will be free to provide services in the territory of either party."

TF-IDF



2. Basic Tools and Workflow of Text Mining

- This is a very simple example, but TF-IDF seems to help identify Document 1 as being about disputes and arbitration, while Document 2 is about services.

- In real world examples, there are many more words and combinations, so it typically takes a good amount of work to analyze TF-IDF in depth.

- But calculating it is easy ("there's a package for that!) and interpretation is also straightforward. Typically easy to explain to policy audiences and non-technical people.

2. Basic Tools and Workflow of Text Mining

- A more sophisticated tool for topic modeling is Latent Dirichlet Allocation (LDA).
  - Assume a corpus is made up of k topics that are latent (hidden), to be revealed as we do the modeling.
  - Assume each topic is made up of tokens.

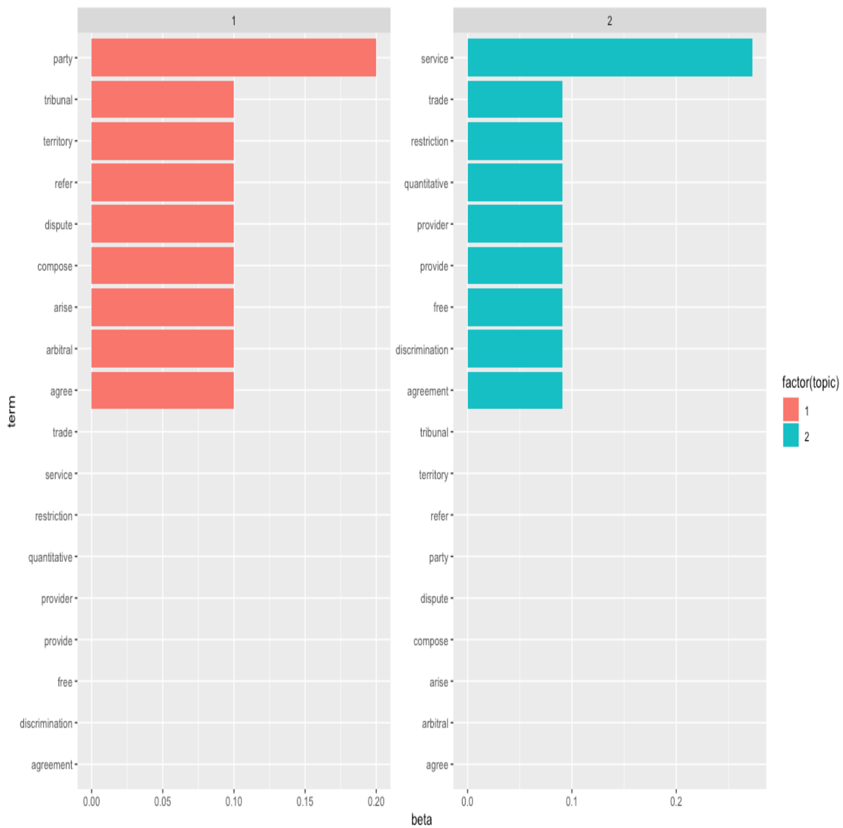
- The algorithm maps documents to topics such that the words in each document are mostly captured by the topics.

- It is a numerical procedure that proceeds iteratively:
  - Assign a word to a topic.
  - For each word in a document, assume its topic is wrong but that others are correct.
  - Probabilistically assign the word a topic based on the topics in the document, and the number of times the word is assigned that topic in the rest of the corpus.
  - Repeat!

- Outputs:
  - "Betas": Per-topic-per-word probabilities. High beta for a word indicate that it is strongly associated with a particular topic, i.e. a high probability of being generated by that topic.
  - "Gammas": Per-document-per-topic probabilities. High gamma for a topic indicates the estimated proportion of words in a document that are generated from that topic.

2. Basic Tools and Workflow of Text Mining

Betas



Gammas

- Document 1:
  - 99.8% topic 1.
  - 0.2% topic 2.
- Document 2:
  - 0.2% topic 1.
  - 99.8% topic 2.

2. Basic Tools and Workflow of Text Mining

- LDA is conceptually much more complex than the other tools we've looked at.

- To implement it is (relatively) straightforward because… "there's a package for that!".

- In practice, it is rarely as straightforward as in the example:
  - Texts rarely so polarized between topics.
  - We typically don't know k in advance, so we have to experiment with different values.

- A little like PCA in basic statistics: we produce measures but the trick is in interpreting the "loadings" (betas and gammas).

2. Basic Tools and Workflow of Text Mining

- A typical workflow for text mining is therefore:
  - Import the text and put it into a format we can work with (sometimes easy, sometimes hard).
  - Tokenize.
  - Remove stop words.
  - Lemmatize.
  - Analyze descriptive statistics:
    - Word counts.
    - Frequency measures.
    - TF-IDF.
  - Compare relevant measures across documents within a corpus.
  - Undertake topic modeling (LDA), if relevant.
  - Next lecture: use the text for prediction or classification.

3. Demonstration in R: Free Trade Agreements

- Using R to work with text is relatively straightforward.

- For importing:
  - CSV is straightforward with tidyverse.
  - For XML, use the XML library. Typically two steps:
    - xmlParse to import the text into a specific object class.
    - xmlToDataFrame to transform it into a dataframe.

- For tokenizing: TidyText has the unnest_tokens function: send it a dataframe with one document per row, and it returns a dataframe with one word per row (retaining the document index).

- For removing stop words: TidyText has a stop_words data object, so simply merge with the tokenized dataframe, and remove rows appearing in both.

- For lemmatizing: TextStem has the lemmatize_words function, but be careful as it receives and sends vectors, not dataframes (i.e., work with columns, not full objects).

3. Demonstration in R: Free Trade Agreements

- For word counts and frequency: use count in the Tidyverse.
- For TF-IDF: TidyText has bind_tf_idf, which works straightforwardly with dataframes.
- For LDA:
  - Use the tm function cast_dtm to put the data in DocumentTermMatrix form.
  - Use the topicmodels LDA function to run LDA.
  - Use tidy to easily extract betas and gammas.
- For presenting results: manipulate the data and send it to ggplot!
- Working effectively with text requires reasonable programming skills to get data and results in the right formats, even though most of the tools are conceptually simple.
- Start with simple texts, then work up to more complex collections.

3. Demonstration in R: Free Trade Agreements

- To see all of this in action, let's use UNCTAD's XML repository of the text of trade agreements.
- The first two agreements in series are Japan-Thailand, and Egypt-Turkey.
- We'll focus on Japan-Thailand, but will also compare texts between the two treaties.
- What are the agreements "about"? To what extent do they talk about similar things? Is one agreement more focused on particular areas than the other?
- Once the structure is in place, this kind of analysis can easily be scaled up to look at the whole repository.
- We've seen the workflow, now let's code it…

Key Takeaways

1. In principle, text is just as much data as numbers are.

2. But working with it requires particular tools. Some are very intuitive (word counts, frequency measures), others require a little more analytical work to understand (TF-IDF), and some are sophisticated (topic mining using LDA).

3. R contains a range of tools that make it (relatively) easy to work with text, ranging from importing data, to cleaning, to analysis.

4. Text as data applications are still rare in international trade, so there is huge scope to add to the policy literature.

5. Understanding how to work with text is a pre-requisite to using it for other purposes, such as predictive modeling or classification.

Additional Resources

- Introduction to text mining for economists using R, with an application to Central Bank texts: https://scholar.harvard.edu/files/jbenchimol/files/text-mining-methodologies.pdf.
- Overview of "text as data" applications in economics: https://web.stanford.edu/~gentzkow/research/text-as-data.pdf.
- Excellent introduction to and overview of Tidytext in R: https://www.tidytextmining.com/index.html.
- UNCTAD's text as data analysis of trade agreements, with links to the XML archive: https://unctad.org/topic/trade-analysis/text-as-data.