Exploiting Synonym Choice to Identify Discrete Components of a Document

Navot Akiva, Idan Dershowitz and Moshe Koppel

Separating Document Components

- Often documents consist of multiple authorial components.
- Our object is to tease apart the components of a composite document.

Basic Idea

- Divide the document into natural chunks (e.g., chapters, paragraphs)
- Vectorize chunks using some feature set
- Cluster the vectors

Classic Example: The Bible

- Great historic and cultural interest
- Much prior research on components
- Has been manually tagged in every conceivable way

Obligatory Disclaimer

- We're not wading into religious territory here.
- That there is some optimal clustering is tautologous.
- That there is some very convincing clustering is of interest to traditionalists and critics alike.
- Why there is such a convincing clustering is not our concern here.

Test Case

Let's munge Ezekiel and Jeremiah and see if we can separate them out.

- Each is presumably the work of a single author.
- There's no reason to think it's the same author.
- Some of the differences between them parallel differences across different sections of the Pentateuch.

Clustering Jeremiah+Ezekiel

- Chunks = chapters (no sequence info)
- Features = bag of words
- Cluster method = Ncut
- K=2

Results using all words

	Jeremiah	Ezekiel
Cluster 1	29	28
Cluster 2	23	20

- Not too exciting. We must be picking up thematic or genre-related differences that cross books.
- Let’s try using only function words.

Results using function words

	Jeremiah	Ezekiel
Cluster 1	34	28
Cluster 2	18	20

- Not any better.
- Let’s try a new approach.

A Better Idea

- Exploit the fact that different authors use different synonyms for the same idea (*e.g., makel/mateh*).
- It would be really convenient if it turned out that Jeremiah and Ezekiel made consistently different choices for various synsets.
- Note that words aren't synonyms, rather word <u>senses</u> are synonyms. (For example *mateh=staff* is a synonym of *makel*, but *mateh=tribe* is not.)

Automatically Finding Synonyms

- There are various clever methods for identifying synsets, but most are not exact enough for our purpose.
- Conveniently, for the Bible, we have many useful tools, including careful translations and manual sense tagging.
- We identify as synonyms word senses that are translated into the same English word (e.g., *makel=staff* and *mateh=staff*).
- Due to polysemy (in English), this method overshoots. We manually delete mistakes. (This is the only manual intervention we will ever do.)

Synonym Method

- The usual similarity measures (e.g., cosine, inverse Euclidean distance) don't make sense here.
  - If two docs use the same synonym, they are similar.
  - If two docs use opposite synonyms, they are different.
  - If one of the docs uses one of the synonyms, but the other doesn't, cosine would regard them as different. But are they?
- For measuring similarity, we only consider synsets represented in both docs.

Results using synonyms

	Jeremiah	Ezekiel
Cluster 1	46	6
Cluster 2	7	41

- Now we’re getting somewhere.
- But we’re not done yet…

Core of a Cluster

- Some chapters are in a cluster because they really belong there; some just have to be somewhere.
- Let's consider only chapters near one centroid and far from the other. These are the "cores" of the respective clusters.
- Let's also consider only synsets that are used differently in the two cores. These are "differentiating" synsets.
- Now cluster again using only cores and differentiating synsets. Iterate as desired.
- This converges quickly to stable cores and differentiating subsets, i.e., a reliable (but partial) clustering.

Cluster cores

	Jeremiah	Ezekiel
Cluster 1	36	2
Cluster 2	0	36

Cluster cores

	Jeremiah	Ezekiel
Cluster 1	36	2
Cluster 2	0	36

Ezekiel 1, 10

Expanding the Core

- Now that we have a core, we can use supervised methods (e.g., SVM) to learn a boundary.
- In fact, we can use function words as our features.
  - Using synonyms will just get us back where we started.
  - And besides, FW are generally very reliable for supervised authorship attribution.

SVM expansion of core

	Jeremiah	Ezekiel
Cluster 1	52	1
Cluster 2	0	47

- Two training examples are “misclassed” by SVM.
- Incredibly, these are Ezekiel 1 and 10, which were part of Jeremiah core, but are on Ezekiel side of optimal SVM boundary.
- The only exception is Ezekiel 42, a non-core chapter which lies in the SVM margin.

So what about the Pentateuch?

- Let's just apply the exact same process that worked on Jeremiah+Ezekiel to the Pentateuch.
- One crucial caveat:
  - In Jeremiah+Ezekiel our units (chapters) were all pure Jeremiah or pure Ezekiel; we have no such guarantee for the Pentateuch
  - We have some beautiful methods for using synonym distribution to automatically identify component boundaries

So what about the Pentateuch?

- Let's just apply the exact same process that worked on Jeremiah+Ezekiel to the Pentateuch.

- One crucial caveat:
  - In Jeremiah+Ezekiel our units (chapters) were all pure Jeremiah or pure Ezekiel; we have no such guarantee for the Pentateuch
  - We have some beautiful methods for using synonym distribution to automatically identify component boundaries

But I'm out of time….

Clustering the Pentateuch

- For two clusters, our method gives something very close to scholars' split between P and non-P.
- But chapters of Genesis commonly assigned to P are in the non-P cluster.
- Clustering of the non-P cluster does not give anything like the scholars' split between J and E.