A Generic Unsupervised Method for Decomposing Multi-Author Documents

Navot Akiva and Moshe Koppel (Corresponding Author)
Dept. of Computer Science
Bar-Ilan University
Ramat-Gan, Israel
moishk@gmail.com   navot.akiva@gmail.com

Abstract.

Given an unsegmented multi-author text, we wish to automatically separate out distinct authorial threads. We present a novel, entirely unsupervised, method that achieves strong results on multiple testbeds, including those for which authorial threads are topically identical. Unlike previous work, our method requires no specialized linguistic tools and can be easily applied to any text.

1. Introduction

Imagine that we are faced with a multi-author document and wish to delineate the contributions of each author. For example, literary scholars are often faced with documents of historical significance that appear to be composites of multiple earlier texts and strive to tease apart the document's various components. A well-known example of this phenomenon, which we will consider here, is the Hebrew Bible. Contemporary examples include analysis of collaborative online works in which one might wish to identify the contribution of a particular author for commercial or forensic purposes.

We assume here that our only input is the composite document itself (as well as the number of authorial components into which we wish to decompose the document). We know nothing about the authors and are not supplied with any identified samples of any author's writing. Thus, our methods must be entirely unsupervised.

In the first part of this paper we consider an easier version of the problem we wish to solve. In this version, the document to be decomposed is given to us segmented into units, each of which is the work of a single author. The challenge is only to cluster the units according to author. The purpose of considering this simpler version of the problem is to establish certain conclusions that will be useful for the second part of this paper in which we consider a more difficult version of the problem. In this more difficult version, we are given an unsegmented document and the challenge includes segmenting the document as well as clustering the resulting units. We will show how our method for solving the easier version can be extended to handle this more difficult problem. We will prove the efficacy of our methods on three separate testbeds.

2. Previous Work

Several active areas of research are closely related to the problem we attack in this paper. Clustering a set of anonymous documents according to authorship has been explored recently in the work of Layton, Watters & Dazeley (2011). However, this work differs from ours in that its objective is to cluster multiple documents, each of which is written by a single author. By contrast, we wish to segment a single unsegmented document according to authorship, with no guarantees that any unit longer than a single sentence is written by a single author.

The work of Graham, Hirst & Marthi (2005) considered the problem of decomposing a document according to authorship, but their method was a supervised one: they had labeled examples of same-author and different-author pairs of paragraphs by the two authors. We have no such labeled examples; indeed our method is entirely unsupervised.

Some research has considered the problem of clustering topics and authors using Latent Dirichlet Analysis (Rosen-Zvi, Chemudugunta, Griffiths, Smyth & Steyvers, 2010). However, that work assumes that each author's work reflects a distinct distribution over topics. In this paper, we explicitly reject that assumption. In fact, in at least one of our testbeds, the respective distributions over topics for the authors are identical.

In addition, some work on intrinsic plagiarism detection, for example Meyer zu Eissen and Stein (2006) and Stamatatos (2009), also attempts to segment a document according to authorship, but it assumes an asymmetric split: there is a single dominant author and the object is to identify outlier segments. We make no such assumption.

Finally, we note that this paper extends the work of Koppel, Akiva, Dershowitz & Dershowitz (2011). That work, however, required special information regarding synonym usage that is available only for carefully tagged corpora such as the Bible. The methods proposed here are completely generic and can be applied to any composite document.

3. A Preliminary Problem: Clustering Chapters

For expository purposes, we first consider a preliminary problem that is considerably simpler than the problem we wish to solve. Suppose we are given a collection of (unlabeled) chapters taken from the works of multiple authors. Our object is to separate out the works of the respective authors. This problem is easier than the one we ultimately wish to solve since we are given the fact that each chapter is the work of exactly one author. In fact, it is basically the author clustering problem considered in Layton et al. (2011).

We will use a canonical example to motivate and illustrate our method for solving this problem. Following Koppel et al. (2011), we use the biblical books Jeremiah and Ezekiel (in the original Hebrew) as the basis for constructing a development corpus. Jeremiah and Ezekiel are two roughly contemporaneous books belonging to the same biblical sub-genre (prophetic works), and each is widely thought to consist primarily of the work of a single distinct author. Jeremiah consists of 52 chapters and Ezekiel consists of 48 chapters. For our preliminary problem, we are given all 100 unlabeled chapters and our task is to separate them out into the two constituent books. (For simplicity, let's assume that it is known that there are exactly two natural clusters.)

As a first try, the basics of which will serve as a foundation for more sophisticated attempts, we do the following:

1 Represent each chapter as a bag-of-words (using all words that appear at least k times in the corpus).
2 Compute the similarity of every pair of chapters in the corpus.
3 Use a clustering algorithm to cluster the chapters into two clusters.

We use k=2, cosine similarity and n-cut clustering (Dhillon, Guan & Kulis,2004). Comparing the Jeremiah-Ezekiel split to the clusters thus obtained, we have the following matrix:

Book	**Cluster I**	**Cluster II**
Jer	29	23
Eze	28	20

As can be seen, the clusters are essentially orthogonal to the Jeremiah-Ezekiel split. Ideally, 100% of the chapters would lie on the majority diagonal, but in fact only 51%

do. Formally, our measure of correspondence between the desired clustering and the actual one is computed by first normalizing rows and then computing the weight of the majority diagonal relative to the whole. This measure, often called *purity* (Manning, Raghavan & Schütze, 2008), runs from 50% (when the clusters are completely orthogonal to the desired split) to 100% (where the clusters are identical with the desired split). For the case of two classes, purity is equivalent to maximal macro-averaged recall where the maximum is taken over the two possible assignments of books to clusters. In this case, we obtain purity of 51.5%, barely above the theoretical minimum.

This negative result is not especially surprising since there are many ways for the chapters to split (e.g., according to thematic elements, sub-genre, etc.) and we can't expect an unsupervised method to read our minds. Thus, to guide the method in the direction of stylistic elements that might distinguish between Jeremiah and Ezekiel, we consider only the 500 words that appear in the greatest number of different chapters in the combined corpus. Features chosen in this way are the ones least likely to be tied to specific topics.

Repeating our experiment of above, though limiting our feature set to these common words, we obtain the following matrix:

Book	**Cluster I**	**Cluster II**
Jer	28	24
Eze	28	20

As can be seen, using generic words yields purity of 52%, which does not improve matters at all. Thus, we try a different approach.

We take now as our inspiration those features commonly used by scholars in the field to distinguish between distinct authorial threads in a text. Literary scholars often seek lexical stylistic markers that are sometimes used by one author but not used at all by the other author (or authors). Thus, we now consider as our feature set the same set of 500 common words as just considered but our vector representation of a text is *binary*: it records only if a feature appears in the text, but not its frequency.

Remarkably, using the exact same method as described above but with this binary vector representation, we obtain the following clusters:

Book	Cluster I	Cluster II
Jer	52	0
Eze	0	48

That is, if we ignore frequencies, the respective vocabularies of the two books are sufficiently distinct that we obtain perfect clustering.

We don't propose to draw far-reaching conclusions regarding document clustering from a single example. The sole purpose of this experiment is to establish the superiority of binary features for authorship clustering. We use this lesson as a springboard for handling the more difficult problem that is our main objective in this paper.

4. Main Problem: Decomposing Unsegmented Texts

We now turn to the problem in which we are given a single unsegmented text and need to identify the transitions from one author to another and to cluster segments according to author.

Formally, the problem we wish to solve is this: we have a single document written by k co-authors. We assume that any given sentence was written by exactly one of the authors and that generally the authors took turns writing reasonably long sequences of sentences. The number of authors, k, is known to us. Our objective is to segment the text and to cluster the resulting segments in accordance with the parts of the text written by the respective co-authors.

Below, we will formally describe a method for generating testbed documents on which we can test our methods for solving this problem. For now, we informally describe a single testbed document, which we will use for expository purposes. We artificially create a merged Jeremiah-Ezekiel text by choosing a random number of sentences from Jeremiah, then a random number of sentences from Ezekiel and so on, in alternating fashion. The random numbers are chosen from a uniform distribution over the integers between 1 and n, where n is a parameter that controls the granularity of the merged text.

5. Our Method

We first present the method in broad outline, leaving the details for the exposition below. Given a merged text and the desired number of authorial threads, k, do the following:

1 Chunk the text into segments of some fixed length c (where c is a parameter that might depend on the granularity of the text, if the granularity is known).

2 Represent each such chunk as a binary vector representing the presence or absence of each of the 500 most common words in the full text. (This draws on the lessons learned on the preliminary problem. As in that case, we use the 500 words that appear in the greatest number of chunks.)

3 Measure the similarity of every pair of chunks (or, more specifically, their vector representations).

4 Cluster the chunks into k clusters using some clustering algorithm.

Note that until this point the algorithm is identical to the one we used for the preliminary problem – except for the fact that in this case we need to first artificially chunk the unsegmented text. Of course, at this point we have clustered chunks each of which is not necessarily the work of a single author. Thus, we need to refine our initial clustering with a much more fine-grained clustering. We proceed as follows:

5 Regard each chunk as being labeled by its cluster assignment and select *some of* the labeled chunks to be used as training examples for supervised learning. In selecting training examples, we choose only those that are most confidently assigned to their respective clusters. Thus for each cluster, we rank all examples in the cluster in two ways. First, according to nearness to the centroid of the cluster. Second, according to the difference in distance from the centroid of the cluster and the minimal distance to any other cluster centroid. For each cluster, we select an example in that cluster only if it is among the p% best (i.e., close to the centroid of the cluster and far from the centroids of other clusters) examples according to each measure (where p is a parameter). For each cluster, we call the selected examples, the *core* of the cluster.

6 Use the selected examples as training examples for learning a classifier that assigns a text to one of the k classes. We use SVM as our learning method.

7 Use the learned classifier to tentatively assign each individual sentence in the merged text to one of the k classes.

8 Every sentence that is assigned to some class with high confidence is locked into that class. Each other sentence is assigned in accordance with the nearest confidently assigned sentences before and after it (as will be explained below). For the case of two authors, this amounts to just locking in the t% of sentences assigned to each class that are furthest from the SVM boundary (where t is a parameter).

The central idea is that we represent chunks of the document in terms of features likely to capture authorship with high precision and cluster accordingly. Then the cores of these clusters can be used as the basis for supervised learning (in conjunction with a wider range of features than used in the initial clustering) which yields a classifier that can be applied to individual sentences – thus resulting in the sort of fine-grained division that we seek.

The full algorithm is summarized in Figure 1.

[Insert Figure 1 here]

Note that there are three parameters in our method: the chunk length, c; the percentage of chunks used as training examples, p; the percentage of sentences locked in according to the SVM assignment, t.

For clarity and specificity, let's run through the method as applied to the artificially merged Jeremiah-Ezekiel document. This document will also serve as a development corpus for optimizing the parameters.

Creating the merged document

Jeremiah consists of 1364 sentences and Ezekiel consists of 1273 sentences. (Biblical sentences are usually referred to as "verses", but for simplicity we just call them sentences.) We create a merged text by alternately taking sentences from each author, the number of sentences drawn from a uniform distribution between 1 and 200. In the resulting merged text, there are 27 transitions from one author to the other, with a median of just above 100 sentences between transitions, as we'd expect. (Below we will consider the impact of using documents of greater – or lesser – granularity.)

Chunking and vectorization

We begin by chunking the text. The number of sentences per chunk (our parameter c) must be chosen so that each chunk is sufficiently long but so that at the same time there is a sufficient number of chunks. For the case of the Jeremiah-Ezekiel document, we set the parameter c to 30. (Results for various choices of chunk size are shown in Table 1 below.) Thus, we obtain 88 chunks of 30 sentences each (except for the last chunk which has only 27 sentences). Of these chunks, 34 are pure Jeremiah, 31 are pure Ezekiel and the other 23 are mixed.

We now represent the chunks as vectors. As noted, our feature set consists of the 500 most common words in the document and each chunk is represented as a binary vector indicating whether the corresponding word does or does not appear in the chunk.

Clustering

Next we use cosine to measure the similarity between each pair of chunks and use n-cut clustering (Dhillon et al., 2007) to cluster the chunks into two clusters, consisting of 44 documents each. We find that 31 of the 34 pure Jeremiah chunks are assigned to Cluster 1 and all 31 of the pure Ezekiel chunks are assigned to Cluster 2. The 24 mixed clusters are split roughly evenly across the two clusters.

Selecting Cluster Cores

Now we select some of the documents in each cluster to be used as training examples for supervised learning, where each cluster represents a class. Since some of the chunks are actually mixed and not pure examples of either class, it is prudent at this stage to only use some of the chunks as training data for the next stage. In particular, we select cluster cores, as defined above, in the hopes that these are more likely to be pure chunks. Recall that core selection depends on the parameter p that determines what proportion of cluster members are actually selected for membership in the core of each cluster. (p is the percentage chosen for each of two separate criteria, so that the overall percentage chosen is typically less than p.)

For the Jeremiah-Ezekiel document, we set the value of p to 80. (Results for various values of p are shown in Table 1 below.) Of the 44 chunks in Cluster 1, 32 are included in the core and of the 44 chunks in Cluster 2, 34 are included in the core. Of the 32 chunks in the core of Cluster 1, 25 are pure Jeremiah (and the rest are mixed), while of the 34 chunks in the core of Cluster 29 are Ezekiel (and the rest are mixed). Note that the three pure Jeremiah chunks in Cluster 2 are not included in the core of that cluster.

Supervised Learning

Using the 66 chunks as labeled training examples (32 for Class 1 and 34 for Class 2), we represent each example in terms of frequencies of all words that appear in the corpus at least five times and use SVM (linear; default settings in Weka's SMO learner) to learn a classifier to distinguish Class 1 text from Class 2 text. Note that the feature set used in this stage is far larger than the feature set used in the initial clustering. In the initial clustering it was important to use a small set of features that would be more likely to capture distinct writing styles than to capture distinct topics. However, now that we have established what we hope is a reasonable (though imperfect) clustering, we are satisfied to capture all distinctions between the clusters. For example, if it happens that different authors do have somewhat different topic preferences, that information can be exploited at this stage.

Classifying individual sentences

We now apply this classifier to all 2637 sentences in the merged text. We find that 93% of Jeremiah sentences are assigned to Class 1 and 93% of Ezekiel sentences are assigned to Class 2. But we wish, at this stage, to take advantage of the fact that there is serial correlation between among the sentences. If a sentence that is not confidently assigned by the SVM classifier is found in a neighborhood of Class 1 sentences, it is very likely itself a Class 1 sentence. Thus, setting the parameter t (see above) to 25, we consider the 25% of sentences that are assigned to one class or the other with highest confidence (reflected in distance from the SVM boundary). Of these 657 sentences, 99% of Jeremiah sentences are assigned to Class 1 and 97% of Ezekiel sentences are assigned to Class 2.

For all other sentences (that is, for the 75% of sentences closest to the SVM boundary), we use a procedure known as "gap-filling" that works as follows. In Type 1 gap-filling, if the last confidently assigned sentence before some sentence S and the first confidently assigned sentence after S are assigned to the same class, we assign S to that class. In Type 2 gap-filling, when there is a string of unassigned sentences following a sentence assigned to Class 1 and succeeded by a sentence assigned to

Class 2, we identify the optimal split point and assign all sentences prior to the split point to Class 1 and all the rest to Class 2. (The optimal split point is the one that maximizes the difference between the respective signed distances from the SVM boundary of the text prior to the split point and the text subsequent to the test point.)

For our example, we have already seen that the 25% of most confidently assigned sentences achieve purity of 98%. Of the sentences now assigned using Type 1 gap-filling, 99% are correctly assigned. Of those sentences assigned using Type 2 gap-filling, only 90% are correctly assigned. (Thus, if we can tolerate some sentences remaining unassigned, we might consider not using Type 2 gap-filling and leave all sentences in such gaps unassigned.) Overall, assigning all sentences in this way, we obtain that 98% of sentences in Class 1 are Jeremiah and 96% of sentences in Class 2 are Ezekiel. The overall purity of this split is thus 97%.

Note that this result was obtained for particular values of the parameters c (chunk size) and p (% of chunks in cluster cores). In Table 1, we show the purity obtained for various choices of these parameters.

[Insert Table 1 here]

Having established that c=30 and p=80 yield optimal results for our development corpus, we use these same values for all the experiments in the rest of the paper.

6. Experimental Setup

We wish to now systematically test our method on a number of disparate test corpora.

Our first corpus consists of blog posts by the Nobel laureate economist Gary Becker and the judge and law scholar Richard Posner. Becker and Posner maintain a joint blog (www.becker-posner-blog.com) in which they each post weekly on a mutually agreed upon topic. We choose this corpus precisely because, unlike pairs of biblical books, the distinct authorial components are thematically indistinguishable. As will be described below in detail, we construct a merged document by choosing a random number of sentences from Becker, followed by a random number of sentences from Posner, and so on. (We pay no attention to boundaries between individual posts by any given author.) Segmenting the resulting merged text into Becker segments and Posner segments presents two challenges. First, the wide variety of topics offers many plausible ways to split the text along thematic lines. Second, the aggregate texts by Becker and by Posner, respectively, are *identically distributed over topics*, so that no thematic clues can help us to distinguish between the respective authors.

Our second corpus consists of four columnists writing for the New York Times (Gail Collins, Thomas Friedman, Maureen Dowd and Paul Krugman). This corpus offers the same challenges as in the case of the Becker-Posner blog, but in addition allows us to tackle the case where there are more than two authors.

Our third corpus consists of other pairs of biblical books (or partial books). For comparison purposes, we use precisely those books used by Koppel et al. (2011): Isaiah (Chapters 1-33), Jeremiah, Ezekiel, Job (Chapters 3-41) and Proverbs. We note that the first three are all in the prophetic literature genre and the latter two are both in the wisdom literature genre. Thus, in merging pairs of books, we will distinguish between same-genre pairs and different-genre pairs.

In each case, we create merged documents using the procedure described above in general terms for the example of Jeremiah-Ezekiel. We now describe the procedure more formally. Given k authors {1,...,k}, let T_j be some text written by author j. We generate a merged document using the following method:

For i=1,2,…

1 Choose text $T_{t(i)}$, where t(i) is a randomly chosen integer from 1 to k subject to the condition that $t(i) \neq t(i-1)$. This is the author from whom we will now draw the next sequence of sentences.

2 Append to the new text the first x unused sentences in $T_{t(i)}$, where x is randomly chosen from a uniform distribution over the integers from 1 to n. Note that n is the maximal length of a sequence of sentences drawn from a single author. It determines the granularity of the merged text.

The procedure continues until all k documents have been exhausted.

Note that, because we use a uniform distribution, the lengths of the single-author segments (that is, the number of sentences between transitions) might vary very widely. This makes the problem more difficult – and more realistic.

We will apply the method exactly as described above to each of our testbed corpora. Our measure of success is purity, as considered above but which, for convenience, we repeat here for the case of k authors: we create a k*k confusion matrix M in which m_{ij} is the number of sentences of author i assigned to class j. For any given ordering of the authors, we order the classes in such manner as to maximize the sum of the diagonal – that is, we associate classes with authors in the most "natural" way. (Typically, this simply means that we associate each class with the author most prominently represented in that class.) Purity (Manning et al., 2008) is the proportion of sentences on the diagonal.

7. Becker-Posner Blog: Controlling for Topic

For our first experiment, we consider the Becker-Posner blog. As noted above, the importance of this experiment lies in the fact that thematic clues cannot be exploited to distinguish the authors - Becker and Posner wrote about precisely the same variety of topics.

Becker's text consists of 14,689 sentences and Posner's text consists of 12,233 sentences. We create a merged text by alternately taking sentences from each author, the number of sentences drawn from a uniform distribution between 1 and 200. In the resulting merged text, there are 261 transitions from one author to the other, with a median of just above 100 sentences between transitions, as we'd expect. (Below, we will consider the case in which the mixture is more granular.)

We apply the algorithm exactly as described above. Importantly, we fix the parameter values as determined in the Jeremiah-Ezekiel experiment, so that there is no retrospective tuning.

We begin to decompose the text into two clusters of sentences by chunking the text into segments of 30 sentences each. Thus, in this case we obtain 898 chunks. Of these chunks, 390 are pure Becker, 296 are pure Posner and the rest are mixed. As above, we represent the chunks as binary vectors recording the presence or absence in the chunk of each of the 500 most common words in the document, use cosine to measure the similarity between each pair of chunks, and use n-cut clustering to cluster the chunks into two clusters. We find that of the 390 pure Becker clusters 383 are assigned to Cluster 1 and of the 296 pure Posner chunks 290 are assigned to Cluster 2. Of the mixed chunks, 88 are assigned to Cluster1 and 131 are assigned to Cluster2. The quality of the clustering is quite remarkable. It seems that there are certain words that are quite discriminative of the two authors. For example, Becker almost never uses the word *thus*.

We now compute the cores of each cluster. 303 chunks are included in the core of Cluster 1 and 279 chunks are included in the core of Cluster 2. Of the 303 chunks in the core of Cluster 1, 272 are pure Becker (and the rest are mixed), while of the 279 chunks in the core of Cluster 2, 227 are Posner (and the rest are mixed).

Representing each of the 582 chunks in terms of frequencies of all words that appear in the corpus at least five times, we treat the respective clusters as classes and use SVM to learn a classifier to distinguish Class 1 text from Class 2 text. Applying this classifier to all 26,922 sentences in the merged text, we find that 86% of Becker sentences are assigned to Class 1 and 87% of Posner sentences are assigned to Class 2. Considering only the 25% of sentences that are assigned to one class or the other with highest confidence (reflected in distance from the SVM boundary), we find that of these 6730 sentences, 97% of Becker sentences are assigned to Class 1 and 96% of Posner sentences are assigned to Class 2. Finally, assigning all other sentences as described above, we find that 95% of Becker sentences are assigned to Class 1 and 95% of Posner sentences are assigned to Class 2. The overall purity of the clustering is 94.9%.

The results reported thus far for the Becker-Posner corpus are based on a merged document with a median of 100 sentences between transitions from one author to the author. It might be wondered how sensitive our results are to the granularity of the merged document. The problem presumably gets harder if transitions from one author to the other are more frequent. To test this, we vary the maximum number of sentences between transitions from one author to another (the parameter n in Step 2 of our construction in section 6 above). In Figure 2, we show results on Becker-Posner for various values of n (median distance between transitions is approximately n/2). Indeed we find that for higher granularity (that is, for lower values of n), purity diminishes somewhat.

[Insert Figure 2 here]

8. NYT Columnists: Separating k>2 Authors

We now apply our method to the New York Times columnists corpus. The corpus includes the cumulative output of four columnists over the same date range (9/2007 – 12/2010) where the amount of words we have for each author ranges from 271,000 to 340,000. Our purpose here is to apply our method to cases in which we have 3 or 4 authors participating in a single merged document.

We note that for documents created by merging the writing of any of the six pairs of columnists, purity ranges from 88% to 97%. Results for documents created by merging more than two authors are shown in Figure 3. We find that for documents with three authors, we obtain purity ranging from 75% to 78% and for a document with four authors, we obtain purity of 71.1%.

[Insert Figure 3 here]

9. Biblical Book Pairs: Exploiting Synonym Usage

For our final experiment, we apply our method to various documents constructed by merging pairs of biblical books. This problem is of great historical interest and has drawn a good deal of attention from scholars interested in identifying distinct authorial layers in various books of the Bible. Most of the work in this area is impressionistic rather than computational, though a few (for example, Radday et al., 1970; Bee, 1971; Holmes, 1994) use multivariate analysis to test whether the clusters in a given clustering of some biblical text are sufficiently distinct to be regarded as probably a composite text. Note that this work differs from ours in that it wishes to determine if a work is composite, whereas our aim is to identify the optimal clustering of a document, given that it is composite. Furthermore, unlike our work, the earlier work has not attempted to empirically prove the efficacy of its methods by testing it against known ground truth.

Before we present our results, we introduce a special feature set that is of special interest for the Bible corpus. Thus far, the features we have used to represent texts – the presence or absence of each of the most common words in the text – are easily extracted from any text at all. This is, of course, a huge advantage of using such features. However, there are other feature sets that, while less easily extracted, hold the promise of greater effectiveness. For example, Koppel et al. (2011) use a feature set traditionally employed by Bible scholars to classify different components of biblical literature, namely, synonym choice. The underlying hypothesis is that different authorial components are likely to differ in the proportions with which alternative words from a set of synonyms (synset) are used. Consider some examples from our Jeremiah-Ezekiel corpus. There are two Hebrew synonyms - *peah* and *miqsoa* corresponding to the word "corner", two (*minḥah* and *terumah*) corresponding to the word "oblation", and two (*naṭa* and *shaṯal*) corresponding to the word "planted". For each of these three synsets, the first synonym is typically used by Jeremiah and the second is typically used by Ezekiel.

Historically, the hypothesis of distinctive synonym usage played a part in the pioneering work of Astruc (1753) on the book of Genesis – using a single synset: divine names – and has been refined for biblical works by many others using broader feature sets, such as that of Carpenter and Hartford-Battersby (1900). More recently, the synonym hypothesis has been used in computational work on authorship attribution of English texts in the work of Clark and Hannon (2007) and Koppel, Akiva & Dagan (2006).

Thus, in addition to the feature set we have used thus far -- the presence or absence of frequent words -- we wish to automatically segment and cluster biblical texts based on differences in synonym choice. We identify synonyms automatically by marking words translated identically in the King James translation into English as synonyms. Strong's (1890) Concordance is used for disambiguation and some remaining false positives, due to polysemy in English, were removed by hand. The above procedure yields a set of 517 synsets including a total of 1551 individual synonyms. Most synsets consist of only two synonyms, but some include many more. For example, there are 7 Hebrew synonyms corresponding to "fear".

The rest of the algorithm is identical to the one above. One detail, however, bears mention. In computing the cosine similarity of two chunks, we first eliminate from the formal representation of the chunks any synsets that do not appear in both chunks (where a synset is said to appear in a chunk if any of its constituent synonyms appear in the chunk). We then compute cosine of the truncated vectors.

For the parameter t (the % of sentences most confidently assigned by SVM and used as a basis for gap-filling), we follow Koppel et al. (2011) and select all sentences that lie outside of the SVM margins.

In Figure 4, we show purity results for same-genre pairs of Biblical books, using common words as described earlier and synonym choice, as just described. In Figure 5, we show the same results for different genre pairs. As can be seen, our method using presence/absence of common words works especially well – above 95% in all cases – for different-genre book pairs where vocabularies tend to be quite distinct. In many of these cases, the use of synonyms actually performs worse, probably due to the relative sparseness of the synonyms. On the other hand, common words and synonyms achieve comparable results for the more difficult same-genre pairs with synonyms decisively out-performing common words for the especially difficult cases of a merged Isaiah-Jeremiah book and a merged Isaiah-Ezekiel book.

[Insert Figure 4 here]

[Insert Figure 5 here]

Having applied our method successfully to artificial documents, we apply it to an actual document, namely, the first four books of the Hebrew Bible. This document has been the subject of considerable scholarly discussion. We use our method to decompose the document into two clusters. We take as our gold standard, the division of this text proposed by Driver (1909) and Friedman (2005) – based on a variety of literary considerations – into what are known as "Priestly" and "non-Priestly" components. (Sentences for which Driver and Friedman disagree are ignored for purposes of this evaluation.)

We find that using common words as our feature set in the first phase of the algorithm yields purity of 89.5% and using synonyms as our feature set yields purity of 91.5%.

10. Conclusions

We have found that it is possible to decompose unsegmented multi-author texts into authorial components with fairly high accuracy. The main idea is to cluster chunks of the text according to respective chunk lexicons (presence/absence of common words) to obtain an initial high-precision sampling of pure chunks for each author and then to exploit this sampling along with supervised learning methods to obtain a classifier that can be applied to individual sentences.

We have seen that the method is effective even for a text in which the authorial components are not thematically distinguished (the Becker-Posner blog). We have further seen that the method can be applied to texts composed of work by more than two authors (NYT columnists). Finally, we have seen that results obtained using our generic method are comparable, and typically superior, to those obtained using more sophisticated features, such as synonym usage (Bible book pairs). Our success on this last test-bed suggests that these methods could be used to replicate and improve work by biblical scholars in decomposing distinct authorial threads in biblical works.

We note that we have assumed here that the number of authorial components is known and the task is merely to optimally decompose the text into a given number of components. The important problem of determining the correct number of components is left for future work.

References

Astruc., J. (1753). Conjectures sur les Memoires originaux donI il parou que Moyse s'est servi pour composer Ie Livre de la Genese. Brussels.

Bee., R. E. (1971). Statistical methods in the study of the Masoretic text of the Old Testament. Journal of the Royal Statistical Society, 134(1):611-622.

Carpenter, J. E., Hartford-Battersby, G. (1900). The Hexateuch: According to the Revised Version. London.

Clark, J., & Hannon, C. (2007). A classifier system for author recognition using synonym-based features. Proceedings of the Sixth Mexican International Conference on Artificial Intelligence. Lecture Notes in Artificial Intelligence, vol. 4827,(pp. 839-849).

Dhillon, I.S., Guan, Y., & Kulis, B. (2007). Weighted graph cuts without eigenvectors: a multilevel approach. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI). vol. 29(11):1944-1957.

Driver, S. R. (1909). *An Introduction to the Literature of the Old Testament* (8th ed.). T&T Clark, Edinburgh.

Friedman, R. E. (2005). *The Bible with Sources Revealed.* HarperCollins.

Graham, N., Hirst, G., & Marthi, B. (2005). Segmenting documents by stylistic character, Natural Language Engineering. 11(4):397-415.

Holmes., D. (1994). Authorship attribution, Computers and the Humanities. 28(2):87-106.

Koppel, M., Akiva, N., & Dagan, I. (2006). Feature instability as a criterion for selecting potential style markers. Journal of the American Society for Information Science and Technology . (JASIST). 57(11):1519-1525.

Koppel, M., Akiva, N., Dershowitz, I., & Dershowitz, N. (2011). Unsupervised decomposition of a document into authorial components. Proceedings of ACL. (pp. 1356-1364). Portland OR.

Layton, R., Watters, P., & Dazeley, R. (2011). Automated unsupervised authorship analysis using evidence accumulation clustering. Natural Language Engineering.

Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to information retrieval. Cambridge University Press.

Meyer zu Eissen, S., & Stein, B. (2006). Intrinsic plagiarism detection. Proceedings of the 28th ECIR. (pp. 565-569). London.

Radday., Y.(1970). Isaiah and the computer: A preliminary report. Computers and the Humanities. 5(2):65-73.

Rosen-Zvi, M., Chemudugunta, C., Griffiths, T., Smyth P., & Steyvers , M. (2010). Learning author-topic models from text corpora. ACM Transactions on Information Systems (TOIS), ACM.

Stamatatos, E. (2009). Intrinsic plagiarism detection using character n-gram profiles.. Proceedings of the 3rd Int. Workshop on Un-covering Plagiarism, Authorship, and Social Software Misuse. SEPLN'09. (pp. 38–46).

Strong., J. (1890). The Exhaustive Concordance of the Bible. Nashville, TN. Retrieved November 14, 2010, from http://www.htmlbible.com/sacrednamebiblecom/kjvstrongs/STRINDEX.htm