Unsupervised Decomposition of a Document into Authorial Components

Moshe Koppel **Navot Akiva**
Dept. of Computer Science
Bar-Ilan University
Ramat Gan, Israel
{moishk,navot.akiva}@gmail.com

Idan Dershowitz
Dept. of Bible
Hebrew University
Jerusalem, Israel
dershowitz@gmail.com

Nachum Dershowitz
School of Computer Science
Tel Aviv University
Ramat Aviv, Israel
nachumd@tau.ac.il

Abstract

We propose a novel unsupervised method for separating out distinct authorial components of a document. In particular, we show that, given a book artificially "munged" from two thematically similar biblical books, we can separate out the two constituent books almost perfectly. This allows us to automatically recapitulate many conclusions reached by Bible scholars over centuries of research. One of the key elements of our method is exploitation of differences in synonym choice by different authors.

1 Introduction

We propose a novel unsupervised method for separating out distinct authorial components of a document.

There are many instances in which one is faced with a multi-author document and wishes to delineate the contributions of each author. Perhaps the most salient example is that of documents of historical significance that appear to be composites of multiple earlier texts. The challenge for literary scholars is to tease apart the document's various components. More 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 treat two versions of the problem. In the first, easier, 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. In the second version, we are given an unsegmented document and the challenge includes segmenting the document as well as clustering the resulting units.

We assume here that no information about the authors of the document is available and that in particular we are not supplied with any identified samples of any author's writing. Thus, our methods must be entirely unsupervised.

There is surprisingly little literature on this problem, despite its importance. Some work in this direction has been done on intrinsic plagiarism detection (e.g., Meyer zu Eisen 2006) and document outlier detection (e.g., Guthrie et al. 2008), but this work makes the simplifying assumption that there is a single dominant author, so that outlier units can be identified as those that deviate from the document as a whole. We don't make this simplifying assumption. Some work on a problem that is more similar to ours was done by Graham et al. (2005). However, they assume that examples of pairs of paragraphs labeled as same-author/different-author are available for use as the basis of supervised learning. We make no such assumption.

The obvious approach to our unsupervised version of the problem would be to segment the text (if necessary), represent each of the resulting units of text as a bag-of-words, and then use clustering algorithms to find natural clusters. We will see, however, that this naïve method is quite inadequate. Instead, we exploit a method favored by the literary scholar, namely, the use of synonym choice. Synonym choice proves to be far more useful for authorial decomposition than ordinary lexical features. However, synonyms are relatively sparse and hence, though reliable, they are not

comprehensive; that is, they are useful for separating out some units but not all. Thus, we use a two-stage process: first find a reliable partial clustering based on synonym usage and then use these as the basis for supervised learning using a different feature set, such as bag-of-words.

We use biblical books as our testbed. We do this for two reasons. First, this testbed is well motivated, since scholars have been doing authorial analysis of biblical literature for centuries. Second, precisely because it is of great interest, the Bible has been manually tagged in a variety of ways that are extremely useful for our method.

Our main result is that given artificial books constructed by randomly "munging" together actual biblical books, we are able to separate out authorial components with extremely high accuracy, even when the components are thematically similar. Moreover, our automated methods recapitulate many of the results of extensive manual research in authorial analysis of biblical literature.

The structure of the paper is as follows. In the next section, we briefly review essential information regarding our biblical testbed. In Section 3, we introduce a naïve method for separating components and demonstrate its inadequacy. In Section 4, we introduce the synonym method, in Section 5 we extend it to the two-stage method, and in Section 6, we offer systematic empirical results to validate the method. In Section 7, we extend our method to handle documents that have not been pre-segmented and present more empirical results. In Section 8, we suggest conclusions, including some implications for Bible scholarship.

2 The Bible as Testbed

While the biblical canon differs across religions and denominations, the common denominator consists of twenty-odd books and several shorter works, ranging in length from tens to thousands of verses. These works vary significantly in genre, and include historical narrative, law, prophecy, and wisdom literature. Some of these books are regarded by scholars as largely the product of a single author's work, while others are thought to be composites in which multiple authors are well-represented – authors who in some cases lived in widely disparate periods. In this paper, we will focus exclusively on the Hebrew books of the Bible, and we will work with the original untranslated texts.

The first five books of the Bible, collectively known as the Pentateuch, are the subject of much controversy. According to the predominant Jewish and Christian traditions, the five books were written by a single author – Moses. Nevertheless, scholars have found in the Pentateuch what they believe are distinct narrative and stylistic threads corresponding to multiple authors.

Until now, the work of analyzing composite texts has been done in mostly impressionistic fashion, whereby each scholar attempts to detect the telltale signs of multiple authorship and compilation. Some work on biblical authorship problems within a computational framework has been attempted, but does not handle our problem. Much earlier work (for example, Radday 1970; Bee 1971; Holmes 1994) uses 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. By contrast, our aim is to find the optimal clustering of a document, given that it is composite. Crucially, unlike that earlier work, we empirically prove the efficacy of our methods by testing it against known ground truth. Other computational work on biblical authorship problems (Mealand 1995; Berryman et al. 2003) involves supervised learning problems where some disputed text is to be attributed to one of a set of known authors. The supervised authorship attribution problem has been well-researched (for surveys, see Juola (2008), Koppel et al. (2009) and Stamatatos (2009)), but it is quite distinct from the unsupervised problem we consider here.

Since our problem has been dealt with almost exclusively using heuristic methods, the subjective nature of such research has left much room for debate. We propose to set this work on a firm algorithmic basis by identifying an optimal stylistic subdivision of the text. We do not concern ourselves with how or why such distinct threads exist. Those for whom it is a matter of faith that the Pentateuch is not a composition of multiple writers can view the distinction investigated here as that of multiple styles.

3 A Naïve Algorithm

For expository purposes, we will use a canonical example to motivate and illustrate each of a

sequence of increasingly sophisticated algorithms for solving the decomposition problem. 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 first challenge, 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.) Note that this is a pre-segmented version of the problem since we know that each chapter belongs to only one of the books.

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 ncut clustering (Dhillon et al. 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, which we call normalized majority diagonal (NMD), runs from 50% (when the clusters are completely orthogonal to the desired split) to 100% (where the clusters are identical with the desired split). NMD 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 an NMD 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 define a class of generic biblical words consisting of all 223 words that appear at least five times in each of ten different books of the Bible.

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

Book	Cluster I	Cluster II
Jer	32	20
Eze	28	20

As can be seen, using generic words yields NMD of 51.3%, which does not improve matters at all. Thus, we need to try a different approach.

4 Exploiting Synonym Usage

One of the key features used by Bible scholars to classify different components of biblical literature is 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. This hypothesis 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 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 et al. (2006).

This approach presents several technical challenges. First, ideally – in the absence of a sufficiently comprehensive thesaurus – we would wish to identify synonyms in an automated fashion. Second, we need to adapt our similarity measure for reasons that will be made clear below.

4.1 (Almost) Automatic Synset Identification

One of the advantages of using biblical literature is the availability of a great deal of manual annotation. In particular, we are able to identify synsets by exploiting the availability of the standard King James translation of the Bible into Eng-

lish (KJV). Conveniently, and unlike most modern translations, KJV almost invariably translates synonyms identically. Thus, we can generally identify synonyms by considering the translated version of the text. There are two points we need to be precise about. First, it is not actually words that we regard as synonymous, but rather word roots. Second, to be even more precise, it is not quite roots that are synonymous, but rather senses of roots. Conveniently, Strong's (1890 [2010]) Concordance lists every occurrence of each sense of each root that appears in the Bible separately (where senses are distinguished in accordance with the KJV translation). Thus, we can exploit KJV and the concordance to automatically identify synsets as well as occurrences of the respective synonyms in a synset.[1] (The above notwithstanding, there is still a need for a bit of manual intervention: due to polysemy in English, false synsets are occasionally created when two non-synonymous Hebrew words are translated into two senses of the same English word. Although this could probably be handled automatically, we found it more convenient to do a manual pass over the raw synsets and eliminate the problems.)

The above procedure yields a set of 529 synsets including a total of 1595 individual synonyms. Most synsets consist of only two synonyms, but some include many more. For example, there are 7 Hebrew synonyms corresponding to "fear".

4.2 Adapting the Similarity Measure

Let's now represent a unit of text as a vector in the following way. Each entry represents a synonym in one of the synsets. If none of the synonyms in a synset appear in the unit, all their corresponding entries are 0. If j different synonyms in a synset appear in the unit, then each corresponding entry is 1/j and the rest are 0. Thus, in the typical case where exactly one of the synonyms in a synset appears, its corresponding entry in the vector is 1 and the rest are 0.

Now we wish to measure the similarity of two such vectors. The usual cosine measure doesn't capture what we want for the following reason. If the two units use different members of a synset, cosine is diminished; if they use the same members of a synset, cosine is increased. So far, so good. But suppose one unit uses a particular synonym and the other doesn't use any member of that synset. This should teach us nothing about the similarity of the two units, since it reflects only on the relevance of the synset to the content of that unit; it says nothing about which synonym is chosen when the synset is relevant. Nevertheless, in this case, cosine would be diminished.

The required adaptation is as follows: we first eliminate from the representation any synsets that do not appear in both units (where a synset is said to appear in a unit if any of its constituent synonyms appear in the unit). We then compute cosine of the truncated vectors. Formally, for a unit x represented in terms of synonyms, our new similarity measure is $\cos'(x,y) = \cos(x|_{S(x\cap y)}, y|_{S(x\cap y)})$, where $x|_{S(x\cap y)}$ is the projection of x onto the synsets that appear in both x and y.

4.3 Clustering Jeremiah-Ezekiel Using Synonyms

We now apply ncut clustering to the similarity matrix computed as described above. We obtain the following split:

Book	Cluster I	Cluster II
Jer	48	4
Eze	5	43

Clearly, this is quite a bit better than results obtained using simple lexical features as described above. Intuition for why this works can be purchased by considering concrete examples. There are two Hebrew synonyms – *pēʾâh* and *miqṣôaʿ* corresponding to the word "corner", two (*minḥâh* and *tĕrûmâh*) corresponding to the word "oblation", and two (*nāṭaʿ* and *šāṯal*) corresponding to the word "planted". We find that *pēʾâh*, *minḥâh* and *nāṭaʿ* tend to be located in the same units and, concomitantly, *miqṣôaʿ*, *tĕrûmâh* and *šāṯal* are located in the same units. Conveniently, the former are all Jeremiah and the latter are all Ezekiel.

While the above result is far better than those obtained using more naïve feature sets, it is, nevertheless, far from perfect. We have, however, one more trick at our disposal that will improve these results further.

5 Combining Partial Clustering and Supervised Learning

Analysis of the above clustering results leads to two observations. First, some of the units belong

[1] Thanks to Avi Shmidman for his assistance with this.

firmly to one cluster or the other. The rest have to be assigned to one cluster or the other because that's the nature of the clustering algorithm, but in fact are not part of what we might think of as the *core* of either cluster. Informally, we say that a unit is in the core of its cluster if it is sufficiently similar to the centroid of its cluster and it is sufficiently more similar to the centroid of its cluster than to any other centroid. Formally, let S be a set of synsets, let B be a set of units, and let C be a clustering of B where the units in B are represented in terms of the synsets in S. For a unit x in cluster C(x) with centroid c(x), we say that x is in the core of C(x) if $\cos'(x,c(x))>\theta_1$ and $\cos'(x,c(x))-\cos'(x,c)>\theta_2$ for every centroid $c \neq c(x)$. In our experiments below, we use $\theta_1=1/\sqrt{2}$ (corresponding to an angle of less than 45 degrees between x and the centroid of its cluster) and $\theta_2=0.1$.

Second, the clusters that we obtain are based on a subset of the full collection of synsets that does the heavy lifting. Formally, we say that a synonym n in synset s is *over-represented* in cluster C if $p(x \in C|n \in x) > p(x \in C|s \in x)$ and $p(x \in C|n \in x) > p(x \in C)$. That is, n is over-represented in C if knowing that n appears in a unit increases the likelihood that the unit is in C, relative to knowing only that some member of n's synset appears in the unit and relative to knowing nothing. We say that a synset s is a *separating* synset for a clustering {C1,C2} if some synonym in s is over-represented in C1 and a different synonym in s is over-represented in C2.

5.1 Defining the Core of a Cluster

We leverage these two observations to formally define the cores of the respective clusters using the following iterative algorithm.

1. Initially, let S be the collection of all synsets, let B be the set of all units in the corpus represented in terms of S, and let {C1,C2} be an initial clustering of the units in B.
2. Reduce B to the cores of C1 and C2.
3. Reduce S to the separating synsets for {C1,C2}.
4. Redefine C1 and C2 to be the clusters obtained from clustering the units in the reduced B represented in terms of the synsets in reduced S.
5. Repeat Steps 2-4 until convergence (no further changes to the retained units and synsets).

At the end of this process, we are left with two well-separated cluster cores and a set of separating synsets. When we compute cores of clusters in our Jeremiah-Ezekiel experiment, 26 of the initial 100 units are eliminated. Of the 154 synsets that appear in the Jeremiah-Ezekiel corpus, 118 are separating synsets for the resulting clustering. The resulting cluster cores split with Jeremiah and Ezekiel as follows:

Book	Cluster I	Cluster II
Jer	36	0
Eze	2	36

We find that all but two of the misplaced units are not part of the core. Thus, we have a better clustering but it is only a partial one.

5.2 Using Cores for Supervised Learning

Now that we have what we believe are strong representatives of each cluster, we can use them in a supervised way to classify the remaining unclustered units. The interesting question is which feature set we should use. Using synonyms would just get us back to where we began. Instead we use the set of generic Bible words introduced earlier. The point to recall is that while this feature set proved inadequate in an unsupervised setting, this does not mean that it is inadequate for separating Jeremiah and Ezekiel, given a few good training examples.

Thus, we use a bag-of-words representation restricted to generic Bible words for the 74 units in our cluster cores and label them according to the cluster to which they were assigned. We now apply SVM to learn a classifier for the two clusters. We assign each unit, including those in the training set, to the class assigned to it by the SVM classifier. The resulting split is as follows:

Book	Cluster I	Cluster II
Jer	51	1
Eze	0	48

Remarkably, even the two Ezekiel chapters that were in the Jeremiah cluster (and hence were essentially misleading training examples) end up on the Ezekiel side of the SVM boundary.

It should be noted that our two-stage approach to clustering is a generic method not specific to our particular application. The point is that there are some feature sets that are very well suited to a particular unsupervised problem but are sparse, so they give only a partial clustering. At the same time, there are other feature sets that are denser and, possibly for that reason, adequate for super-

vised separation of the intended classes but inadequate for unsupervised separation of the intended classes. This suggests an obvious two-stage method for clustering, which we use here to good advantage.

This method is somewhat reminiscent of semi-supervised methods sometimes used in text categorization where few training examples are available (Nigam et al. 2000). However, those methods typically begin with some information, either in the form of a small number of labeled documents or in the form of keywords, while we are not supplied with these. Furthermore, the semi-supervised work bootstraps iteratively, at each stage using features drawn from within the same feature set, while we use exactly two stages, the second of which uses a different type of feature set than the first.

For the reader's convenience, we summarize the entire two-stage method:

1. Represent units in terms of synonyms.
2. Compute similarities of pairs of units using cos'.
3. Use ncut to obtain an initial clustering.
4. Use the iterative method to find cluster cores.
5. Represent units in cluster cores in terms of generic words.
6. Use units in cluster cores as training for learning an SVM classifier.
7. Classify all units according to the learned SVM classifier.

6 Empirical Results

We now test our method on other pairs of biblical books to see if we obtain comparable results to those seen above. We need, therefore, to identify a set of biblical books such that (i) each book is sufficiently long (say, at least 20 chapters), (ii) each is written by one primary author, and (iii) the authors are distinct. Since we wish to use these books as a gold standard, it is important that there be a broad consensus regarding the latter two, potentially controversial, criteria. Our choice is thus limited to the following five books that belong to two biblical sub-genres: Isaiah, Jeremiah, Ezekiel (prophetic literature), Job and Proverbs (wisdom literature). (Due to controversies regarding authorship (Pope 1952, 1965), we include only Chapters 1-33 of Isaiah and only Chapters 3-41 of Job.)

Recall that our experiment is as follows: For each pair of books, we are given all the chapters in the union of the two books and are given no information regarding labels. The object is to sort out the chapters belonging to the respective two books. (The fact that there are precisely two constituent books is given.)

We will use the three algorithms seen above:

1. generic biblical words representation and ncut clustering;
2. synonym representation and ncut clustering;
3. our two-stage algorithm.

We display the results in two separate figures. In Figure 1, we see results for the six pairs of books that belong to different sub-genres. In Figure 2, we see results for the four pairs of books that are in the same genre. (For completeness, we include Jeremiah-Ezekiel, although it served above as a development corpus.) All results are normalized majority diagonal.

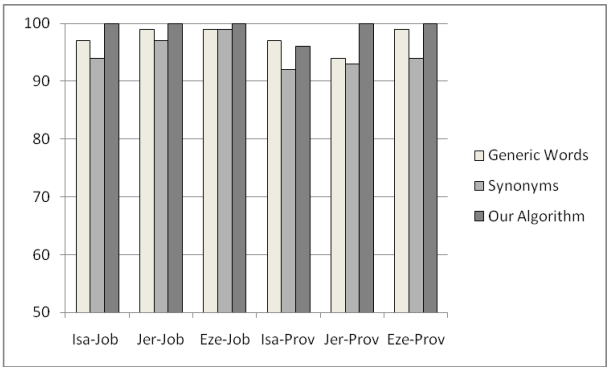

Figure 1. Results of three clustering methods for different-genre pairs

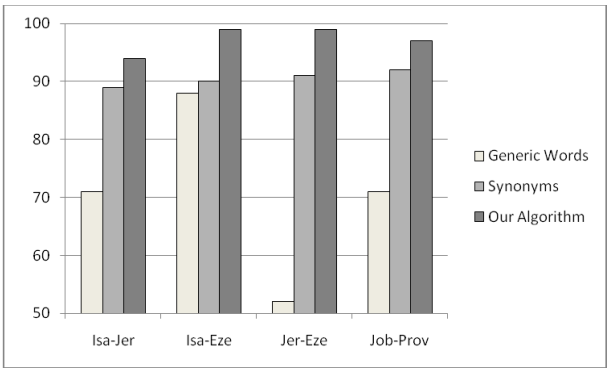

Figure 2. Results of three clustering methods for same-genre pairs

As is evident, for different-genre pairs, even the simplest method works quite well, though not as well as the two-stage method, which is perfect for five of six such pairs. The real advantage of the two-stage method is for same-genre pairs. For

these the simple method is quite erratic, while the two-stage method is near perfect. We note that the synonym method without the second stage is slightly worse than generic words for different-genre pairs (probably because these pairs share relatively few synsets) but is much more consistent for same-genre pairs, giving results in the area of 90% for each such pair. The second stage reduces the errors considerably over the synonym method for both same-genre and different-genre pairs.

7 Decomposing Unsegmented Documents

Up to now, we have considered the case where we are given text that has been pre-segmented into pure authorial units. This does not capture the kind of decomposition problems we face in real life. For example, in the Pentateuch problem, the text is divided up according to chapter, but there is no indication that the chapter breaks are correlated with crossovers between authorial units. Thus, we wish now to generalize our two-stage method to handle unsegmented text.

7.1 Generating Composite Documents

To make the problem precise, let's consider how we might create the kind of document that we wish to decompose. For concreteness, let's think about Jeremiah and Ezekiel. We create a composite document, called *Jer-iel*, as follows:

1. Choose the first k_1 available verses of Jeremiah, where k_1 is a random integer drawn from the uniform distribution over the integers 1 to m.
2. Choose the first k_2 available verses of Ezekiel, where k_2 is a new random integer drawn from the above distribution.
3. Repeat until one of the books is exhausted; then choose the remaining verses of the other book.

For the experiments discussed below, we use m=100 (though further experiments, omitted for lack of space, show that results shown are essentially unchanged for any $m \geq 60$). Furthermore, to simulate the Pentateuch problem, we break Jer-iel into initial units by beginning a new unit whenever we reach the first verse of one of the original chapters of Jeremiah or Ezekiel. (This does not leak any information since there is no inherent connection between these verses and actual crossover points.)

7.2 Applying the Two-Stage Method

Our method works as follows. First, we refine the initial units (each of which might be a mix of verses from Jeremiah and Ezekiel) by splitting them into smaller units that we hope will be pure (wholly from Jeremiah or from Ezekiel). We say that a synset is *doubly-represented* in a unit if the unit includes two different synonyms of that synset. Doubly-represented synsets are an indication that the unit might include verses from two different books. Our object is thus to split the unit in a way that minimizes doubly-represented synonyms. Formally, let M(x) represent the number of synsets for which more than one synonym appear in x. Call $\langle x_1, x_2 \rangle$ a *split* of x if $x = x_1 x_2$. A split $\langle x_1', x_2' \rangle$ is *optimal* if $\langle x_1', x_2' \rangle = \text{argmax } M(x) - \max(M(x_1), M(x_2))$ where the maximum is taken over all splits of x. If for an initial unit, there is some split for which $M(x) - \max(M(x_1), M(x_2))$ is greater than 0, we split the unit optimally; if there is more than one optimal split, we choose the one closest to the middle verse of the unit. (In principle, we could apply this procedure iteratively; in the experiments reported here, we split only the initial units but not split units.)

Next, we run the first six steps of the two-stage method on the units of Jer-iel obtained from the splitting process, as described above, until the point where the SVM classifier has been learned. Now, instead of classifying chapters as in Step 7 of the algorithm, we classify individual verses.

The problem with classifying individual verses is that verses are short and may contain few or no relevant features. In order to remedy this, and also to take advantage of the stickiness of classes across consecutive verses (if a given verse is from a certain book, there is a good chance that the next verse is from the same book), we use two smoothing tactics.

Initially, each verse is assigned a raw score by the SVM classifier, representing its signed distance from the SVM boundary. We smooth these scores by computing for each verse a refined score that is a weighted average of the verse's raw score and the raw scores of the two verses preceding and succeeding it. (In our scheme, the verse itself is given 1.5 times as much weight as its immediate neighbors and three times as much weight as secondary neighbors.)

Moreover, if the refined score is less than 1.0 (the width of the SVM margin), we do not initially

assign the verse to either class. Rather, we check the class of the last assigned verse before it and the first assigned verse after it. If these are the same, the verse is assigned to that class (an operation we call "filling the gaps"). If they are not, the verse remains unassigned.

To illustrate on the case of Jer-iel, our original "munged" book has 96 units. After pre-splitting, we have 143 units. Of these, 105 are pure units. Our two cluster cores, include 33 and 39 units, respectively; 27 of the former are pure Jeremiah and 30 of the latter are pure Ezekiel; no pure units are in the "wrong" cluster core. Applying the SVM classifier learned on the cluster cores to individual verses, 992 of the 2637 verses in Jer-iel lie outside the SVM margin and are assigned to some class. All but four of these are assigned correctly. Filling the gaps assigns a class to 1186 more verses, all but ten of them correctly. Of the remaining 459 unassigned verses, most lie along transition points (where smoothing tends to flatten scores and where preceding and succeeding assigned verses tend to belong to opposite classes).

7.3 Empirical Results

We randomly generated composite books for each of the book pairs considered above. In Figures 3 and 4, we show for each book pair the percentage of all verses in the munged document that are "correctly" classed (that is, in the majority diagonal), the percentage incorrectly classed (minority diagonal) and the percentage not assigned to either class. As is evident, in each case the vast majority of verses are correctly assigned and only a small fraction are incorrectly assigned. That is, we can tease apart the components almost perfectly.

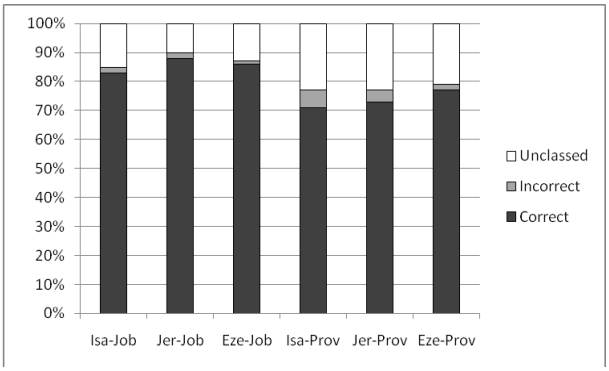

Figure 3. Percentage of verses in each munged different-genre pair of books that are correctly and incorrectly assigned or remain unassigned.

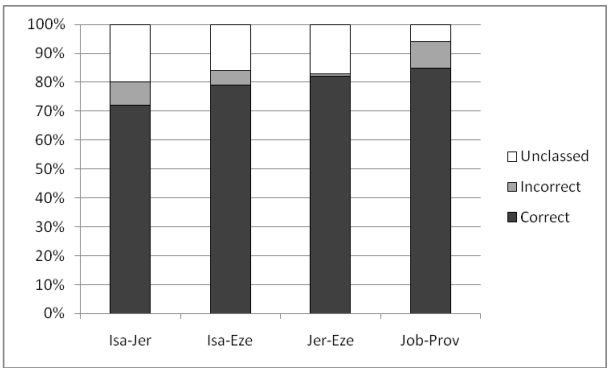

Figure 4. Percentage of verses in each munged same-genre pair of books that are correctly and incorrectly assigned or remain unassigned.

8 Conclusions and Future Work

We have shown that documents can be decomposed into authorial components with very high accuracy by using a two-stage process. First, we establish a reliable partial clustering of units by using synonym choice and then we use these partial clusters as training texts for supervised learning using generic words as features.

We have considered only decompositions into two components, although our method generalizes trivially to more than two components, for example by applying it iteratively. The real challenge is to determine the correct number of components, where this information is not given. We leave this for future work.

Despite this limitation, our success on munged biblical books suggests that our method can be fruitfully applied to the Pentateuch, since the broad consensus in the field is that the Pentateuch can be divided into two main authorial categories: Priestly (P) and non-Priestly (Driver 1909). (Both categories are often divided further, but these subdivisions are more controversial.) We find that our split corresponds to the expert consensus regarding P and non-P for over 90% of the verses in the Pentateuch for which such consensus exists. We have thus been able to largely recapitulate several centuries of painstaking manual labor with our automated method. We offer those instances in which we disagree with the consensus for the consideration of scholars in the field.

In this work, we have exploited the availability of tools for identifying synonyms in biblical literature. In future work, we intend to extend our methods to texts for which such tools are unavailable.

References

J. Astruc. 1753. *Conjectures sur les mémoires originaux dont il paroit que Moyse s'est servi pour composer le livre de la Genèse.* Brussels.

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

M. J. Berryman, A. Allison, and D. Abbott. 2003. Statistical techniques for text classification based on word recurrence intervals. *Fluctuation and Noise Letters,* 3(1):L1-L10.

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

J. Clark and C. Hannon. 2007. A classifier system for author recognition using synonym-based features. *Proc. Sixth Mexican International Conference on Artificial Intelligence, Lecture Notes in Artificial Intelligence*, vol. 4827, pp. 839-849.

I. S. Dhillon, Y. Guan, and B. Kulis. 2004. Kernel k-means: spectral clustering and normalized cuts. *Proc. ACM International Conference on Knowledge Discovery and Data Mining (KDD),* pp. 551-556.

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

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

D. Guthrie, L. Guthrie, and Y. Wilks. 2008. An unsupervised probabilistic approach for the detection of outliers in corpora. *Proc. Sixth International Language Resources and Evaluation (LREC'08),* pp. 28-30.

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

P. Juola. 2008. *Author Attribution*. Series title: *Foundations and Trends in Information Retrieval*. Now Publishing, Delft.

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

M. Koppel, J. Schler, and S. Argamon. 2009. Computational methods in authorship attribution. *J. of the American Society for Information Science and Technology*, 60(1):9-26.

D. L. Mealand. 1995. Correspondence analysis of Luke. *Lit. Linguist Computing*, 10(3):171-182.

S. Meyer zu Eisen and B. Stein. 2006. Intrinsic plagiarism detection. *Proc. European Conference on Information Retrieval (ECIR 2006), Lecture Notes in Computer Science,* vol. 3936, pp. 565–569.

K. Nigam, A. K. McCallum, S. Thrun, and T. M. Mitchell. 2000. Text classification from labeled and unlabeled documents using EM, *Machine Learning*, 39(2/3):103-134.

M. H. Pope. 1965. *Job (The Anchor Bible*, Vol. XV). Doubleday, New York, NY.

M. H. Pope. 1952. Isaiah 34 in relation to Isaiah 35, 40-66. *Journal of Biblical Literature,* 71(4):235-243.

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

E. Stamatatos. 2009. A survey of modern authorship attribution methods. *J. of the American Society for Information Science and Technology*, 60(3):538-556.

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