Razvan Surdulescu
CS391L: Machine Learning
Final Project Report: Verifying Authorship
12/21/2004

Introduction

There are many literary texts whose author's identity has been lost in time. Literary detectives use historical and stylistic clues to solve such scholarly puzzles, but they rarely withstand close scientific scrutiny due to the inherently subjective nature of these methodologies. The problem of objectively determining authorship is of great importance, both in the literary and historical fields as well as, more broadly, in law enforcement (e.g. plagiarism), music, art, and so on.

[KOPP04] introduces a new and promising algorithm, called *unmasking*, which takes a sample X and tells us whether it belongs to a *single* author A. They call this the authorship *verification* problem. The key difficulty in this problem is that it is impossible to generate a representative training set for A vs. non-A, so we cannot use standard classification techniques. The key idea in solving this problem is to iteratively evaluate the depth of difference between X and A as we remove discriminating features. There is a closely related, but significantly easier authorship *attribution* problem which takes a sample X and assigns it to one of *multiple* given authors using standard classification approaches.

In this project, we implement unmasking and meticulously analyze the key parameters that affect its performance on the original English corpus, as the description in [KOPP04] is terse for some important implementation details. We then evaluate the performance of unmasking on a corpus of Romanian literary works, and find that the algorithm is robust and performs surprisingly well, despite a very naïve stop-word and grammatical framework for the Romanian language. We conclude with a set of open questions and future directions.

Problem Definition and Algorithm

Task Definition

We start with an anonymous text X and a set of texts belonging to an author A. We want a binary yes/no answer (or a confidence interval) that indicates whether X has been written by A. We want the method to depend as little as possible on any particular language, genre, or other literary idiosyncrasies, in order to make the result more credible (less dependent on subjective features), and also to enhance its applicability to other domains.

In this paper, we use the same notation as in [KOPP04]: A denotes the texts written by some author, and X denotes the unknown text whose authorship we wish to ascertain. We further use the notation A_X to denote all the works in the corpus of A (if A did *not* write X), or all the works in the corpus of A *except* X (if A *did* write X). In the real world, this distinction is immaterial because X is by definition an anonymous text (it is not known to belong to A) so A_X will always be congruent to A. The distinction between A_X and A is important in this paper because, during our experiments, we *do* know beforehand to which A does X belong and we do not wish to include X into the training data for determining its own authorship.

The key difficulty in authorship verification is that it is nearly impossible to generate the training data. In authorship attribution, we can successfully learn a model of A vs. B. vs. C etc., where A, B, and C are authors with well known corpuses. However, in authorship verification, we want to learn a model of A vs. non-A, and there is no good, exhaustive, and representative definition of non-A. This suggests that we should rely only on positive data (text written by A) so we must employ only features from this domain in our algorithm.

To further complicate matters, given A and X, it is possible to train a baseline SVM ([KOPP04 sec. 5.2]) to learn an accurate model of A vs. X (we do not reproduce this particular result here because it is peripheral to this report). Even with no tweaking, such an SVM will have *very high* classification accuracy. This means that *regardless* of whether A wrote X or not, we can learn to distinguish A from X with very high accuracy, so *any* author will appear as very different from X. Conversely, no author will appear as any more likely than others to have written X. This suggests that an exact classifier that learns A vs. X will not be appropriate so we have to use a more subtle approach.

As we will see later on, it is in fact possible to build a classifier that uses only positive data and employs a subtle classification technique to ascribe authorship to X. Almost by definition, such a classifier can be easily applied to other similar problems in completely different areas, which makes it even more appealing. To wit, we apply this classifier to a body of Romanian text and show that it performs very well despite the fact that we used a limited set of stop words and a very impoverished word stemmer for the Romanian language; this validates the assertion that the algorithm is robust and does not depend on diminutive features of our datasets.

Algorithm Idea

Given an SVM model for A_X vs. X, we rank all the input features used in creating the model. If we iteratively remove the most important features from being used in the SVM model, we observe a very peculiar behavior: if A *did* write X, the classification accuracy decreases dramatically as we remove features. In other words, A and X become more alike. If, however, A did *not* write X, then the classification accuracy stays roughly high as we remove features. In other words, A and X remain relatively different. This is the core of the unmasking algorithm.

The idea behind unmasking makes sense intuitively: if A wrote X, there are few important input features that set them apart (i.e. there are fundamental similarities that underlie authorship even if the author's style may have changed over time). These few important features do most of the work in training an accurate SVM for A vs. X; once they are removed, A and X start to look increasingly the same. Conversely, if there are no fundamental similarities, A and X continue to look different even as features are removed, which suggests that A probably did not write X.

The speed with which A and X start to look the same is the key indicator for authorship. We will demonstrate this phenomenon visually, on the graphs below, as well as empirically, by means of a meta-learner that runs on the classification accuracy degradation curves.

Note that all assumptions about how to represent the feature vectors used in training the SVM are largely orthogonal to the essence of the algorithm (comparing the speed of degradation). Although the *actual* value of the degradation speed is naturally dependent on the choice and representation of features (we will see concretely in subsequent sections), the algorithm will run the same no matter what the feature representation. This is a powerful characteristic, for it allows unmasking to be run against many other problem domains.

Algorithm Definition

Here is the algorithm definition (based on the description in [KOPP04 sec. 5.1, 5.3]). The algorithm is provided in pseudo-code format, the reader is strongly encouraged to peruse the attached source (`Unmask.java`) for the complete details:

```
1. Generate the vocabulary
    1.1. Compute the frequency for all the words in $A_X$
    1.2. Compute the frequency for all the words in X
    1.3. Create a list of all the common words between $A_X$ and X
    1.4. Sort it by the average of their frequencies
    1.5. Select the top most frequent 250 words as the problem vocabulary
2. Generate the train vectors
    2.1. For every file in $A_X$
        2.1.1. Split text in chunks of at least 500 words without breaking paragraphs
        2.1.2. For every chunk
            2.1.2.1.Output a vector with label +1
3. Generate the test vectors
    3.1. For every file in X
        3.1.1. Split text in chunks of at least 500 words without breaking paragraphs
        3.1.2. For every chunk
            3.1.2.1.Output a vector with label -1
4. Unmask
    4.1. For every feature removal iteration
        4.1.1. For every cross validation fold
            4.1.1.1.Exclude the features removed thus far from this fold's data
            4.1.1.2.Train a linear SVM model on this fold's data
            4.1.1.3.Compute the accuracy of this SVM on this fold's data
            4.1.1.4.Find the 3 strongest positive and negative features
            4.1.1.5.Add them to the exclusion list
        4.1.2. Output the average of the SVM accuracy for this iteration
```

Since the code written for [KOPP04] is not available, we implemented a version of the unmasking algorithm above from scratch for this project. As you read the algorithm pseudo-code above, the following questions come to mind:

A. What is the definition of frequency (1.1 and 1.2 above): is it the ratio of term count to document length (TDF) or term count to maximum term count (TF)?
B. Do we employ a stop word list and stem the words[1] before computing their frequency (1.1 and 1.2)?
C. Do we include actual words or their stems in the vocabulary (1.3)?
D. Does the word count for every chunk include stop words or just vocabulary words (2.1.1 and 3.1.1)?
E. The paragraph structure of various texts is very different; do the chunks end up being approximately equal, and does it matter (2.1.1 and 3.1.1)?
F. What is the feature vector representation (2.1.2.1 and 3.1.2.1): is it binary, TF, or TF-IDF?
G. Do the unmasking SVM experiments use a one-class SVM or a multi-class C-SVC (4.1.1.2)?
H. How do we determine the optimal SVM parameters, in particular C and γ (4.1.1.2)?

In our initial unmasking implementation, we selected reasonable, intuitive answers to the questions above, but the accuracy of this implementation was unsatisfactory (nowhere near close to the accuracy in [KOPP04]). We used the same SVM library[2] and the same English corpus (Appendix A) as [KOPP04]. All the texts in the English corpus came from Project Gutenberg[3], except the Emerson texts that came from The University of Adelaide eText Library[4] since they are not available at Project Gutenberg. All these texts were pre-processed by removing automatic copyright pre- and post-ambles, page markers, and other delimiters.

We spent a great deal of time building and tweaking experiments in order to better answer the questions above and improve the accuracy of unmasking. We detail some of this effort in the following subsections.

Vocabulary Selection

To build the vocabulary, we have to answer the following questions:

What is the definition of frequency (1.1 and 1.2 above): is it the ratio of term count to document length (TDF) or term count to maximum term count (TF)?

[1] Although stop words and stemming are standard practice in text classification problems, we could not be sure of its effect in this text *verification* problem. See the important differences and difficulties between classification/attribution and verification in the previous sections.
[2] LIBSVM 2.6: http://www.csie.ntu.edu.tw/~cjlin/libsvm/
[3] Project Gutenberg: http://www.gutenberg.org/
[4] The University of Adelaide eText Library: http://etext.library.adelaide.edu.au/e/emerson/ralph_waldo/

Since the vocabulary consists of the top 250 words with highest average frequency, the choice of frequency will produce different vocabularies. In practice, the actual words in vocabularies produced with TDF vs. TF are largely the same (they differ in fewer than 10 words), so the unmasking accuracy numbers vary little between these two approaches. The length of A can be much larger than the length of X, so the TDF approach produces frequency numbers that are smaller for A vs. X. In contrast, the most frequent term in A may not be much larger than the most frequent term in X, so the TF approach produces frequency numbers that are more similar for A vs. X. We want to have the length of the text reflected in the vocabulary selection, so we prefer the TDF approach.

Do we employ a stop word list and stem the words before computing their frequency (1.1 and 1.2)? Do we include actual words or their stems in the vocabulary (1.3)?

We used a standard English Porter stemmer and a list of 524 English stop words[5] (see `stopwords.en.lst`). Stemming significantly improves accuracy because by removing a feature (a word stem) we remove *all* the words that originate in that stem so unmasking converges within a few feature removal rounds (A and X look similar faster). It is possible that tense selection, and other morphological features that are collapsed by stemming differentiate between authors, but we find that eventually removing these morphological features achieves the same result as stemming, it just takes longer. Eliminating stop words improves accuracy because they are overwhelmingly more frequent than non-stop words, so they conceal words that might otherwise capture characteristics unique to each author; furthermore, these words are used similarly by English authors so they do not reflect authorship.

Vector Representation

To build the feature vectors, we have to answer the following questions:

Does the word count for every chunk include stop words or just vocabulary words (2.1.1 and 3.1.1)?

We included all consecutive words towards the size of the chunk. Intuitively, this should provide a more representative sample of vectors that describes the text since the distribution of feature words is different from the distribution of all the words. More importantly, this will provide more vectors, which means more training data: a chunk that consists only of 500 feature words will be much larger (so there will be fewer of them) than a chunk that consists of 500 consecutive words. The sparseness of the feature vectors (the number of zero entries) is also affected by the chunk word count: if we only consider vocabulary words, the feature vectors end up being denser. Naturally, this affects the accuracy of unmasking significantly.

The paragraph structure of various texts is very different; do the chunks end up being approximately equal, and does it matter (2.1.1 and 3.1.1)?

[5] The stop words came from the Rainbow library: http://www.cs.cmu.edu/~mccallum/bow/rainbow/

The chunks do not end up being approximately equal. More importantly, the number of chunks we obtain with the splitting approach above ends up being very different than the number of chunks reported by the authors (Appendix A). The algorithm accuracy appears to be sensitive to the size of the chunk: we experimented with sizes from 500 to 2000 and ended up settling on 1500 as the optimal chunk size for this problem space. Of interest is the fact that the works by James F. Cooper (*The Last of the Mohicans*, *The Spy*, *Water Witch*) have roughly the same paragraph structure and size (866k, 878k, and, respectively, 907k), and end up with roughly the same number of chunks in our implementation (95, 94, and, respectively, 95). In the authors report, however (Appendix A), they end up with a very different number of chunks (49, 63, and, respectively, 80) which is inconsistent both among the texts as well as with any chunk size from 500 to 2000. Since the algorithm accuracy appears to be sensitive to the size of the chunk, this is an area where additional experimentation would be beneficial.

What is the feature vector representation (2.1.2.1 and 3.1.2.1): is it binary, TF, or TF-IDF?

The TF-IDF representation is favored among document retrieval and text classification approaches, with the binary representation being a close second. However, in this experiment, the TF representation turns out to provide better accuracy than either of the other two. The goal of the IDF term (in TF-IDF) is to dampen the importance of words that appear in many documents and highlight the importance of words that appear in few documents. Intuitively, we believe that the speed with which unmasking converges is intimately tied to the frequency of words: by removing frequent words, the classification accuracy decreases rapidly if A wrote X, and stays roughly the same otherwise. By making all the words look and behave more like each other in TF-IDF, we are essentially putting the breaks on unmasking and making it converge more slowly. It is possible that by running more unmasking iterations with TF-IDF we could obtain the same unmasking accuracy as by running fewer unmasking iterations with TF, but we did not explore this avenue fully.

SVM Setup

To build the feature vectors, we have to answer the following questions:

Do the unmasking SVM experiments use a one-class SVM or a multi-class C-SVC (4.1.1.2)?

In [KOPP04 sec. 5.3] the authors train a one-class RBF kernel SVM ([SCHO00] and [SCHO01]) for the baseline experiments (A_X vs. X), but in [KOPP04 sec. 5.5] they use a linear kernel SVM for the unmasking experiments (without specifying whether it's one-class or C-SVC) and perform cross-validation to assess its accuracy. After some experimentation, we settled on the following approach:

1. Use a linear C-SVC.
2. Label all the A_X vectors with +1, and all the X vectors with -1.

3. For cross-validation, shuffle and fold the data once at the beginning, then preserve the same folds across all the feature removal iterations.
4. The accuracy of cross-validation is the misclassification rate of the C-SVC (the rate of misclassifying a -1 as +1 or vice-versa), which is precisely the extent to which the algorithm can tell X and A_X apart.

How do we determine the optimal SVM parameters, in particular C and γ (4.1.1.2)?

The C-SVC parameters (in particular the cost C and the kernel function γ) are left at the default (C = 1.0, γ = 1 / k where k is the number of features in the data). The way we determined that the defaults work best is by performing an exhaustive grid search over a set of possible (C, γ) pairs[6]:

1. Vary C over $\{2^{-5}, 2^{-4}, 2^{-3}, \ldots, 2^{0} = 1, \ldots, 2^{15}\}$ using step 3 (pick every 3rd element)
2. Vary γ over $\{2^{3}, 2^{2}, \ldots, 2^{-15}\}$ using step 3 (pick every 3rd element)
3. Train a C-SVC using the (C, γ) parameters above and assess its accuracy via 3-fold cross-validation.
4. Select the (C, γ) pair that maximizes the cross-validation accuracy.

In a majority of cases, the highest accuracy was achieved by using the default (C = 1.0, γ = [very small]) combination. The fact that the defaults tend to work well is encouraging because running the grid search for every unmasking experiment (in order to determine the optimal parameters) takes about 30 times longer than running the regular experiment (circa 36 hours), so it is unfeasible in practice.

Given that we employ a linear kernel SVM, we use the definition of feature weight from [BRAN02 sec. 3.1.3]. We compute the normal vector $w = \sum_i \alpha_i y_i \mathbf{x}_i$, where α is the SVM model coefficient for the given support vector, y is the vector label (+1 or -1) and $\boldsymbol{x}$ is the support vector. This normal vector represents the normal to the hyperplane that separates the two classes. In this weight vector, the magnitude of every entry is the weight of the corresponding feature. In a related work ([KOPP03 sec. 6]) the authors suggest scaling the feature weight by its frequency (TDF, TF or TF-IDF). Our experiments suggest that leaving the weight unaltered works best; this is probably because the SVM model already accounts for word frequency in the way it generates the support vectors and, therefore, in the way the normal weight vector comes out.

Updated Algorithm Definition

Based of the observations above, here is the updated algorithm definition (changes highlighted in red):

1. Generate the vocabulary
   1.1. Compute the TDF frequency for all the non-stop word stems in A_X
   1.2. Compute the TDF frequency for all the non-stop word stems in X
   1.3. Create a list of all the common words between A_X and X

[6] Parameter grid search is LIBSVM: http://www.csie.ntu.edu.tw/~cjlin/libsvm/

```
    1.4. Sort it by the average of their TDF frequencies
    1.5. Select the top most frequent 250 words as the problem vocabulary
2. Generate the train vectors
    2.1. For every file in $A_X$
        2.1.1. Split text in chunks of at least 500 words without breaking paragraphs
        2.1.2. For every chunk
            2.1.2.1. Output a TF vector using word stems with label +1
3. Generate the test vectors
    3.1. For every file in X
        3.1.1. Split text in chunks of at least 500 words without breaking paragraphs
        3.1.2. For every chunk
            3.1.2.1. Output a TF vector using word stems with label -1
4. Unmask
    4.1. For every feature removal iteration
        4.1.1. For every cross validation fold
            4.1.1.1. Exclude the features removed thus far from this fold's data
            4.1.1.2. Train a linear C-SVC model on this fold with default parameters
            4.1.1.3. Compute the accuracy of this C-SVC on this fold's data
            4.1.1.4. Find the 3 strongest positive and negative features using only the SVM
                     weight vector and no scaling
            4.1.1.5. Add them to the exclusion list
        4.1.2. Output the average of the C-SVC accuracy for this iteration
```

Meta-learner

The output from the unmasking algorithm is a set of points that defines a degradation curve for the specified A_X, X pair as we iteratively remove features from the model. This curve is labeled as T (if A_X wrote X) or F (if A_X did not write X). We use another meta-learner to learn the labels for these curves and apply to future experiments.

In [KOPP04 sec. 5.4], the authors suggest representing each curve as a set of tuples where for every feature removal iteration *i*, the tuples denote:

- The accuracy after *i* elimination rounds
- The accuracy difference between round *i* and $i + 1$
- The accuracy difference between round *i* and $i + 2$
- The i^{th} highest accuracy drop in one iteration
- The i^{th} highest accuracy drop in two iterations

The authors learn that the following two rules achieve 100% accuracy on the T curves and 97.3% accuracy on the F curves:

- Accuracy after 6 elimination rounds if less than 89%
- The second highest accuracy drop in two iterations is greater than 16%

We briefly experimented with a simple J48 decision tree learner, and found that it has good classification accuracy on these curves, both for the original English corpus and for the test Romanian corpus. The details of these experiments are presented in the next section.

Experimental Evaluation

Methodology

To validate our unmasking implementation, we tested on the same English corpus as the one used in [KOPP04] (see Appendix A). Visually, we would expect one curve to sharply decrease as it moves to the right (the curve of the *actual* author of X) while the other curves should stay mostly flat or decrease slightly as they move to the right (the curve of all the other authors that did not write X). Experimentally, the meta-learner has its own accuracy of detecting the curve that degrades the fastest, which we compare to the results in [KOPP04] and present as the conclusion to the individual degradation experiments.

To determine how applicable unmasking is to other problems, we tested on a corpus or Romanian literary texts (see Appendix B). There were four significant challenges to using Romanian text in this experiment:

1. **Text availability**: There are few long digital Romanian literary texts available on-line. We selected a set of folk fairy tales[7] for this experiment. Both these authors are fixtures of Romanian literature, and folk fairy tales are some of the oldest and most prevalent forms of Romanian literature. We believe that this body of work, although limited, is sufficiently representative of Romanian literature in general.
2. **Spelling**: In 1989, after the change of regime in Romania, the official spelling rules were altered to the way they were before 1945. Many extant digital texts may not have been updated to reflect these spelling changes. This could severely influence the accuracy of our unmasking algorithm if we tried to unmask a new-style text against an old-style author corpus, with different spelling. Romanian has a number of diacritic characters that do not exist in English (â, î, ă, ş, and ţ) and very few digital texts go to the trouble of representing these accurately in Unicode. This is another reason why it is important to use text from corpuses that were represented the same way.
3. **Stop words**: There is no known existing set of Romanian stop-words, so we used a direct translation of the set of English stop-words. Many English stop-words (such as "please" or "mostly") translate to multiple-word Romanian stop-phrases (such as "te rog" or, respectively, "în cea mai mare parte", see `stopwords.ro.lst` for details). Changing the chunking algorithm to account for stop-phrases is sufficiently complex that we decided to not do it in this project due to time constraints. Fortunately, this turns out to have little effect on accuracy, since sufficiently many one-word English stop-words translate to one-word Romanian stop-words.
4. **Grammar**: The Romanian grammar is similar in complexity to Latin, in that it has genders, declinations, highly irregular verbs, and many more verb tenses than English. Implementing even a moderately accurate Romanian stemmer is incredibly difficult, far more so than for English. The function of the word (adjective vs. noun vs. etc.) in the context determines its stemming rules. Furthermore, if the appropriate diacritics are not used, words can become even

[7] Romanian Folk Fairy Tales: http://www.romanianvoice.com/culture/povesti/index.php

more ambiguous, which makes stemming more difficult (for example, “tari” means “strong” [adjective, plural, masculine], while “țări” means “countries” [noun, plural, feminine]). For this project, we implemented a very simple Romanian stemmer that simply eliminates (or in a few cases replaces) the word endings that are unambiguously known to have a specific morphological function (see `StemmerRo.java` for details).

Despite these severely simplifying assumptions in unmasking Romanian text, we believe that the performance of unmasking on this new body of work is very telling. Romanian folk stories tend to have very distinctive regional and temporal flavor, far more so than the texts from the original English corpus, so we are interested in determining if unmasking is really unaffected by these literary styles. Furthermore, the first known text written in Romanian was a letter[8] from the year 1521, using the Cyrillic alphabet (the Latin alphabet was adopted many years later); prior to this, there is a lot of evidence of oral literature and poetry, but nothing written down. As a result, we believe that there is significant variance in writing styles across preserved Romanian literary texts, which makes them a particularly good test for unmasking. We present the results for the Romanian corpus in the same way as for the English corpus: visually, in terms of degradation graphs, and experimentally, by using the meta-learner described above.

Results

English Corpus

See Appendix A for the contents of the original English corpus. At a high level, there are two cases where unmasking completely fails to distinguish between A_X and X: Figure 6: Cooper *The Spy* and Figure 17: Shaw *Pygmalion*. In [KOPP04 sec. 5.5], the authors indicate that their classifier also missed Shaw’s *Pygmalion*, so this is encouraging, however the fact that Cooper’s *The Spy* is also utterly misclassified is disconcerting. Furthermore, the quality of our other degradation curves is not always very good (visually speaking). Of particular interest is the fact that almost all the Bronte sisters curves[9] start out fairly well separated from the other degradation curves, but then flatten out, while the other degradation curves descend past them (see Figure 7: Anne Bronte *Agnes Gray*, Figure 20: Anne Bronte *Tenant of Wildfell Hall*, Figure 13: Charlotte Bronte *Jane Eyre*, and Figure 18: Charlotte Bronte *The Professor*). This indeed very peculiar behavior, both because it does not match our expectations, and also because the degradation pattern is so similar among these related authors!

In the interest of time, we did not analyze all the departures from expected behavior noted above; however we *did* analyze in detail the case of Cooper’s *The Spy* as it is the most aberrant of them all (based on the degradation curves in Figure 6, it would seem that *The Spy* has been authored by a majority of the authors in our corpus, so strong and similar is the degradation of these authors’ curves). We created a naïve, yet useful framework for manually analyzing *The Spy* and for understanding its unmasking behavior.

[8] The Letter of Neacşu from Câmpulung: http://www.cimec.ro/Istorie/neacsu/eng/default.htm

[9] The Jane Eyre unmasking scenario is missing, since we only have one Jane Eyre book in the corpus (*Wuthering Heights*), so we cannot build a degradation curve of *Wuthering Heights* against itself.

We select the features that were consistently removed in all the 10 feature removal iterations. By consistently we mean that a feature was selected for removal in at least 5 of the 10 cross-validation runs executed in every feature removal iteration. The idea is that these features are (by a majority vote) more likely to discriminate among the texts considered than the other features. We then compute the delta between the occurrence count of these features in A_X and X and we look for patterns. Naturally, we realize that the count is a poor proxy for SVM feature strength; however we believe it is sufficient for this naïve, manual analysis.

The size of the delta is important:

- If the delta is high, then this feature is probably a strong discriminator
- If the deltas are high and stay high, then there are lots of strong discriminators, so the X was probably not written by A
- If the deltas are high but they drop quickly, there are few strong discriminators. so X was probably written by A

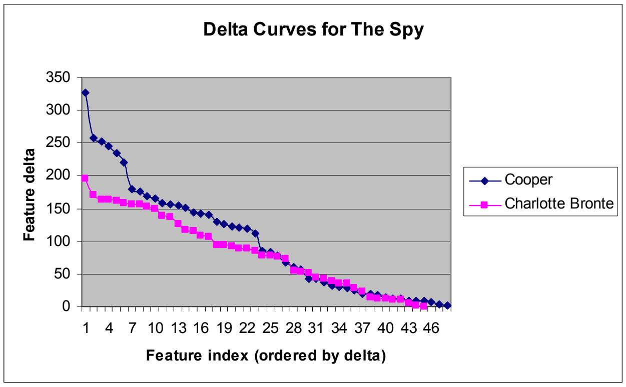
The relative placement of the delta curves between two author candidates (A_1 and A_2) and X is important:

- If one delta curve (say A_1) is consistently above the other (A_2), then one relative discriminator strength is higher, so X is unlikely to have been written by A_1 and more likely to have been written by A_2
- If the two delta curves are indistinguishable, then the relative discriminator strength is the same, so we cannot be sure about authorship

Lastly, we look at how many discriminators there are in common between A_1 and A_2:

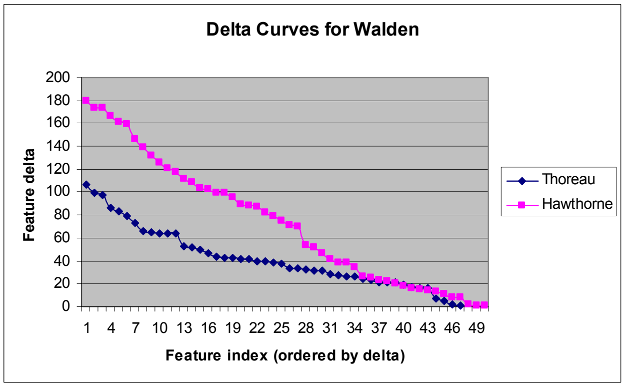
- If there are lots of common discriminators, then we cannot make a strong statement about authorship, since the same discriminators can be used to distinguish between A_1 and A_2
- If there are few common discriminators, then we can be more confident about attributing authorship

We apply all these observations, in concert, to the scenario for *The Spy* (which is particularly aberrant) as well to the scenario of *Walden* (which is particularly regular). Here are the delta curves for Cooper vs. *The Spy* and Charlotte Bronte vs. *The Spy*:



Remember that in Figure 6 the corresponding degradation curves are almost indistinguishable. In this case, the two delta curves start out fairly close together, and then quickly merge. This means that the discriminator strength (as evidenced by the feature delta) is roughly the same between Cooper and Charlotte Bronte, so it is no surprise that we cannot ascribe authorship to *The Spy* when pitted against these two authors.

Now take a look at the delta curves for Thoreau vs. *Walden* and Hawthorne vs. *Walden*:



In this case, the two delta curves start out fairly far apart, and then continue to stay separated. This means that the discriminator strength (as evidenced by the feature delta) is fairly different between Thoreau and Hawthorne, so it is no surprise that we can safely ascribe authorship to *Walden*. Furthermore, there are 12 common discriminators between Thoreau and Hawthorne, whereas there are 25 common discriminators between Cooper and Charlotte Bronte. The fact that there are fewer common discriminators between Thoreau and Hawthorne means that they are more likely to be separated via unmasking than Cooper and Charlotte Bronte.

Leaving statistics and graphs aside for a moment, what is the real (literary or stylometric) reason why *The Spy* would appear so similar to two completely different authors (Cooper, the original author, and Charlotte Bronte)? The most likely explanation is that the bag-of-words approach it too coarse and fails to capture subtler idiosyncrasies of these authors' writing styles. Both Cooper and Charlotte Bronte lived and wrote in the first half of the

19th century, and although the former was American and the latter British, perhaps they shared some common characteristics of the literary currents of the time.

We ran the Weka[10] J48 decision tree algorithm on the set of degradation curves (represented simply as the set of points that define the curve and a T or F label):

```
point7 <= 75.714286
|   point7 <= 71.333333: T (11.0)
|   point7 > 71.333333
|   |   point7 <= 72.5: F (4.0/1.0)
|   |   point7 > 72.5: T (3.0)
point7 > 75.714286: F (182.0/5.0)
```

The classification criterion for T curves is “`point7 <= 71.3 OR (point7 > 72.5 AND point7 < 75.7)`”. This is not as simple or as intuitive as the criterion from [KOPP04] (“Accuracy after 6 elimination rounds if less than 89%” AND “The second highest accuracy drop in two iterations is greater than 16%”), but it achieves reasonably high accuracy (70% for T, and 100% for F) as seen from the confusion matrix:

```
   a   b   <-- classified as
  14   6 |   a = T
   0 180 |   b = F
```

Interestingly, our J48 classifier was able to distinguish among these curves by means of a single point (point7), which happens to be the same point that discriminates in [KOPP04] (“after 6 elimination rounds”). The rule that involves point7 above spans two intervals as opposed to just one (as in [KOPP04]). Since we did not represent the accuracy drop in our degradation curves’ feature vectors, we could not involve these drops in the J48 rules, but it is intuitively likely that these additional degradation features would provide a stronger, more accurate meta-classification criterion.

We are now going to look at the complete set of degradation graphs obtained on the English corpus. The important thing to look for in these graphs is whether there is one curve that is clearly separated and under all the other curves (the curve of the actual author of our mystery text) while all the other curves stay roughly flat. The better this curve separates from the others, the more accurate and confident is our authorship verification result. The legend for these curves is presented after all the graphs.

[10] Weka, Data Mining Software in Java: http://www.cs.waikato.ac.nz/ml/weka/

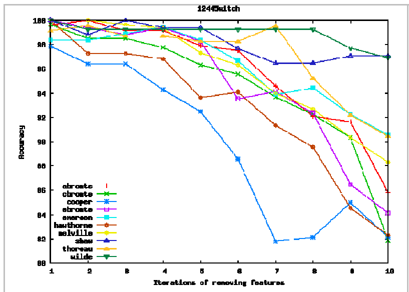


Figure 1: Cooper *Water Witch*
The Cooper curve is fairly well separated, but strong degradation occurs everywhere.

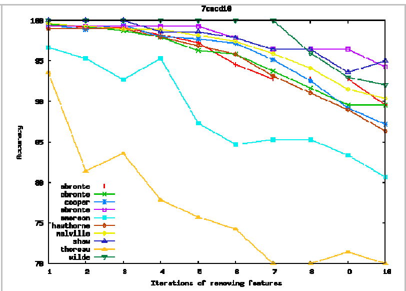


Figure 2: Thoreau *A Week On Concord*
The Thoreau curve is very well separated, but the Emerson curve also breaks from the rest.

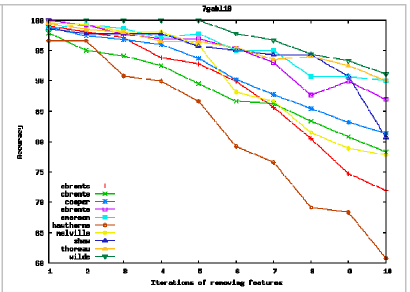


Figure 3: Hawthorne *House of Seven Gables*
The Hawthorne curve is well separated, but strong degradation occurs everywhere.

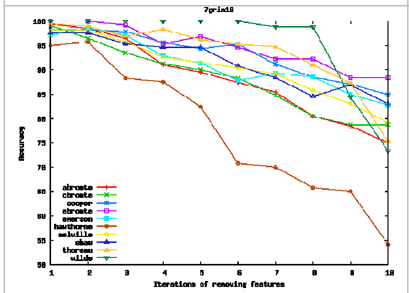


Figure 4: Hawthorne *Dr. Grimshawe's Secret*
The Hawthorne curve is very well separated.

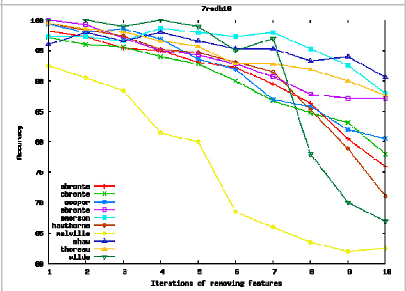


Figure 5: Melville *Redburn*
The Melville curve is well separated, but strong degradation occurs everywhere.

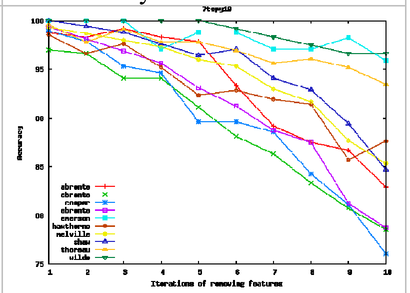


Figure 6: Cooper *The Spy*
There is no clear separation among the curves!

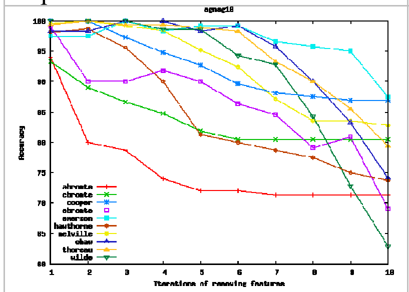


Figure 7: Anne Bronte *Agnes Gray*
The Anne Bronte curve starts out fairly well separated, but all the other curves catch up.

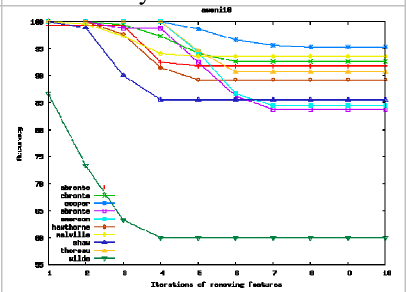


Figure 8: Wilde *A Woman of No Importance*
The Wilde curve is very well separated.

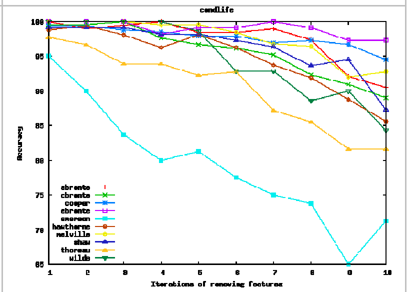


Figure 9: Emerson *Conduct of Life*
The Emerson curve is very well separated.

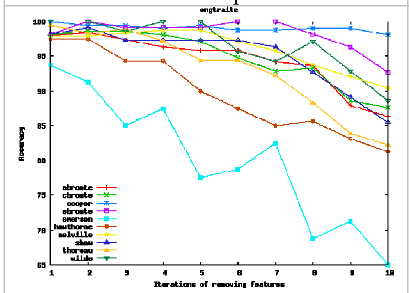


Figure 10: Emerson *English Traits*
The Emerson curve is well separated, but jittery.

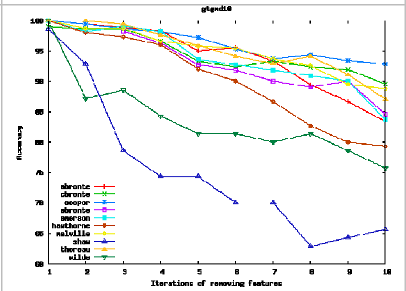


Figure 11: Shaw *Getting Married*
The Shaw curve is fairly well separated.

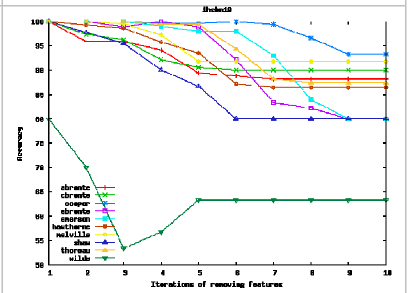


Figure 12: Wilde *An Ideal Husband*
The Wilde curve is very well separated.

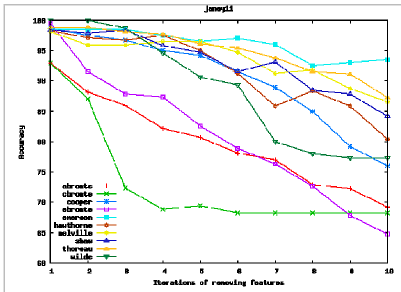


Figure 13: Charlotte Bronte *Jane Eyre*

The Charlotte Bronte curve starts out fairly well separated, but all the other curves catch up.

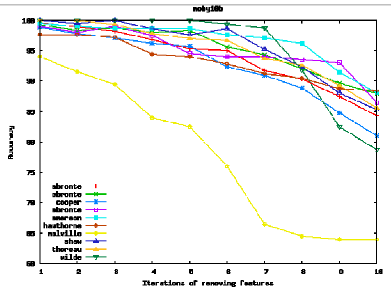


Figure 14: Melville *Moby Dick*

The Melville curve is very well separated.

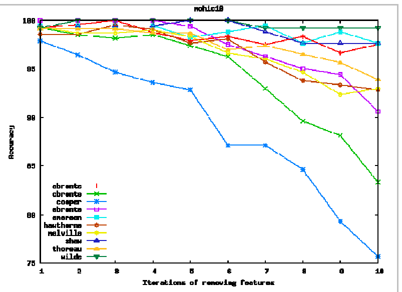


Figure 15: Cooper *The Last of the Mohicans*

The Cooper curve is well separated.

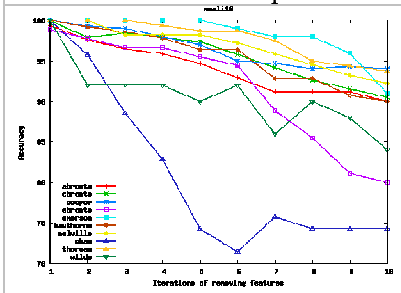


Figure 16: Shaw *Misalliance*

The Shaw curve is fairly well separated.

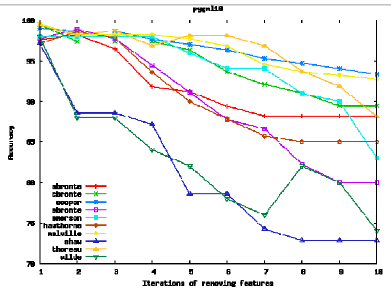


Figure 17: Shaw *Pygmalion*

There is no clear separation among the curves!

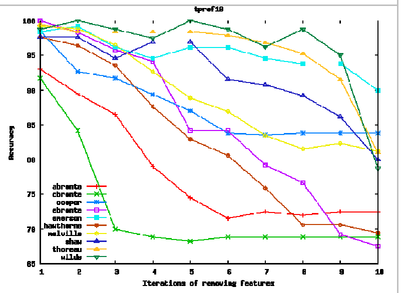


Figure 18: Charlotte Bronte *The Professor*

The Charlotte Bronte curve starts out fairly well separated, but all the other curves catch up.

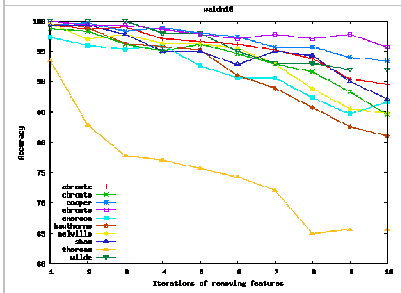


Figure 19: Thoreau *Walden*

The Thoreau curve is very well separated.

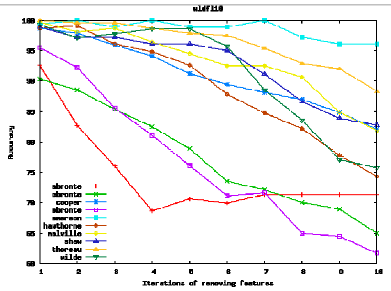


Figure 20: Anne Bronte *Tenant of Wildfell Hall*

The Anne Bronte curve starts out fairly well separated, but all the other curves catch up.

Legend:

abronte —+—
cbronte —×—
cooper —✳—
ebronte —□—
emerson —■—
hawthorne —⊕—
melville —⊕—
shaw —△—
thoreau —▲—
wilde —▽—

Romanian Corpus

See (Appendix B) for the contents of the Romanian corpus. Here are the degradation curves:

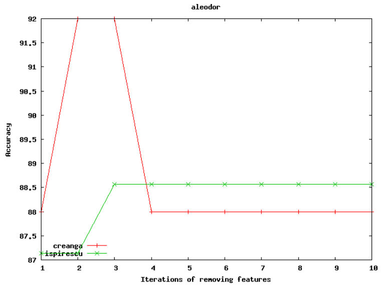


Figure 21: Ispirescu's *Aleodor Imparat*

The Ispirescu curve loses!

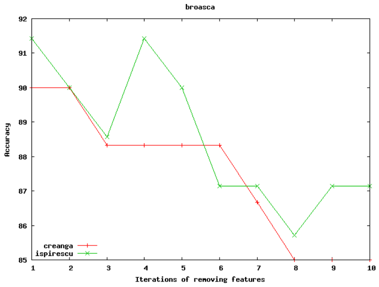


Figure 22: Ispirescu's *Broasca Testoasa cea Fermecata*

The two curves are difficult to distinguish.

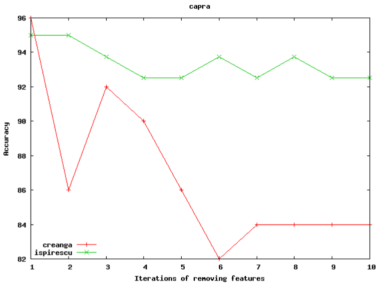


Figure 23: Creanga's *Capra cu Trei Iezi*

The Creanga curve clearly wins.

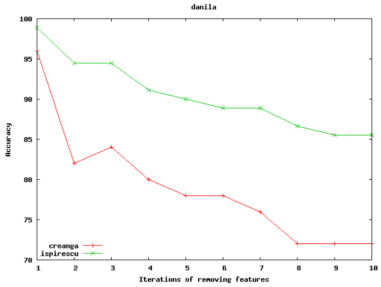


Figure 24: Creanga's *Danila Prepeleac*

The Creanga curve clearly wins.

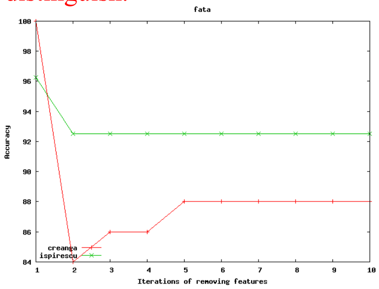


Figure 25: Creanga's *Fata Babei si Fata Mosneagului*

The Creanga curve clearly wins.

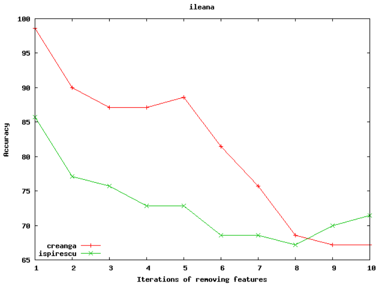


Figure 26: Ispirescu's *Ileana Simziana*

The Ispirescu curve clearly wins.

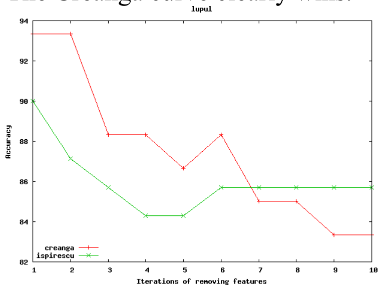


Figure 27: Ispirescu's *Lupul cel Nazdravan*

The Ispirescu curve starts out well, but the Creanga curve catches up.

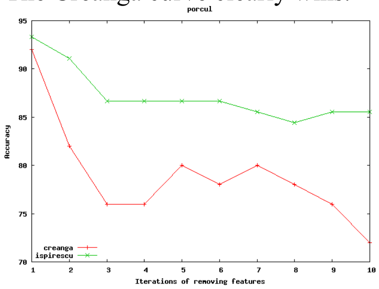


Figure 28: Creanga's *Povestea Porcului*

The Creanga curve clearly wins.

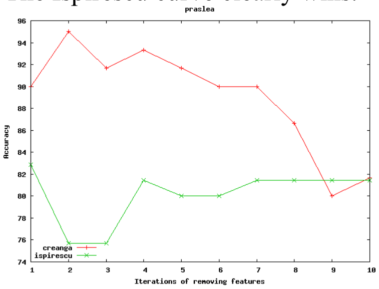


Figure 29: Ispirescu's *Praslea Cel Voinic si Merele de Aur*

The Ispirescu curve clearly wins.

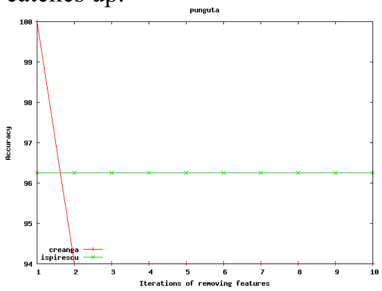


Figure 30: Creanga's *Punguta cu Doi Bani*

The Creanga curve clearly wins.

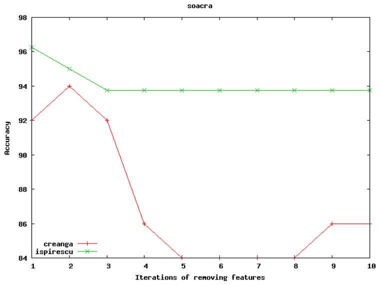


Figure 31: Creanga's *Soacra cu Trei Nurori*

The Creanga curve clearly wins.

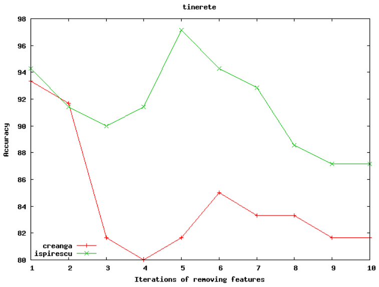


Figure 32: Ispirescu's *Tinerete fara Batrinete si Viata fara de Moarte*

The Ispirescu curve clearly loses!

Legend:

creanga —+—

ispirescu —×—

Despite three cases where the results are backwards or inconclusive, unmasking performs well on this data set! In general, the Creanga curves perform far better than the Ispirescu curves. It is difficult to attribute a single likely explanation to this given that our stop-word and stemming approach is far from perfect, and we use a much smaller number of words per chunk in these experiments (250 vs. 1500 in the English corpus) since the Romanian texts are much smaller. In sum, any of these factors could contribute to this behavior.

We run the J48 decision tree algorithm on the set of degradation curves (represented simply as the set of points that define the curve and a T or F label):

```
point2 <= 87.142857: T (8.0)
point2 > 87.142857
|   point10 <= 85.714286: F (7.0)
|   point10 > 85.714286
|   |   point9 <= 87.142857: T (3.0)
|   |   point9 > 87.142857: F (6.0/1.0)
```

The classification criterion for T curves is “`point2 <= 87.1 OR (point10 > 85.7 AND point9 <= 87.14)`”. This criterion achieves very high accuracy (91% for T, and 100% for F) as seen from the confusion matrix:

```
  a  b   <-- classified as
 11  1 |  a = T
  0 12 |  b = F
```

Overall, unmasking performs well on the Romanian corpus, despite the notable limitations described above and despite using only a fraction of the training vectors from the English corpus experiment. This is another point in favor of the robustness and applicability of the unmasking algorithm.

Discussion

On the positive side, we demonstrated that the algorithm is indeed robust when it comes to running on texts from another language (Romanian), in the presence of a very impoverished stop-word and stemming approach, and on a fraction of the number of feature vectors from the original English corpus experiment.

On the negative side, we were not able to obtain results as impressive as those described in [KOPP04] in terms of accuracy on the English corpus. Specifically, unmasking does not work at all in the case of Cooper’s *The Spy*, and in general our degradation curves are less well visually and programmatically separable for all the texts by the Bronte sisters. One likely reason for this is that we were not able to chunk the original texts in the same number of chunks as the original authors, and the algorithm seems to be sensitive to the number and sparseness of feature vectors. Another likely reason is that we had to intuit the settings and architecture for many of the algorithm parameters, as they were not explicitly described in the original paper, and it is possible that some subtle alteration that we did not explore may positively and dramatically influence the performance of the algorithm.

Overall, the algorithm performs remarkably well on a traditionally difficult problem. Even in the absence of a negative (non-A) training set, the algorithm is able to recognize authorship for some unknown text X with very high fidelity. The reason is that the algorithm looks for the depth of difference between two sets, and uses it to determine membership in that set, as opposed to relying on external (negative) sets to provide counter-examples. This method makes few assumptions about how the training set is represented, and, therefore, the algorithm is very likely easily transferable to other domains, as seen in the Romanian corpus experiment.

A closely related algorithm in the visual domain is implemented in the Fast Multiresolution Image Querying[11] project. Briefly, in this project a database of images is reduced to a very coarse wavelet representation, and the user queries this database by drawing a sketch of the image they wish to find. It turns out that the sketch can be easily matched against the wavelet thumbnails, so the method has high accuracy and is easy to use. The idea behind unmasking is similar in spirit: we strip down both the author corpus and the query text by removing all essential features, and whatever is left can be easily matched as an indicator of similarity or difference. The unmasking case is, in a way, the dual of the wavelet problem: for unmasking we remove all major features, whereas for the wavelet query we remove all minor features.

Future Work

The meta-learner model for the degradation of English curves is clearly different from the model for the Romanian curves. We suspect that this would likely happen across domains, but the domain boundaries are unclear: is all English literature one domain, or are there sub-domains within it? In general, how transferable are the meta-learner rules from one domain to another?

The size of the chunk is difficult to select a-priori. A chunk that is too high produces too few high-quality (dense) vectors, whereas a chunk that is too small produces too many low-quality (sparse) vectors. The algorithm appears to be quite sensitive to the chunk size and the sparseness of the feature vectors. A set of heuristics has to be developed in order to make the appropriate choice for this parameter.

Not surprisingly, unmasking depends on eliminating stop words (as explained in the section on Vocabulary Selection). What is slightly surprising is that unmasking is also rather dependent on stemming. If we do not stem at all in either the English or the Romanian corpus experiments, the unmasking accuracy drops dramatically. Conversely, if we stem (even poorly!) in the Romanian corpus experiment, the accuracy improves noticeably. It would be illuminating to produce a learning curve where the accuracy of unmasking is graphed against the quality of stemming to see at what point stemming

[11] Fast Multiresolution Image Querying: http://grail.cs.washington.edu/projects/query/

provides marginal benefits or even gets in the way. It would also be interesting to think about what stemming would mean in a non-literary domain.

The way we selected among possible choices for the algorithm parameters (e.g. the vector representation: TF vs. TDF vs. TF-IDF) was by running extensive experiments and visually comparing the degradation graphs. With more time available, a better way would be to use statistical significance tests on the accuracy of the meta-learner and simultaneously search across all parameter combinations. Of particular interest for such rigorous testing are the feature vector representation techniques and the vocabulary selection: the authors suggest evenly averaging the word frequency from the positive and the negative set, regardless of the relative sizes of these two sets, in order to give them equal weight; perhaps a more refined approach would yield better results?

In [KOPP04 sec. 6], the authors propose an extension to the algorithm where they use both positive and negative examples (a negative example would be works of authors that share the same geographical, chronological, and cultural characteristic as the author we wish to unmask). This extension helps eliminate additional false positives. It would be interesting to implement this extension and test it on the same corpus of Romanian literature to see how much it improves the accuracy.

In [KOPP04 sec. 7], the authors apply unmasking to solve an actual literary mystery involving a Jewish religious text. There are likely many other such controversial authorship problems (for example Shakespeare vs. Marlowe) and it would useful and fun to apply unmasking to them and interpret the results.

It would be interesting to apply unmasking to another domain that has nothing to do with literature, for example music or visual art. The most interesting aspect in such an experiment would be the representation, chunking, and stemming techniques for these domains.

Conclusion

We implemented the unmasking algorithm and validated an experiment on an extensive English corpus as described in [KOPP04]. In the process, we highlighted the parameters that must be tuned in order to improve the accuracy of the unmasking algorithm. We performed an additional experiment, on a limited Romanian corpus, and showed that the performance of unmasking is very robust on a brand new domain, even in the presence of a significantly impoverished stop-word and stemming scheme for the Romanian language. This strongly reinforces the idea that unmasking can transfer well to new domains. Lastly, we suggested a number of important open questions, primarily around further tuning of parameters that affect the performance of unmasking and around domain transfer.

Appendix A

The English corpus from [KOPP04]:

Group	**Author**	**Book**	**[KOPP04] # Chunks** (word count 500)	**Our # Chunks** (word count 1500)
American Novelists	*Hawthorne*	Dr. Grimshawe's Secret	75	43
		House of Seven Gables	63	107
	Melville	Redburn	51	121
		Moby Dick	88	55
	Cooper	The Last of the Mohicans	49	99
		The Spy	63	94
		Water Witch	80	95
American Essayists	*Thoreau*	Walden	49	74
		A Week on Concord	50	42
	Emerson	Conduct Of Life	47	40
		English Traits	52	65
British Playwrights	*Shaw*	Pygmalion	44	58
		Misalliance	43	75
		Getting Married	51	133
	Wilde	An Ideal Husband	51	36
		Woman of no Importance	38	20
Bronte Sisters	*Anne*	Agnes Grey	45	21
		Tenant Of Wildfell Hall	84	71
	Charlotte	The Professor	51	69
		Jane Eyre	84	14
	Emily	Wuthering Heights	65	20

Appendix B

The Romanian corpus:

Author	**Book**	**Chunks** (word count 250)
Ion Creanga	Soacra cu trei nurori	7
	Capra cu trei iezi	8
	Punguța cu doi bani	3
	Dănilă Prepeleac	13
	Povestea porcului	14
	Fata babei şi fata moşneagului	6
Petre Ispirescu	Tinerețe fără bătrânețe şi viață fără de moarte	11
	Ileana Simziana	24
	Aleodor Împărat	8
	Broasca țestoasă cea fermecată	9
	Lupul cel năzdrăvan şi Făt-Frumos	10
	Prâslea cel voinic şi merele de aur	15

Appendix C

The following additional resources for this project are available on-line at http://www.cs.utexas.edu/~surdules/cs391l/final/:

1. Complete source: `code.zip`. The Java files are in the `src` subdirectory. The precompiled binary (`unmask.jar`) has one dependency (`JSAP_1.03a.jar`). You can run the code using Java 1.4.2 or later (http://java.sun.com/j2se/1.4.2/):

```
Usage: java edu.utexas.cs391l.surdules.Unmask
                <Author> <Text> [--words-chunk <Words Chunk>] [--stop-words
<Stop Words>] [--stemmer <Stemmer>] [--verbose <Verbose>] [--dump <Dump>]

  <Author>                         The author against whose works we unmask.

  <Text>                           The text that we unmask against the author.

  [--words-chunk <Words Chunk>]    The number words per chunk in the text
                                   files.

  [--stop-words <Stop Words>]      The file containing the stop words.

  [--stemmer <Stemmer>]            The stemmer to use for stemming ('English'
                                   or 'Romanian').

  [--verbose <Verbose>]            Turn on logging.

  [--dump <Dump>]                  Dump vector files created during training
                                   and testing.
```

2. The English corpus: `gutenberg.zip`. The texts for every author are available in eponymous subdirectories.

3. The Romanian corpus: `romania.zip`. The texts for every author are available in eponymous subdirectories.

4. This report in PDF format: `report.pdf`.

References

[KOPP04] M. Koppel and J. Schler, “Authorship Verification as a One-Class Classification Problem”, KDD 2004.

[KOPP03] M. Koppel, S. Argamon, and A. R. Shimoni, “Automatically Categorizing Written Texts by Author Gender”, Literary and Linguistic Computing, June 2003.

[BRAN02] J. Brank, M. Grobelnik, N. Milić-Frayling, and D. Mladenić, “Feature Selection Using Linear Support Vector Machines”, Microsoft Research, Technical Report MSR-TR-2002-63, 12 June 2002.

[SCHO00] B. Schölkopf, A. Smola, R. Williamson, and P. L. Bartlett, “New support vector algorithms”, Neural Computation, 12, 2000, 1207-1245.

[SCHO01] B. Schölkopf, J. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. “Estimating the support of a high-dimensional distribution”, Neural Computation, 13, 2001, 1443-1471.