EVALUATING AUTOMATIC SUPPORT FOR WRITING BY EXAMPLE

Katie Kuksenok and Hao Lu

As students, we often wonder how we can write better. What if we had a system, which could provide direct, actionable cues that would help us produce writing that resembles our best work, as opposed to our worst? Or writing text that is more typical of a certain venue or genre? There are few tools to support writing tasks beyond enforcing grammar and spelling rules, and none to support writing to mimic some specific, conceptually-defined class of documents. We explore the feasibility of automatically learning differences between categories in a large conceptual space using low-level linguistic features of texts.

INTRODUCTION

We tested the feasibility of classification methods for supporting *writing by example*, where the author seeks to mimic aspects of other texts, defined by some conceptual categorization. Writing by example support does not have a specific goal, promising instead a nebulous demonstration of "how a *writing draft* compares to *m example classes*" (Fig. 1). In testing the feasibility of writing by example support, we have focused on two interpretations of that goal, each requiring different classification mechanisms.

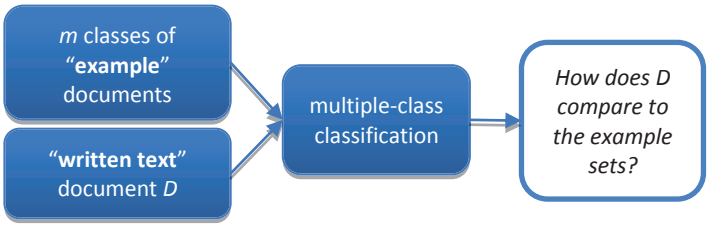


Figure 1: General formulation of writing by example support

We interpret the goal first as, "*which of the* m *example classes is* D *most like?*" This lends itself to *multiclass* classification. We also interpret it as, "*how similar is* D *to each of the* m *example classes?*" This can be more directly addressed using *binary* classification identifying a given pair of documents as *"same class/different class*." At the user interface level, either classifier can serve as "litmus test" for determining whether a similarity condition has been met, or as a source of continuous feedback for evaluating whether successive drafts are achieving such a condition.

We experimentally tested the feasibility of each of the two interpretations of such a system. The goal of experiments was to *explore whether low-level features could be used to find meaningful differences between different, conceptually-defined classes of documents*. In the following pages, we will describe related work, present experimental findings, and conclude with a discussion and example application interfaces that might leverage our classification approaches.

BACKGROUND

Although we are not aware of work specifically aimed at building or testing systems for supporting writing by example, this approach has the potential lead to a highly useful application supporting a yet-unsupported writing task. There are many ways that NLP techniques can help support better writing. Checking spelling and grammar, and similar "rigid-language tools" that work by detecting violations of rules do not cover the range of needs users have in composing written material. For example, the *AwkChecker* tool helps non-native speakers of a language test how "awkward" a phrase is relative to n-grams gathered from a large corpus, which includes Wikipedia [7].

The analysis of written text can make use of a range of complex feature extraction methods and algorithms. Exposing those algorithms enables gaining the user's trust by building understanding of the system. This strategy has become increasingly relevant in other domains, such as context-aware sensing, where the primary functionality of a tool requires that the user trust the output of apparently mysterious learning algorithms [4]. Prior work in literary analysis has noted the potential utility of providing visual feedback of analysis results as a means to communicate ways of improvement, not only as means of gaining a deeper understanding of a literary text [3]. In the case of a writing support tool, the specifics of the NLP techniques involved can either enhance or detract from the resulting system's usefulness; therefore the choices we have support not only comparison but also actionable suggestions as part of system output.

TWO DATASETS: *GUTENBERG* AND *ACM*

We used two datasets, one of literary works scraped from Project Gutenberg[1] and another of abstracts from the ACM Digital Library[2]. After initial baseline results, we loaded all the data, segmented[3], into a MySQL database. The ACM dataset contained example documents illustrating Computer Science publication abstracts for 296 venues, and the Gutenberg dataset contained example documents for about 3,500 authors. The Gutenberg dataset was roughly 100 times the size of the ACM dataset, with 12 times the number of distinct *classes*, which were *venues* and *authors*, respectively (Table 1).

Dataset	# docs	# sentences / document			class	# classes	# documents / class		
ACM	117,823	min=1	max=170	avg≈6	venue	296	min=4	max=4918	avg≈400
Gutenberg	12,912,372	min=2	max=1362	avg≈148	author	3654	min=1	max=1444	avg≈24

Table 1: Details about the size of each dataset

Since the abundance of documents per class helps with interpreting subsequent results, *Figure 2* provides some deeper information about how big classes are in the two datasets. For example, it shows that there are roughly 40 ACM venues that have less than 25 documents associated with them, but there are 200 Gutenberg authors who have exactly 1 document associated with them. Unsurprizingly, many conferences are rather large, while most authors are not especially prolific.

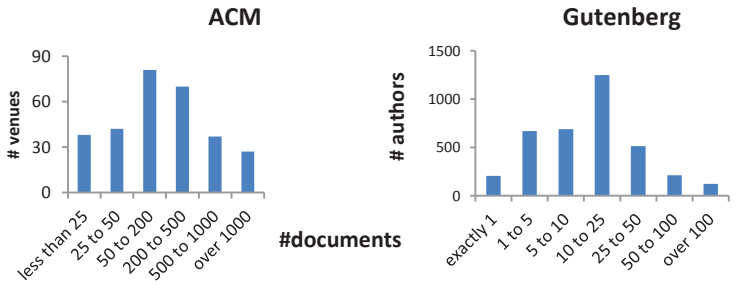


Figure 2: Document abundance across classes in both datasets. Each bar indicates the *number of classes* (venue for ACM, author for Gutenberg) that have the specified *number of documents*. Bin sizes were chosen by hand are not consistent.

TWO CLASSIFICATION APPROACHES

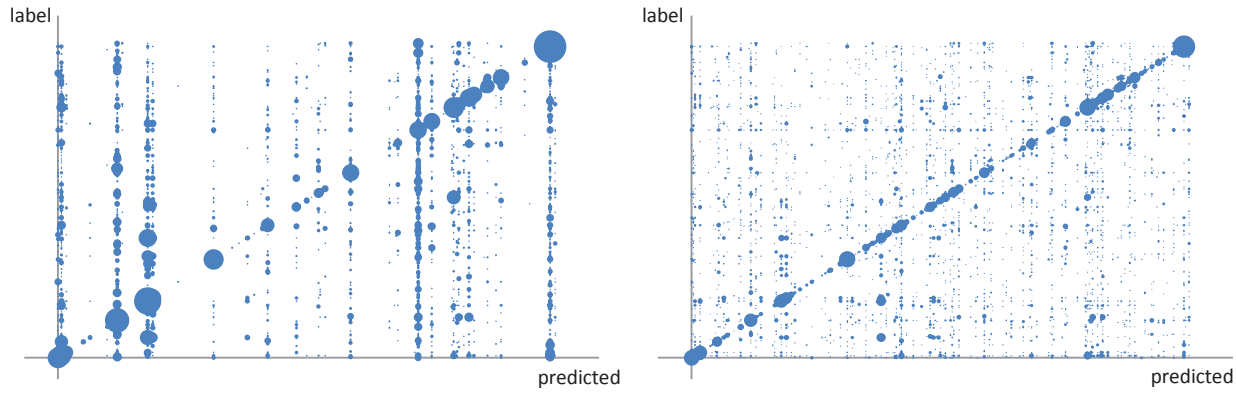
We experimented with two classification approaches: multiclass and binary classification. Experimental setup and results are presented for each in the following two sections.

MULTICLASS

Text classification is a classic problem and has been well discussed in the NLP textbooks. A commonly used example is to classify online news-group articles to groups. Naïve Bayes has been an effective approach for this class of problems. We revisit text classification in our project in a much larger scale in terms of the number of classes: we want to classify papers (abstracts) to their venues.

The challenge is that we have 296 venues in our ACM dataset. A random guess will only result 0.3% accuracy on average. In contrast, in a 20 news groups setting, a random guess will result 5% accuracy on average. As baseline system, we use Naïve Bayes with bag-of-words features. We did a 10-cross validation in rainbow [5]. The average accuracy is around 29% (we got 25% initially because we included empty papers). Considering that we have 296 classes, accuracy 29% is actually not bad.

[1] http://www.gutenberg.org
[2] http://portal.acm.org
[3] ACM data used NLTK (http://www.nltk.org/) for segmentation, and the Gutenberg data used JTextPro [8]



(a) (b)

Figure 3: Confusion matrices: (a) with no feature selection (b) with feature selection

In order to try different ideas, we ran an experiment system in Java using MALLET [6], a statistical natural language processing toolkit. To reduce memory requirement and overfitting, we first tried altering the feature selection technique. Instead of using all the words as features, we kept the top 10,000 words according to their information gain. The best number of words was determined experimentally. This gave us a big boost in performance. The result was an average of 38% accuracy over 10 runs. Figure # shows the visualization of the confusion matrices from the two runs with and without feature selection on a separate test data. When there is no feature selection, the classification is dominated by a few classes probably due to overfitting (Figure 3a). With feature selection, such domination has disappeared (Figure 3b).

We noticed that the venues in similar areas can be hard to be distinguished from each other. For example, UIST and CHI are two top venues in HCI. Many papers can be published in either one of them. The fact that some venues are closer to each other than the other ones leads to the clusters of the venues.

There are two ways that the clusters of venues can potentially help. First, it could help the classification task through a two-stage classification. In the first stage, papers are classified into clusters using a classifier trained on the original training data with labels changed from venues to clusters. In the second stage, papers are classified into venues using classifiers individually trained on each cluster. This process is shown the Figure 4. The hope is that we can be more specific when training the classifiers for each cluster so that we can distinguish the subtlety between venues within a cluster.

Secondly, the clusters of venues could help us better evaluate our classifiers. The errors made by the classifier could be irrelevant when the error is within a cluster, for example, a paper is classified as UIST instead of CHI. Thus it is more interesting and valuable to know how many errors that a classifier makes between clusters.

It is possible to manually label these clusters, given that there are only 296 venues. However, it is non-trivial to get enough researchers to cover all venues. Also for the limited time of the class project, we compute the clusters automatically. The computation is based on heuristics based on the author data. The basic idea is that when an author publishes multiple papers in both venues, it is likely that the two venues are related. When there are many such authors, such guess gets more evidence and

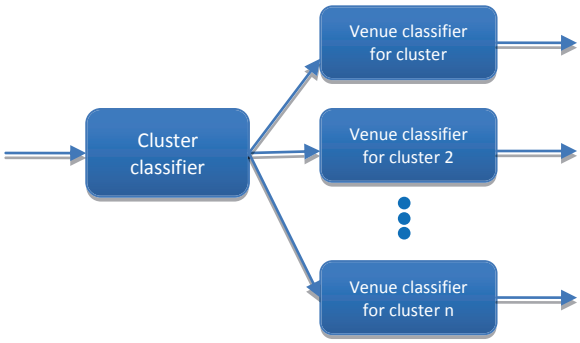


Figure 4: two-stage classification

becomes more likely. The actual algorithm is as follows. We first compute all the authors that have published no less than N papers in at least two different venues respectively. We then bind two venues together if they share more than M such authors. Next, we compute the closure of these connections and get the clusters of the graph.

We tried several (N, M) pairs. We found that there aren't a lot of big clusters. Reducing N and M will reduce the number of clusters but tend to group everything together. In our experiments, the pair (4, 20) gives some reasonable clusters, but it also results 275 clusters with 240 of them containing only one venue. Some of the clusters are listed in table 2. They are not complete but reasonable. We also tried clustering using K-Means with authors as features, but failed to get a good result.

DATE, ASPDAC, ISSS ISPD, CODES, GLSVLSI, DAC, EURO-DAC, ICCAD, ISLPED
SIGIR, WWW, CIKM, KDD, PODS, SIGMOD
PODC, SoCG, SPAA, STOC, SODA
CHI, UIST, CSCW
POPL, PLDI

Table 2: Some clusters from our computation

We use this clustering information in our two stage classification system. Unfortunately, there is no obvious improvement in the 10-cross validation. We also change the feature selection process to a two-stage setting. Each cluster has its own feature selection step. But still, we did not see obvious improvement. It is possible that the clusters that we compute are still too many. A better clustering scheme would help.

On the other hand, we tried use the clusters instead of the venues as labels. We did feature selection before the learning process. The average accuracy over 10 run is 48.7%. This is promising, considering that we have 275 clusters. When looking at the data, we noticed that venues like Mobile-HCI, Ubicomp, GROUP, NordiCHI are all have their own clusters. This shows that there is still a lot of space for improving the clusters.

Pairwise Binary Classification

Another approach tested was binary classification of pairs of documents as either "same" or "different," where sameness is determined by membership to a class of example documents. A perfect binary classifier would make for an effective multiclass classifier, as well, and we hope that an *im*perfect binary classifier might still be somewhat useful.

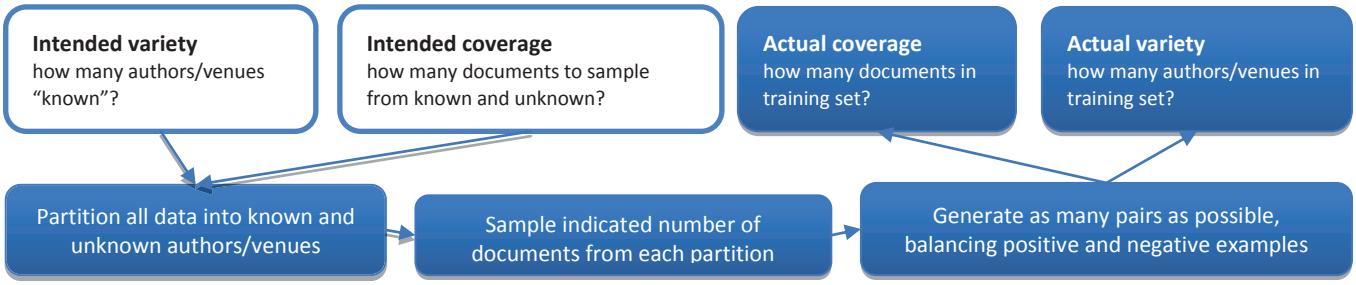


Figure 5: Sampling pipeline: parameters *influence, but do not determine,* the actual variety and coverage of the dataset.

We tested the classifier with a range of class *variety* and *coverage*. Here, variety refers to the number of different classes (out of all possible) exposed in the training, and coverage to the number of documents used for training (Figure 5). Ideally, the classifier would perform well with low coverage and/or low variety, so it would be just as disheartening to see low accuracy as it would be to see a dramatic increase in accuracy accompanying an increase in either coverage or variety.

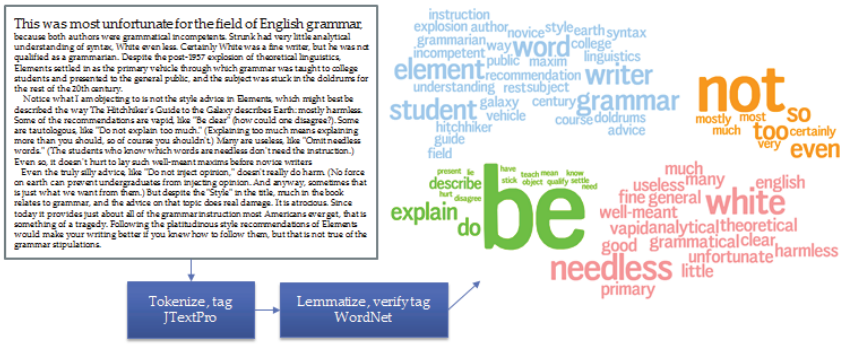


Figure 6: A sample cloud extracted using JTextPro and WordNet. The result is actually four clouds, including lemmatized nouns (top left, blue), verbs (bottom left, green), adjectives (bottom right, pink), and adverbs (top right, orange). Excerpt from *chronicle.com/article/50-Years-of-Stupid-Grammar/25497*, visualized using *wordle.net*

We developed a Java application for polling the database for data, using libraries for feature extraction, and training Naïve Bayes classifiers with Weka [2]. JTextPro [8] was used for tokenization and tagging, and WordNet [1] for lemmatization and verifying part-of-speech: a word identified as a noun by JTextPro but which is not known to be a noun by WordNet is not considered a noun. We also created word "clouds," part-of-speech-specific lemma-count maps (Figure 6).

Given two documents, find **difference between**…

average number of words per sentence	
quotes: '\'' '`'	ratios of special characters (indicated to the left) to the rest of the characters across all words.
periods: '.'	
questions: '?'	
exclams: '!'	
commas: ','	
semis: ';'	
colons: ':'	
parens: '(' ')'	
dashes: '-'	
average number of characters per word	
ratios of non-alpha characters across all words	

Table 3: 12 features based on low-level counts

Given two documents, find…

noun	difference between ratio of nouns/verbs/adjectives/adverbs per all detected words in each document
verbs	
adj.	
adv.	
difference between the ratio of the total number of words found after segmentation and the number of words accepted by the cloud	
difference between ratios of "be" to all other verbs in the verb cloud for each document	
difference between ratio of adjectives and adverbs to all other words in the word clouds	
noun	a measure of how much cloud distributions for the two documents overlap for noun/verb/adjective/adverb clouds.
verbs	
adj.	
adv.	

Table 4: 11 Word cloud-based features

A total of 43 features were extracted: 12 for character and word counts (Tab. 3), 11 for part-of-speech clouds (Tab. 4), and 20 for auto-generated synonym sets (functionally, clouds). The many features specific to parts of speech were motivated by the results of baseline tests, which indicated part of speech counts as most effective for binary classification. The synonym set features, and one of the word-cloud-based features, were based on *cloud overlap*: giving a pair of documents a score corresponding to how different two cloud distributions are. Overall word clouds can be interpreted as a measure of topic similarity, whereas synonym set features sought to more directly address "style" similarity.

Synonym sets were constructed from an aggregate word cloud sampled from the training distribution. Then, a synonym set was generated for each word, using WordNet. Each set was modeled as a cloud for a particular part-of-speech. Of thousands of synonym sets, 20 were chosen as grounds for features on the basis of (1) having some minimal number of occurrences (e.g., 10 across 150 documents), (2) being the most balanced, and (3) avoiding duplicate synonym sets. Here, balanced means that the distribution of occurrences of individual words in a set is roughly uniform. This heuristic was used to avoid extremely large synonym sets that were associated with extremely common words. For example, one of the experiments on the ACM data got *"agree check fit match correspond hold accord harmonize"* as one of its 20 synonym sets, and another experiment on the Gutenberg data got *"finish stop end terminate cease."*

To mitigate the large number of features that could hurt rather than help learning, hill-climbing for feature pruning was also implemented. For each experiment, a development set was chosen from the training set (30% training data were designated development data). Half of the experiments trained a Naïve Bayes model repeatedly removing features as long as accuracy increased on the development set, and half did not do any hill-climbing. Then, all final models were tested against the test

data. At each stage of hill climbing, features could be removed one by one, *or* all of the synonym set features could be removed or added simultaneously, to allow ignoring bad synonym sets.

We ran 44 experiments, with ACM and with Gutenberg data, with and without hill-climbing. We varied intended coverage and. variety parameters for each run. Test set data was sampled from authors and venues that had not appeared anywhere in the training data; the development set was from the same distribution as the training data. Therefore, any success on the part of this binary classifier can be taken to indicate successful learning of the underlying concept of "authorship" or "publication venue" rather than individual variation between document sets. The *lack* of great success that we actually observed is more difficult to reason about, as it can mean either that learning was ineffective due to algorithm or feature choices, or that the underlying concepts were not sufficiently disparate to learn in the first place. The results are presented in Table 5.

Most models tended to skew errors toward one of the classes rather than the other (though all classified data as positive and negative roughly in equal parts). Coverage and variety of experiments spanned a large space, albeit sparsely, and did not seem to lead to an obvious effect on test set accuracy. There is not enough data to conclude the lack of such an effect in general, but it is promising: if there is *no* such effect, then the low accuracy (60%) coupled with a skew toward one class rather than another can still be exploited at an application level.

Hill-climbing to prune features did not make matters better or worse consistently, though it is possible that there is some coverage/variety combination that makes hill-climbing especially prone to finding a feature set that leads to overfitting. Development set results were clearly better in the case of hill-climbing, as we would expect, but the relatively similar accuracy between hill-climbing and no-hill-climbing test results implies that hill-climbing, if anything, makes it more difficult to use development set results to reason about test set results.

	ACM									**Gutenberg**								
	Training Set Information			Dev Set Results		Test Set Results				Training Set Information			Dev Set Results		Test Set Results			
	variety	coverage	size	error skew	accuracy	F0	F1	error skew	**accuracy**	variety	coverage	size	error skew	accuracy	F0	F1	error skew	**accuracy**
Hill	86	36	91	0.45	0.72	0.50	0.38	0.38	**0.44**	40	30	20	0.00	0.88	0.71	0.78	0.50	**0.75**
	181	79	192	0.71	0.66	0.47	0.61	0.67	**0.55**	46	33	19	NaN	1.00	0.49	0.61	0.65	**0.56**
	185	78	223	0.14	0.69	0.59	0.45	0.34	**0.53**	65	47	28	1.00	0.75	0.46	0.69	0.79	**0.60**
	193	83	238	0.70	0.68	0.47	0.61	0.66	**0.55**	73	53	40	0.33	0.81	0.64	0.71	0.63	**0.68**
	199	48	580	0.51	0.64	0.57	0.56	0.48	**0.57**	85	59	49	0.25	0.81	0.71	0.70	0.38	**0.70**
	200	25	2271	0.32	0.70	0.55	0.44	0.38	**0.50**	115	83	56	0.40	0.79	0.62	0.64	0.53	**0.63**
	299	78	881	0.72	0.65	0.54	0.62	0.61	**0.59**	188	106	322	0.10	0.78	0.65	0.65	0.42	**0.65**
	398	90	1234	0.63	0.64	0.51	0.62	0.64	**0.57**	239	162	189	0.61	0.78	0.58	0.67	0.64	**0.63**
	498	84	2221	0.71	0.59	0.54	0.64	0.66	**0.60**	375	243	321	0.36	0.82	0.69	0.69	0.50	**0.69**
	597	93	4055	0.66	0.62	0.54	0.63	0.64	**0.59**	510	309	608	0.19	0.72	0.64	0.59	0.40	**0.62**
	693	107	3491	0.65	0.60	0.50	0.62	0.66	**0.57**	532	320	544	0.67	0.79	0.49	0.69	0.80	**0.61**
No hill	92	42	100	0.62	0.50	0.54	0.59	0.54	**0.56**	24	19	9	1.00	0.33	0.00	0.72	1.00	**0.56**
	178	81	177	0.33	0.60	0.53	0.55	0.52	**0.54**	49	36	19	1.00	0.86	0.52	0.68	0.75	**0.62**
	185	74	245	0.52	0.58	0.56	0.52	0.45	**0.54**	**61**	44	27	0.67	0.73	0.69	0.72	0.59	**0.71**
	197	65	341	0.56	0.52	0.56	0.63	0.60	**0.60**	84	59	41	0.40	0.71	0.67	0.69	0.50	**0.68**
	198	26	1844	0.25	0.66	0.57	0.47	0.38	**0.53**	94	59	52	0.43	0.68	0.68	0.60	0.30	**0.64**
	199	52	460	0.48	0.60	0.51	0.54	0.52	**0.53**	141	86	96	0.57	0.83	0.63	0.70	0.57	**0.67**
	295	86	674	0.34	0.60	0.54	0.51	0.46	**0.53**	171	114	111	0.14	0.70	0.57	0.66	0.64	**0.62**
	398	88	1491	0.24	0.60	0.57	0.43	0.36	**0.51**	236	153	153	0.33	0.77	0.63	0.62	0.47	**0.62**
	499	94	2222	0.18	0.53	0.62	0.33	0.21	**0.52**	364	223	286	0.48	0.76	0.69	0.70	0.53	**0.70**
	600	103	4150	0.46	0.60	0.60	0.54	0.43	**0.57**	434	275	384	0.15	0.68	0.57	0.56	0.49	**0.57**
	694	106	3570	0.52	0.60	0.54	0.58	0.56	**0.56**	605	366	798	0.50	0.78	0.53	0.69	0.76	**0.62**

Table 5: Results of binary classification on 44 experiments, split by dataset (ACM and Gutenberg) and feature pruning (hill-climbing on or off). Variety (# classes), Coverage (# documents) and size (# data points) are reported for the training set. Accuracy and error skew (number of times error was toward one class rather than the other) are presented for both development and test data, with F-measures for each of the classes presented for the training data.

Application-level Considerations

All of the feature extraction and classification methods can be used to provide actionable suggestions at the application level, as well as measures of relative similarity of a given text to the specified example classes. Multiclass classification with Naïve Bayes can provide confidence levels for the document's membership in the classes, and binary classification can be used to find how many documents in a given class are, and are not, "same" as the given text. Bag of words (and, similarly, word cloud) features can be presented in a "word cloud" visualizations, where words used commonly in the desired class(es) are highlighted in one way, and words used commonly in the undesired class(es) in another. Other features that use ratios or counts can be used to generate verbal suggestions, such as "use more descriptive text with more adjectives and adverbs," or "use sentences with fewer words." Here, suggestions can be presented based on how much they favor desired class(es) over undesired one(s).

Figure 7: Writing by example system UI Prototype

Discussion

Even the limited success of our experiments is promising from the standpoint of an application. These models and approaches are not enough to extrapolate subtle "stylistic" differences between hundreds and thousands of conceptually-defined document classes with any satisfying degree of reliability, but our experiments do show that some differences are learned correctly. On a smaller scale, defined by a human being to be meaningful, they might be enough. Our tests are akin to an evaluation of a clustering algorithm on randomly-generated data: it can certainly be quite insightful, but is ultimately limited in shedding light on the performance or success of the algorithm in a real-world setting. Therefore, further work on this project must encompass evaluation of applications and more realistic scenarios.

Another possible direction for exploration involves modification to the algorithms. While we could define more features, it may be particularly interesting to explore how interactive learning approaches an benefit the system and user alike, either though allowing manipulation and selection of features, or through direct interaction with the algorithms where possible.

Though imitation is the sincerest form of flattery, especially successful examples of writing by example in daily life remain, unfortunately, plagiarism. Making a tool that effectively support writing by example available has implications for teaching, and enforcing, academic integrity policy: given the difficulty of explaining the difference between "similar" and "plagiarized" writing, could giving actionable suggestions actually make some people more apt to err on the "plagiarized" side? On the other hand, could such a tool be used to detect inconsistent "style" for a given author, within and between documents, making plagiarism detection easier? Or to provide suggestions to the author for *not* crossing the "similar/plagiarized" line in the first place? Moreover, can existing plagiarism detection methodologies be effective at supporting a less problematic kind of writing by example? And, lastly, can prior plagiarism detection results be legitimately and effectively used for measuring effectiveness of tools supporting certain writing by example tasks?

References

[1] Fellbaum C. *WordNet: An Electronic Lexical Database*. Cambridge, MA: MIT Press. 1998.

[2] Hall, M., Frank E., Holmes G., Pfahringer B., Reutemann P., Witten I.H. *The WEKA Data Mining Software: An Update.* SIGKDD Explorations, Volume 11, Issue 1. 2009.

[3] Keim, D. a, & Oelke, D. (2007). Literature Fingerprinting: *A New Method for Visual Literary Analysis.* IEEE Symposium on Visual Analytics Science and Technology, 115-122. 2007.

[4] Lim, B. Y., Dey, A. K., & Avrahami, D. *Why and why not explanations improve the intelligibility of context-aware intelligent systems*. CHI 2009

[5] McCallum, A. K. *Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering.* http://www.cs.cmu.edu/~mccallum/bow. 1996.

[6] McCallum, A. K. *MALLET: A Machine Learning for Language Toolkit.* http://mallet.cs.umass.edu. 2002.

[7] Park, T., Lank, E., Poupart, P., & Terry, M. *" Is the Sky Pure Today ?" AwkChecker : An Assistive Tool for Detecting and Correcting Collocation Errors.* UIST 2008.

[8] Phan X.. *JTextPro: A Java-based Text Processing Toolkit.* http://jtextpro.sourceforge.net/. 2006.

APPENDIX 1 – ADJECTIVE AND ADVERB USE IN THIS PAPER

APPENDIX 2 – NOUN AND VERB USE IN THIS PAPER