External and Intrinsic Plagiarism Detection Using Vector Space Models

Markus Muhr, Mario Zechner, Roman Kern
Know-Center
8010 Graz
{mzechner, mmuhr, rkern}@know-center.at

Michael Granitzer
Know-Center Graz
Graz University of Technology
mgranitzer@tugraz.at

Abstract: Plagiarism detection can be divided in external and intrinsic methods. Naive external plagiarism analysis suffers from computationally demanding full nearest neighbor searches within a reference corpus. We present a conceptually simple space partitioning approach to achieve search times sub linear in the number of reference documents, trading precision for speed. We focus on full duplicate searches while achieving acceptable results in the near duplicate case. Intrinsic plagiarism analysis tries to find plagiarized passages within a document without any external knowledge. We use several topic independent stylometric features from which a vector space model for each sentence of a suspicious document is constructed. Plagiarized passages are detected by an outlier analysis relative to the document mean vector. Our system was created for the first PAN competition on plagiarism detection in 2009. The evaluation was performed on the challenge's development and competition corpora for which we report our results.
Keywords: Plagiarism Detection, Nearest Neighbor Search, Stylometry, Outlier Detection

1 Introduction

Plagiarism, defined as the theft of intellectual property (Maurer, Kappe, and Zaka, 2006), has been a problem for centuries not only in academic circles. There exist different forms of plagiarism, ranging from simply copying and pasting original passages to more elaborate paraphrased and translated plagiarism. Anecdotal evidence and studies such as (Sheard et al., 2002) strengthen the suspicion that plagiarism is on the rise, facilitated by new media such as the World Wide Web. Growing information sources ease plagiarism while plagiarism prevention and detection become harder.

To combat these problems the first competition on plagiarism detection was held at the third PAN workshop in 2009 in which we participated. The competition was split into two tasks: external plagiarism detection and intrinsic plagiarism detection. External plagiarism detection deals with the problem of finding plagiarized passages in a suspicious document based on a reference corpus. Intrinsic plagiarism detection does not use external knowledge and tries to identify discrepancies in style within a suspicious document. For both tasks extensive, machine generated training corpora were provided upon which we developed and evaluated our solutions. Our contribution is as follows:

- We present a robust external plagiarism detection method based on indexing by balanced clustering in a high dimensional term vector space with a focus on full and very near duplicate detection.

- We present an intrinsic plagiarism detection method based on a combined set of stylometric features spanning a vector space, using outlier analysis to determine plagiarized passages without a reference corpus or any other external knowledge source.

- We report our results for both problems on the PAN 09 development and competition corpora.

This paper is outlined as follows: in section 2 we present related work. Section 3 describes our system for the external plagiarism task, section 4 gives insight on our intrinsic plagiarism detection system. In section 5 we describe the PAN 09 dataset and report our results for the external and intrinsic plagiarism tasks. We conclude and give an outlook on future work in section 6

2 Related Work

External plagiarism detection relies on a reference corpus composed of documents from which passages might have been plagiarized.

A passage could be made up of paragraphs, a fixed size block of words, a block of sentences and so on. A suspicious document is checked for plagiarism by searching for passages that are duplicates or near duplicates of passages in documents within the reference corpus. An external plagiarism system then reports these findings to a human controller who decides whether the detected passages are plagiarized or not.

A naive solution to this problem is to compare each passage in a suspicious document to every passage of each document in the reference corpus. This is obviously prohibitive. The reference corpus has to be large in order to find as many plagiarized passages as possible. This fact directly translates to very high runtimes when using the naive approach.

External plagiarism detection is similar to textual information retrieval (IR) (Baeza-Yates and Ribeiro-Neto, 1999). Given a set of query terms an IR system returns a ranked set of documents from a corpus that best matches the query terms. The most common structure for answering such queries is an inverted index. An external plagiarism detection system using an inverted index indexes passages of the reference corpus' documents. For each passage in a suspicious document a query is send to the system and the returned ranked list of reference passages is analyzed. Such a system was presented in (Hoad and Zobel, 2003) for finding duplicate or near duplicate documents.

Another method for finding duplicates and near duplicates is based on hashing or fingerprinting. Such methods produce one or more fingerprints that describe the content of a document or passage. A suspicious document's passages are compared to the reference corpus based on their hashes or fingerprints. Duplicate and near duplicate passages are assumed to have similar fingerprints. One of the first systems for plagiarism detection using this schema was presented in (Brin, Davis, and Garcia-Molina, 1995).

External plagiarism detection can also be viewed as nearest neighbor problem in a vector space R^d. Passages are represented as vectors within this vector space. If passages from the reference corpus are "near enough" to a passage of the suspicious document in this vector space, the passage is marked as potentially plagiarized. The dimensions of the vector space are usually defined by features extracted from the passages such as terms (usually called the "bag of words" vector space model). This often yields high dimensional vector spaces. Neighbors are defined either by a distance metric $D : R^d \times R^d \to R$ or by a similarity function $S : R^d \times R^d \to R$. Smaller values for D signal nearness while high values for S signal similarity. Nearest neighbor searches in a vector space with a distance metric can be made sub linear by the use of space partitioning techniques that rely on the triangle inequality guaranteed by a metric. Famous partitioning schemes are the Kd-Tree (Bentley, 1975) or the metric tree (Ciaccia, Patella, and Zezula, 1997). However, these techniques degrade to linear searches for high dimensional vector spaces due to the curse of dimensionality: average inter point distances become more similar. This can be somewhat overcome by assuming that the data indeed lies on a lower dimensional manifold which can be captured by dimensionality reduction. In the reduced space partitioning schemas might be applicable again while the original neighborhoods are preserved by the reduction. Common dimensionality reduction approaches are Principle Component Analysis (M.E., 2003) for linear reduction or Isomap (Tenenbaum, de Silva, and Langford, 2000) for non linear reduction. These methods are generally very costly so other methods for nearest neighbor searches in high dimensional vector spaces have been devised. Locality sensitive hashing (LSH) (Gionis, Indyk, and Motwani, 1999) received a lot of attention in recent years due to its simplicity and theoretical guarantees. LSH is the equivalent of the aforementioned fingerprinting schemes applied to high dimensional vector spaces. The nearest neighbors returned by LSH are only approximate in nature. Another approximate nearest neighbor schema was introduced in (Chierichetti et al., 2007) that uses hierarchical cluster trees for space partitioning.

Intrinsic plagiarism detection only recently received attention from the scientific community. It was first introduced in (Meyer zu Eissen and Stein, 2006) and defined as detecting plagiarized passages in a suspicious document without a reference collection or any other external knowledge. A suspicious document is first decomposed into passages. For each passage a feature vector is constructed. Features are derived from sty-

lometric measures like the average sentence length or the average word length known from the field of authorship analysis. These features have to be topic independent so as to capture the style of an author and not the domain she writes about. Next a difference vector is constructed for each passage that captures the passages deviation from the document mean vector. Meyer zu Eissen and Stein (2006) assume that a ground truth is given, marking passages actually from the author of the suspicious document. A model is then trained based on one-class classification, using the ground truth as the training set. The model is then used to determine which passages are plagiarized. However, it is not clear how the ground truth is derived from a suspicious document when no information about the document is known beforehand.

3 External Plagiarism Detection

We treat external plagiarism detection as a nearest neighbor search in a high dimensional term vector space. This is motivated by the extensive literature that exists for the nearest neighbor search problem as well as its conceptual simplicity. Our system consists of three stages:

- Vectorization of the passages of each document in the reference corpus and partitioning of the reference corpus vector space.

- Vectorization of the passages of a suspicious document and finding each passage's nearest neighbor(s) in the reference corpus vector space. Detection of plagiarism for each suspicious document is based on its nearest neighbor list via similarity thresholding.

- Post processing of the detected plagiarized passages, merging subsequent plagiarized passages to a single block.

3.1 Reference Corpus Vectorization & Partitioning

We adopt the vector space model for textual data as given in (Salton, Wong, and Yang, 1975). Each unique term in the reference corpus is represented as a dimension in the vector space $\mathcal{R}^d$, where d is the number of unique terms. Instead of creating a single vector for a complete document we create vectors for

each sentence in a document as we want to detect plagiarism on a per sentence level.

We use the OpenNLP Framework [1] for tokenization and sentence splitting. For each sentence in every reference corpus document a term frequency vector based on the sentence's lower cased tokens is constructed, excluding stop words based on a stop word list. We did not apply any stemming or lemmatization. The resulting vectors are then normalized to unit length to overcome difference in sentence length. We use the standard cosine similarity to asses the similarity between sentences given by:

$$\text{cosine similarity}(\mathbf{x}, \mathbf{y}) = \frac{\langle \mathbf{x}, \mathbf{y} \rangle}{\|\mathbf{x}\| \|\mathbf{y}\|}$$

were $\mathbf{x}, \mathbf{y} \in \mathcal{R}^d$. The denominator of the above equation can be dropped as all vectors are scaled to unit length.

To achieve sub linear nearest neighbor searches we implement a variation of cluster pruning (Chierichetti et al., 2007). The set of reference corpus sentence vectors is first clustered into l partitions using a balanced online spherical k-means implementation (Zhong, 2005). Balancing is crucial as it provides more equal runtimes when searching for nearest neighbors. The balancing is achieved by introducing a penalty to clusters that have more samples than others during the clustering process. Each sentence vector is associated with a single partition, each partition has a representative centroid vector. Our approach deviates from cluster pruning as presented in (Chierichetti et al., 2007) by associating each sentence vector only with the nearest cluster. Additionally we store a sorted list of similarities for each cluster, holding the similarities between the centroid of the cluster and the sentence vectors associated with that cluster. The resulting structure serves as an index. It allows searching the approximate nearest sentences for a given query sentence.

3.2 Suspicious Document Vectorization & Plagiarism Detection

We vectorize a suspicious document in the same way we vectorize reference corpus documents. For each sentence vector of a suspi-

---

[1]http://opennlp.sourceforge.net/

cious document we search the reference corpus for the k most similar sentences as follows:

- Determine the nearest cluster to the query sentence based on the cosine similarity between the centroids and the query sentence.

- Find the position in the sorted similarity list the query sentence would be inserted at based on its similarity to the cluster centroid. Gather $\frac{k}{2}$ sentence vectors to the left and right of that position in the list.

- Perform the same search in the second nearest cluster returning $\frac{k}{2}$ potential nearest neighbors and merge the two sets of candidate reference corpus sentences.

We justify this procedure as follows: The cluster a vector belongs to is likely to also contain the vector's nearest neighbors. To increase the probability of catching most true nearest neighbors we also use the second nearest cluster. An original sentence in the reference corpus and a full duplicate in a suspicious document will belong to the same cluster as they have the same vector representation. Consequently both vectors have the same similarity with the centroid of that cluster. The search in a cluster's similarity list should thus return the duplicated sentence. This can fail if there are more than k vectors in the cluster having the same similarity with the centroid as the suspicious sentence vector. The outcome is dependent on the quality of the sentence splitter as this type of search for full duplicates relies on correct sentence boundaries. The schema will also work in case a sentence was plagiarized with small modifications, albeit with a much lower probability of finding the correct nearest neighbor. We thus expect our system to work well for detecting full duplicates, acceptable in the case of slightly modified sentences and poorly for highly obfuscated sentences. Figure 1 illustrates the function of the similarity list of a cluster.

With the presented schema we can reduce the search for the nearest neighbors of a sentence from being linear in the number of sentences in the reference corpus to roughly $O(l + k + k/2)$ cosine similarity evaluations per sentence, where l is the number of centroids or clusters, k is the number of vectors

taken from the nearest cluster's similarity list and $\frac{k}{2}$ is the number of vectors taken from the second nearest cluster's similarity list.

A sentence of a suspicious document is marked as plagiarized if its cosine similarity to the most similar candidate sentence from the reference corpus exceeds a threshold α. This parameter allows controlling the sensitivity of the system. The outcome of this stage is a set of sentences from the suspicious document that are plagiarized with high probability. The information about the plagiarized sentences' position in the original documents is also retained.

3.3 Post Processing

The final stage assembles the sentences marked as plagiarized in the second stage to continuous blocks. This is accomplished by simply checking whether sentences marked as plagiarized are in sequence in the suspicious document. If this is the case they are merged. This is repeated until no more merging is possible. To further increase the recall, we compare a plagiarized sentence's left and right neighbor sentences to the neighbors of the original sentence. These might have been missed in the nearest neighbor search and have now a chance to be detected. The neighborhood sentences are again compared via the cosine similarity and marked as plagiarized if the similarity to an original sentence is above some threshold β.

4 Intrinsic Plagiarism Detection

Our intrinsic plagiarism detection system is based on the ideas presented in (Meyer zu Eissen and Stein, 2006) and (Grieve, 2007). Meyer zu Eissen and Stein were the first to define the problem of intrinsic plagiarism detection: determine whether passages in a suspicious document are plagiarized based only on changes in style within the document. An author's style is also of importance in the field of authorship classification. Both problems rely on so called stylometric features. These features should be topic and genre independent and reflect an author's style of writing. Changes of style within a document can be detected by various methods. We choose a simple outlier detection scheme based on a vector space spanned by various stylometric features. The system is composed of 3 stages:

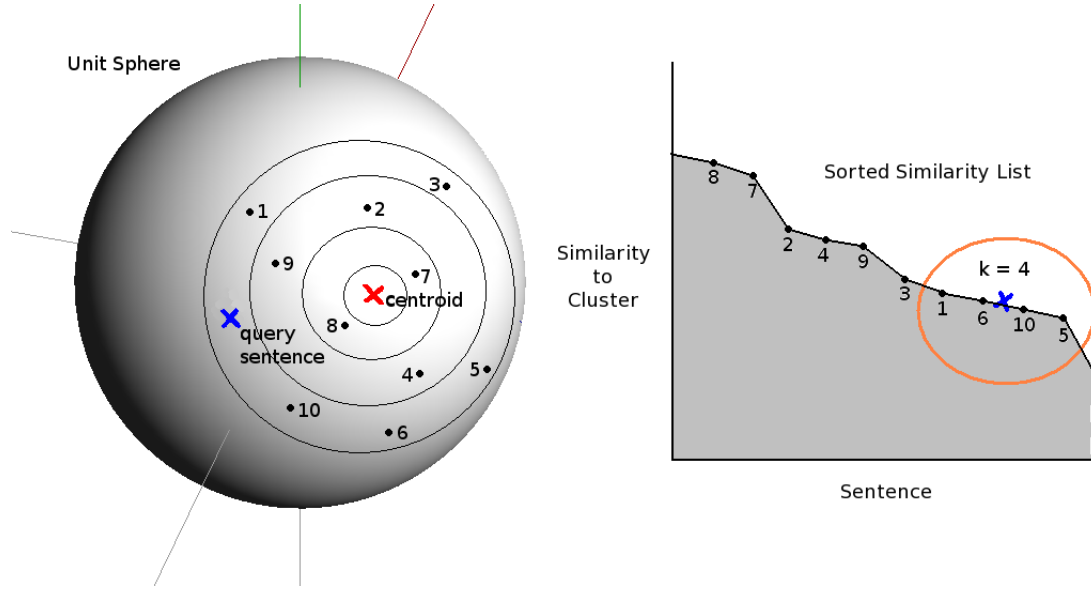- Vectorization of each sentence in the suspicious document.

Figure 1: A cluster with its centroid, ten associated sentence vectors as well as a query sentence vector. The similarity sorting induces concentric rings of near equal similarity around the centroid. The search of the query sentence in the sorted similarity list with $k = 4$ is illustrated to the left.

- Determination of outlier sentences based on the document's mean vector.

- Post processing of the detected outlier sentences.

4.1 Suspicious Document Vectorization

Plagiarism is determined on a per sentence level in our intrinsic plagiarism detection system. However, we chose to use a window of k sentences around a given suspicious sentence. The window is composed of $\frac{k}{2}$ sentences to the left and right of the sentence. We adopt various stylometric features as presented in (Meyer zu Eissen and Stein, 2006) and (Grieve, 2007) that together form a single vector space:

- **Average word frequency class:** Each token w in a sentence window is assigned to an average word frequency class. The class is calculated by $\lfloor \log(freq_{w*}/freq_w) \rfloor$, where $freq_{w*}$ is the absolute number of occurances of the most frequent word in a huge corpus and $freq_w$ is the number of occurances of the token in that same corpus. We derrived our frequency table from tokenizing the English Wikipedia, resulting in aproximately 6 million unique terms. Each average word frequency class is represented by a single dimension in the final vector space. The values in these dimension specify the number of tokens belonging to that class.

- **Punctuation:** For each sentence window the number of occurances of a certain punctuation character is measured. Each punctuation character is represented by a single dimension in the final vector space. The values in these dimensions reflect the frequencies of punctuation characters in the sentence window. [2]

- **Part of speech tags:** Each token w in a sentence window is assigned a part of speech tag from the Penn Treebank part of speech tag set. Each part of speech tag is represented by a dimension in the final vector space. The values in these dimensions reflect the frequencies of part of speech tags in the sentence window.

- **Pronouns:** For each sentence window the number of occurances of a certain pronoun is measured. Each pronoun is represented as a single dimension in the final vector space. The values reflect the

---

[2]We used the following punctuation characters in our experiments: .,?!:;()-
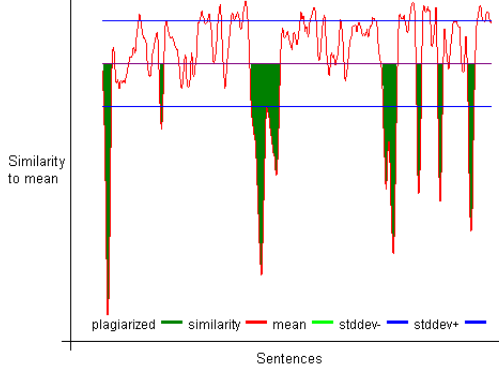
Figure 2: Similarity curve for a document. The x-axis shows the sentence as they appear in the document, the y-axis shows the similarity of the sentence with the document's mean vector. Filled areas indicate the location of actually plagiarized sentences.

frequencies with which each pronoun occured in a sentence window. [3]

- **Closed class words:** For each sentence window the number of occurances of a certain stop word is measured. Each stop word is represented by a single dimension in the final vector space. The values reflect the frequencies of stop words in a sentence window. We used the stop word list from the Snowball stemming framework [4].

For each sentence we construct a vector for each stylometric feature space based on the sentence's window. Each vector is normalized to unit length. The vectors of a sentence are then combined to a single vector by concatenation. The resulting vector is again normalized. Based on the final vectors of all sentences we construct a document mean vector.

4.2 Outlier Detection

The outlier detection tries to determine which sentences deviate from the document's mean. We use a simple detection scheme: we measure the cosine similarity from the mean vector to each sentence vector. We smooth

the list of similarities by an average window smoothing procedure, where the size of the window l is a parameter. We determine the mean cosine similarity as well as the standard deviation from this list of similarities as:

$$mean = \frac{1}{n} \sum_{i=1}^{n} cos(\mathbf{v_i}, \mathbf{m})$$

$$stddev = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (cos(\mathbf{v_i}, \mathbf{m}) - mean)^2}$$

Where n is the number of sentences, $\mathbf{v_i}$ is the ith sentence vector in the document, $\mathbf{m}$ is the document mean vector and cos is the cosine similarity. The jth sentence is marked as an outlier if the following inequality holds:

$$cos(\mathbf{v_j}, \mathbf{m}) < mean - \epsilon * stddev$$

where ϵ is some small constant ≥ 1, and $\mathbf{v_j}$ is the sentence's vector. Marked sentences form the input to the last stage of the system. Figure 2 presents the similarity curve obtained for the suspicious document 5 in the development corpus of the PAN 09 challenge after smoothing.

4.3 Post Processing

Based on the sentences that deviate to much from the mean we derive the final blocks of plagiarized passages. As in the case of external plagiarism detection we simply merge all sentences marked that are neighbors until no further merging is possible.

5 Evaluation

We evaluated our system on the development corpora of the PAN 09 plagiarism detection competition. We describe description of the dataset as well as precision, recall, granularity and f1-measure results achieved by our system for various parameter settings. The measures are defined as follows:

$$precision = \frac{1}{|S|} \sum_{i=1}^{|S|} \frac{\#\text{detected chars of } s_i}{|s_i|}$$

$$recall = \frac{1}{|R|} \sum_{i=1}^{|R|} \frac{\#\text{plagiarized chars of } r_i}{|r_i|}$$

---

[3] We used the following pronouns: i, you, he, she, it, we, you, they, me, you, him, her, it, us, you, them, myself, yourself, himself, herself, itself, ourselves, yourselves, themselves, mine, yours, his, hers, its, ours, yours, theirs, my, your, his, her, its, our, your, their.

[4] http://snowball.tartarus.org/

$$f1 = \frac{2 * precision * recall}{precision + recall}$$

$$granularity = log_2(1 + \frac{1}{|S_R|} \sum_{i=1}^{|S_R|} \#\text{detections of } s_i \text{ in } R)$$

were S is the set of detected passages and s_i is a single detected passage, R is the set of real plagiarized passages and r_i is a single plagiarized passage and S_R is the set of real plagiarized passages for which at least one detection exists. $|S|$, $|R|$ and $|S_R|$ denotes the number of passages in a set, $|s_i|$ and $|r_i|$ denote the number of characters in a passage.

5.1 PAN 09 Development Corpora

For each of the two tasks of the PAN 09 competition a development corpus was available.

The external plagiarism detection dataset consisted of 7214 suspicious documents and as many reference documents. Artificial plagiarized passages were added to the suspicious documents via an artificial plagiarist. For each document the plagiarist decided whether or not to plagiarize, from which reference documents to plagiarize, how many passages to plagiarize, which type of plagiarism to use for each passage and how long each passage would be. The type of plagiarism could either be obfuscated or translated plagiarism. Obfuscation was achieved by shuffling and deleting words, inserting words from an external source and replacing words with synonyms, antonyms, hypernyms or hyponyms. Obfuscation levels ranged from none to low to high.

For the intrinsic plagiarism detection task a corpus of 3091 suspicious documents was available. Plagiarized passages were generated similar to the external task.

5.2 External Plagiarism Detection Results

We performed a parameter study, evaluating precision, recall and f1-measure, for 500 randomly drawn suspicious documents from the external plagiarism development corpus. We studies the effect of the following parameters:

- l, the number of clusters for the index.
- k, the number of candidates taken from the similarity lists.
- α, the threshold above which the similarity between a suspicious and reference sentence has to be so that the suspicious sentence is marked as plagiarized.
- β, the threshold above which the similarity between neighbors of a marked sentence and the neighbors of the original sentence have to be in order to be marked as plagiarized.

Table 5.2 gives the results for various parameter settings. The threshold α was set dynamically depending on the length of a sentence. Very small sentences of five words, which are most likely sentence splitting errors or headlines, are completely ignored. For smaller sentences with fewer than fifteen words, the threshold is set to to 0.7. Sentences with up to thirty five words the threshold is set to 0.5, for longer sentences the threshold is set to 0.4. The threshold β for expanding detected blocks is set to 0.4 with the reasoning that it is very unlikely that nearby sentences will have high similarity values by chance without being actually copied. We arrived at this settings via manual trial and error on a small subset of suspicious documents.

The results show that the influence of the number of centroids l as well as the number of candidates k is marginal. In fact only a considerable change of the parameters to $l = 500$ and $k = 2000$ can affect the recall significantly. However, using such high settings degrades the nearest neighbor search for a sentence according to the previously stated complexity of $O(l + k + \frac{k}{2})$ similarity measurements. We believe that the marginal improvement of the precision and recall due to higher values of l and k do not compensate the much higher runtime. For larger corpora l and k would have to be set even higher, further increasing the runtime. We thus suggest using moderate values for l and k and instead focus on tuning α and β. Especially β is a good candidate to improve the overall recall.

5.3 Intrinsic Plagiarism Detection Results

We evaluated precision, recall and f1-measure for all suspicious documents of the internal plagiarism development corpus for various parameter settings. The parameters of the system are as follows:

- k, the size of the sentence window
- l, the size of the smoothing window by which the similarity list is smoothed.

l - k	Precision	Recall	F1-Measure	Granularity	Recall None	Recall Low
50 - 2	0.9616	0.4045	0.5695	1.9817	0.7044	0.4937
50 - 20	0.9523	0.4119	0.5750	1.9774	0.7053	0.4983
50 - 200	0.9411	0.4210	0.5818	1.9738	0.7053	0.5075
100 - 2	0.9597	0.4101	0.5746	1.9767	0.7044	0.4954
200 - 2	0.9419	0.4132	0.5745	1.9739	0.7050	0.4988
500 - 2000	0.8149	0.4782	0.6027	1.8497	0.7027	0.5534
Competition	0.6051	0.3714	0.4603	2.4424	-	-

Table 1: Results for the extrinsic plagiarism detection system on a split of the development corpus. The split contains 500 plagiarized documents plus the original documents from which the plagiarized passages were taken. The last row shows the results on the competition corpus. The postprocessing was always the same. The first column contains the number of centroids l and the number of evaluated neighbors k in the similarity list. Each row holds the precision, recall, f1-measure and granularity for all obfuscation levels. The column *Recall None* presents the recall on none obfuscated plagiarized passages and *Recall Low* contains the recall on none and low obfuscated plagiarized passages. The discrepancies between the measure on the development corpus split and the competition corpus are due to β being to low for the competition corpus.

Feature Space (k-l)	Precision	Recall	F1-Measure	Granularity
Word Freq. Class (6-3)	0.2215	0.0934	0.1314	-
Punctuation (12-9)	0.1675	0.1908	0.1784	-
Part of Speech Tags (6-6)	0.1797	0.1791	0.1794	-
Pronouns (12-9)	0.1370	0.3587	0.1983	-
Closed Class Words (12-9)	0.1192	0.1467	0.1316	-
Combined Feature Space (12-6)	**0.1827**	**0.2637**	**0.2159**	-
Competition Corpus	0.1968	0.2724	0.2286	1.2942

Table 2: Results for the intrinsic plagiarism detection system on the development corpus. Each cell holds the precision, recall and f1-measure for a given feature space and parameter setting. We only present the top performing parameter settings due to space constraints. We did not evaluate the granularity on the development corpus.

- ϵ, the constant by which the standard deviation of the similarity list is multiplied

Table 5.3 gives the results for various feature spaces and settings for the parameters k and l. Parameter ϵ was fixed at 1.1 for all the experiments. We varied k from 6 to 12 in steps of 2 and l from 3 to 12 in steps of 3. The ranges arose from preliminary experiments where they gave the best results.

We performed the experiment for each separate feature space in order to determine which features have high discriminative power. Surprisingly, pronouns performed very well when compared to more sophisticated features like the average word frequency class. It should be noted that pronouns do not fulfill the constraint of genre independence. A play by Shakespeare is likely to contain many more pronouns then a man-

ual. We can also reproduce the results by Grieve (2007) for the punctuation feature which performed acceptable as well.

Based on the results for the separate feature spaces we took the three best performing spaces, part of speech tags, pronouns and punctuation as the basis for the combined vector space. This combined vector space was then again evaluated on the complete development corpus with varying parameters, showing significant improvements in all measures compared to the separate feature spaces. We used the best parameter settings found in this evaluation for the competition corpus.

6   Conclusion and Future Work

We presented our methods and results for the intrinsic and external plagiarism task of the PAN 09 competition. Our systems performed acceptable, taking the 5th out of 13 places in

the external task, the 3rd out of 4 places in the intrinsic task and the 5th overall place place out of 13 in the competition.

We plan on extending our external plagiarism detection system by incorporating term expansion via synonyms, hyponyms and hypernyms in order to cope with highly obfuscated plagiarism. We also plan to use a more sophisticated cluster pruning scheme that is hierarchical in nature, using hebbian learning to construct a topology of the search space to further increase the probability that the true nearest neighbor of a vector can be determined.

For future work for the intrinsic plagiarism problem, we aim at a better outlier detection method. We will also try to analyze and incorporate more stylometric features, combining them with the best performing features found in this competition. Dynamically adapting the parameters k and l for each document as well as for each feature space is also planed.

References

Baeza-Yates, Ricardo and Berthier Ribeiro-Neto. 1999. *Modern Information Retrieval*. Addison Wesley, May.

Bentley, Jon Louis. 1975. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517.

Brin, S., J. Davis, and H. Garcia-Molina. 1995. Copy detection mechanisms for digital documents. In *ACM International Conference on Management of Data (SIGMOD 1995)*.

Chierichetti, Flavio, Alessandro Panconesi, Prabhakar Raghavan, Mauro Sozio, Alessandro Tiberi, and Eli Upfal. 2007. Finding near neighbors through cluster pruning. In *PODS '07: Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 103–112, New York, NY, USA. ACM.

Ciaccia, Paolo, Marco Patella, and Pavel Zezula. 1997. M-tree: An efficient access method for similarity search in metric spaces. In Matthias Jarke, Michael J. Carey, Klaus R. Dittrich, Frederick H. Lochovsky, Pericles Loucopoulos, and Manfred A. Jeusfeld, editors, *VLDB*, pages 426–435. Morgan Kaufmann.

Gionis, Aristides, Piotr Indyk, and Rajeev Motwani. 1999. Similarity search in high dimensions via hashing. In *VLDB '99: Proceedings of the 25th International Conference on Very Large Data Bases*, pages 518–529, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Grieve, Jack. 2007. Quantitative authorship attribution: An evaluation of techniques. *Lit Linguist Computing*, 22(3):251–270, September.

Hoad, Timothy C. and Justin Zobel. 2003. Methods for identifying versioned and plagiarized documents. *J. Am. Soc. Inf. Sci. Technol.*, 54(3):203–215.

Maurer, Hermann, Frank Kappe, and Bilal Zaka. 2006. Plagiarism - a survey. *Journal of Universal Computer Science*, 12(8):1050–1084.

M.E., Timmerman. 2003. Principal component analysis (2nd ed.). i. t. jolliffe. *Journal of the American Statistical Association*, 98:1082–1083, January.

Meyer zu Eissen, Sven and Benno Stein. 2006. Intrinsic plagiarism detection. In Mounia Lalmas, Andy MacFarlane, Stefan M. Rüger, Anastasios Tombros, Theodora Tsikrika, and Alexei Yavlinsky, editors, *ECIR*, volume 3936 of *Lecture Notes in Computer Science*, pages 565–569. Springer.

Salton, G., A. Wong, and C. S. Yang. 1975. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620.

Sheard, Judy, Martin Dick, Selby Markham, Ian Macdonald, and Meaghan Walsh. 2002. Cheating and plagiarism: perceptions and practices of first year it students. *SIGCSE Bull.*, 34(3):183–187.

Tenenbaum, J. B., V. de Silva, and J. C. Langford. 2000. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, December.

Zhong, Shi. 2005. Efficient online spherical k-means clustering. In *Neural Networks, 2005. IJCNN '05. Proceedings. 2005 IEEE International Joint Conference on*, volume 5, pages 3180–3185 vol. 5, July-4 Aug.