# Applying Plagiarism Detection to Engineering Education

Juan José García Adeva, Nicholas L. Carroll, and Rafael A. Calvo
School of Electrical and Information Engineering
University of Sydney, NSW 2006
{jjga,ncarroll,rafa}@ee.usyd.edu.au

*Abstract*— **We describe a novel plagiarism detection system and its integration with an e-portfolio used in first year engineering teaching. The tool addresses an important issue arising from the decreasing barriers to information access. Academics know that information can support valuable learning experiences, but these experiences are diminished when students plagiarise by copying assignments and getting credit for work they have not done. While it is possible for academics to develop project-based activities to make it harder for students to plagiarise work from outside sources, some students will still copy work done by others within the same class, which can be especially difficult to detect within large cohorts. According to student feedback received while assessing an e-portfolio activity, we found that students were also concerned about plagiarism, and that they modify their approaches to learning based on this concern. We developed a plagiarism detection tool called Beagle, which uses an internal method (also known as collusion): whenever a student submits an assignment to the e-portfolio system, it is compared to those previously submitted by other students. Beagle measures the statistical similarity between students' work using text mining methods. When a specific similarity threshold is reached, the work can be flagged as possible plagiarism or the system can automatically warn the student and request that they resubmit their work. In this paper we present the design of the system, a performance evaluation in terms of accuracy and execution time, and a description of its application integration capabilities through web services.**

## I. Introduction

As with all new technological developments, the Internet has brought new challenges and opportunities to the field of Education. The Internet changes students' perceptions on how they access information, how they collaborate, and how they approach learning. These changes are sometimes helpful, leading students towards deeper approaches to learning. However, they may also have negative influences, leading students towards superficial approaches, hence making their tasks easier at the expense of a diminished learning experience.

Recent pedagogical research shows that learning is not simply assimilating knowledge with the help of a more knowledgeable person or computer system, but also jointly constructed by problem-solving with peers [1], [2] through a process of building shared understanding and reflection [3]. The need to support these activities in an online environment, together with recent advancements in web applications, have led to the greater use of tools for online learning.

Regrettably, many students take a superficial approach to learning, undertaking it as a process where they are only required to complete certain tasks. In this situation, the time spent *looking for an answer* is seen as wasted, since it is only an obstacle to completing those tasks. Furthermore, collaboration and reflection are often seen as additional tasks that must be fulfilled to satisfy the assessment, and not as a learning experience.

Many learners see the Internet as their main source of information, they use it as an optimal way of finding the *right* answers for these tasks. They are often unable to discern the reliability of a particular source and in this superficial approach, learners also tend to fulfill the task with ideas, and even literal text, provided by these sources. Furthermore, an increasing number of students do not acknowledge the sources, misleading the teachers into believing the work is their own contribution. This is one of the forms of what universities call academic dishonesty.

Online teaching may have negative side effects, reinforcing these superficial approaches. Teachers have addressed these problems in several ways: *i*) with novel education designs that promote the right approaches to learning by supporting meaningful collaboration and reflective processes and *ii*) with technologies that detect dishonest behaviours, which are symptomatic of inappropriate approaches.

Our approach is novel in that it supports both evidence-based educational design and technological tools that support deep learning approaches.

Blogs are the most current trend in publishing content on the Internet [4]. They are designed so that users can easily maintain a journal with periodic postings, and its concept has attracted the attention of researchers and publishers alike. They can be integrated into educational support tools so that they can be used as an extension of classroom activities. They can support collaboration and reflection. They can also become part of an electronic portfolio, that allows students to show their journals as evidence of their learning experiences. Using a method for determining whether a technology can support the learning outcomes of a course [5], it was found that blogs can easily be adapted to suit pedagogy, and are considered to be a viable technology for supporting reflective learning [6], [7].

Unfortunately, as teachers engaged in online teaching quickly recognise, online supervision and support are too time consuming tasks. In a class of 100 students, a weekly posting per student during a 10 week period becomes an

1-4244-0406-1/06/$20.00 ©2006 IEEE.

insurmountable 1000 documents to read and provide feedback on. In engineering degrees, first year cohorts of several hundred students are the standard. Detecting plagiarism is hard or impossible for these large classes, as the sheer amount of documents to be checked makes the task too expensive. This is recognised by teachers who are hence inclined to not give any assessment weight and remove these activities altogether.

Language technologies such as Information Retrieval (e.g. search engines) or document comparison systems (e.g. services such as Turnitin) are usually employed to detect plagiarism. Teachers can post a portion of the students posting in a search engine (or submit an assignment to the service) and see if other students have posted the same text. Although technically sound, these are not always 1) scalable, since it takes time for the lecturer to act on each posting, 2) part of a more profound educational design that tackles the deeper issue of students' learning, as opposed to just students' dishonesty.

Our approach brings Language Technologies, also called Language Engineering or Text Mining, to evidence-based design of online teaching activities and software tools.

In Section II we describe the different procedures commonly used to detect or deter plagiarism. They include both software tools and instructional design strategies. Although the text mining techniques we describe could be applied to other educational goals, we have opted to focus on plagiarism detection as it is a well-defined field, with plenty of literature and even commercial products that offer partial solutions to which we can compare our approach. In Section III we describe an electronic portfolio system called dotFOLIO and the student's activity that provides the educational context in which the plagiarism detection tool is used. Section IV introduces the statistical similarity analysis methods for plagiarism detection and our tool called Beagle that implements them. This section also describes its underlying architecture and how it can be integrated into an e-learning platform, particularly dotFOLIO. Section VI presents a set of experiments where the system is used in an anonymous cohort of engineering students and analyses the outcomes. Section VII offers some concluding remarks.

## II. Background

A number of research projects have studied the motivations that students have for committing academic dishonesty. The Joint Information Systems Committee (JISC) in the UK published a report [8] with recommendations for producing policies, staff development (on the reasons for plagiarism and how to design assignments that make plagiarism harder to achieve), and selecting appropriate software tools to fight it. The JISC report identified some of the perceived reasons for plagiarism that provide an insight into preventive solutions to this problem:

- Widening participation: more students with a wide spread of backgrounds and interests [9].
- Higher cost of education, accentuated by the reduction in funding, has forced a larger percentage of students to work while they study.
- The growth of the Internet.

On a survey of $50,000$ students, a study by the Center of Academic Integrity [10] showed that $70\%$ of students admit to some cheating during their studies, and close to $25\%$ admitted to serious test cheating in the past year.

Several commercial products and Internet services provide free-text plagiarism detection, as for instance Turnitin [11] and MyDropBox [12]. In these systems, students' assignments are submitted to the remote service where they are compared with document collections that may include other similar assignments, the Internet, books, and journal publications. The strength of these systems is the coverage of the documents they contain. This is evidenced by their strong partnership with content vendors such as Pearson Education and Longman. These products also offer out-of-the-box integration with commercial Learning Management Systems (LMS) and, in some cases, an API for web services, so that other LMS (e.g. open source products) or educational systems can be integrated. The emphasis of these solutions has been on plagiarism detection, and it appears that all the tools have focused in detecting similar copies of assignments submitted to the LMS' dropbox. The industry behind these systems will probably move in this direction, where integration is an essential feature in their offerings.

Anecdotal information shows that there are students and academics who oppose to the use of systems like Turnitin. According to [13], a student at McGill University legally challenged the university partly on the grounds that Turnitin subsequently adds the paper that the university submits to its internal repository of documents, providing an economic benefit to Turnitin without compensation to the student.

Technically, there are several main methods to detecting plagiarism [14], [15]. They include complete document comparison, document fingerprinting or tagging, stylometry, etc. Behind these methods there is always a concept of similarity like occurrences of words, fingerprints or tags, or the similarity between the writing style of an author and the document being analysed.

The statistical text mining techniques discussed later in this work allows us to apply statistical similarity measures that provide us with an indication about the relationship between two documents. Our contribution provides evidence for possible approaches towards transforming some of these systems designed for plagiarism detection into other educational ventures.

The systems mentioned in this section rely on unknown techniques that due to commercial strategies are not disclosed, remaining hidden from the institutions that use them. We believe that a more productive integration requires an understanding about how the system performs internally.

## III. Electronic Portfolios in Engineering Education

A key aspect in education is reflective learning. Reflective learning refers to a deeper understanding of the subject

material by drawing connections between learning experiences through critical thinking. This process of introspection can be supported using an e-learning application called an E-Portfolio. E-Portfolios are viewed as "personal, life-long content-management systems for collecting, reflecting on, selecting, and presenting learning outcomes" [16]. E-Portfolio systems allow learners to archive their learning experiences as artifacts in a repository [17], and draw connections between these experiences through reflection. The artifacts can also be used for personal goals, such as to receive credit for prior learning, or to demonstrate competencies to potential employers [18]. E-Portfolios are frequently used with long-term educational goals, and are said to support life-long learning experiences.

We developed an E-Portfolio system called dotFOLIO, and trialled the system within a first year engineering course at an Australian university. The system was used to replace a paper-based engineering log book exercise that has been a core assessment component since the course was first introduced. The weblog (or blog) application built into dotFOLIO was considered a suitable online replacement of the paper-based log book. The blog was used by students to: *i*) facilitate reflective learning through the practice of maintaining a journal that encourages "active intellectual monitoring and evaluation of [their] own formal learning and professional practice activities, to examine them for new understandings, and to add to the individual's accumulated knowledge and experience" [3], and *ii*) promote meaningful collaboration, and sharing of ideas between students and with a wider audience, by making postings publicly available.

Students were asked to post in their blog entries that contained their reflections on an engineering topic that changed each week. Since the postings were public, students were able to view their peers' submissions, something not possible through the paper-based approach. The e-portfolio system essentially allowed for the possibility of peer-to-peer learning [19]. This type of collaborative experience in learning from peers [20] is valuable because students are able to gauge their responses with the rest of the class, and determine for themselves whether their submissions were inline with the expectations of the assessment. Furthermore, students were able to read the comments written by the tutors for each of their peers' submissions. Students therefore not only learn from feedback provided on their own work, but can learn from the feedback provided to their peers' blog entries that may differ in perspective or opinion from their own.

The students were surveyed for feedback on using dotFOLIO for the log book exercise. The results of the survey were presented in [6], and shows that students were generally receptive towards using a blog for their log book assessment, and making their work publicly available. However, there were a few comments made by students that expressed concern towards their work being plagiarised by other students. To confirm whether or not this was a consensus of concern, a second survey was carried out, this time explicitly asking students to state whether they agreed or disagreed with the statement 'plagiarism is a concern with online learning tools'. The results for this particular survey question is shown in Figure 1.

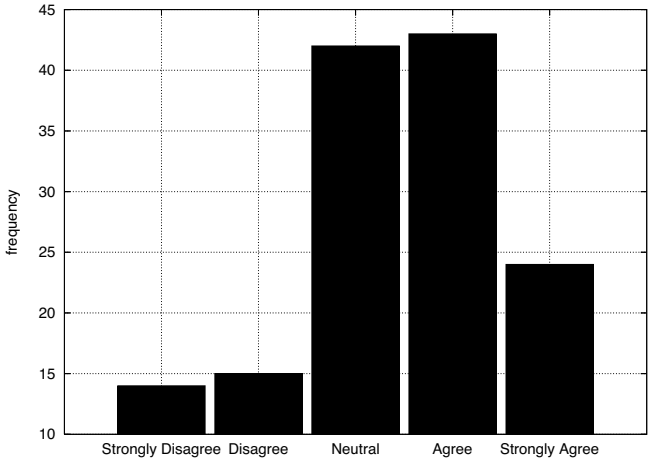



Fig. 1. Students' response to the statement 'plagiarism is a concern with online learning tools'.

Figure 1 conveys a general consensus that even students agree that plagiarism is a concern when using online learning tools. This concern was also shared by the teaching staff, especially after discovering a plagiarism case by chance, from using dotFOLIO's integrated search engine. It was found that plagiarised blog entries could be identified by searching for unique words or phrases using the search engine, then manually reviewing the search results for similar entries. From a set of 2151 blog entries, only 5 plagiarised documents could be found using this method. An exhaustive search of the data set was by no means possible using this cumbersome and highly inefficient method. It therefore raised the question of exactly how many plagiarised documents existed within the data set of blog entries. The resulting challenge was to be able to identify all plagiarised documents within the data set, and to be able to achieve this through a text mining application, so as not to burden the teaching staff with manually performing an exhaustive search of the data set. We describe our solution to this problem in the following section.

## IV. Beagle

Beagle is a document similarity analysis tool used in this work for plagiarism detection. This section covers our analysis of the user requirements, a description of the statistical text mining methods used for the similarity analysis of text documents, and details on the implementation of the tool. We also describe the web service-based interface that allows Beagle to be used by SOAP-enabled applications.

### A. Requirements

The success or failure of applications in general, and online teaching tools in particular, depends on the requirements imposed by the stakeholders. Text document analysis applied to plagiarism detection is no different. Teachers should be

able to use these functionalities without having to worry about issues such as the complexity implicit in a text mining engine, the scalable management of text documents, access control, communication protocols, and so forth.

When working online, teachers normally interact with students through a LMS or some other applications like dotFOLIO. The administrator of these systems should be able to integrate the tool to the system used by the teachers, providing educational benefit (according to the teacher's requirement) but without increasing their workload.

This integration can be achieved through a conventional API, which must be designed to allow new and existing applications to take advantage of the tool. The system administrator or the person in charge of the integration will at least require:

- **Document management**: the tool should provide document persistence, so that users can send documents that are stored and maintained automatically by the tool.
- **Multilingual**: the tool should be capable of analysing text document written in multiple languages.
- **Configurability**: the tool should be flexible in the number of methods and their options used to measure the similarity of text documents.

The main general system requirements for a comprehensive plagiarism detection tool are the following:

- **Open architecture and interfaces**: the system should have an open architecture and open application interfaces to enable interaction and integration seamlessly between the application and the plagiarism detection tool. The architecture should be able to take advantage of the open, dynamic nature of the Internet by supporting rapid integration.
- **Interoperability:** plagiarism functionalities should be easily movable from one host to another without affecting the different applications using them. At the same time, an application that uses the system should be allowed to do so from anywhere, without location initiating additional security concerns.
- **Flexibility:** the plagiarism platform should be loosely coupled, so that an application can easily be integrated to the service without any risk of changes introduced to the system that might affect the client's ability to continue using it.
- **Accessibility:** plagiarism detection and document similarity functionalities provided by the platform should be specified and published in a universal repository for search, discovery, and retrieval.
- **Sustainability:** the platform should be able to grow in the number of features it offers without affecting the application currently using it. It should also last, passing the test of time, and evolving through different technology trends.
- **Scalability:** the performance of the tool should not be influenced by the number of concurrent client applications simultaneously utilising the system.
- **Security**: the tool functionalities should be available for secure use not only within intranets but also on the Internet. Its adoption by a company or institution, should not imply the introduction of any security-related risk, such as opening special ports in the organisation's firewall.

### *B. Approaches to Similarity Analysis*

Beagle uses statistical text mining methods for computing the similarity between text documents. The general approach is based on first building a global representation of all the documents using the traditional Vector Space Model (VSM) [21]. This is an algebraic model that represents natural language documents and queries in a high-dimensional space, where each dimension of the space corresponds to a word in the document collection. According to this model, for each of the documents a feature vector in this space is generated. This feature vector can also be seen as a point in this space. The text documents may have a preliminary removal of both numeric characters or *stop-words*, so that words that provide low information content are not taken into account. Besides, a Porter-type stemming [22] process may be applied to the text in order to remove the commoner morphological and inflectional endings from words. For weighting words in this VSM, we selected the TF/IDF function (term frequency *vs* inverse document frequency). The term frequency in the given document offers a measure of the relevance of the term within a document. The document frequency is a measure of the global relevance of the term.

The TF/IDF function takes a collection of documents $D$ with length $|D|$, so that $D = \{d_1, \ldots, d_{|D|}\}$. Each document $d_j$ is expressed by a collection of terms with length $|d_j|$, such as $d_j = \{t_1, \ldots, t_{|d_j|}\}$. Therefore the TF/IDF value for a particular term $t_i$ in a document $d_j$ of $D$ is given by

$$\mathrm{TF/IDF}(t_i, d_j, D) = \mathrm{TF}(t_i, d) \cdot \log_2 \mathrm{IDF}(t_i, D) = \frac{|t_i|}{\max\{\mathrm{TF}(t_1, d_j), \ldots, \mathrm{TF}(t_{|d_j|}, d_j)\}} \cdot \log_2 \frac{|D|}{|D \supset t_i|}, \quad (1)$$

where $|t_i|$ is the number of times that the term $t_i$ occurs in the document $d_j$ (which is normalised using the maximum term frequency found in $d_j$), $|D|$ is the number of documents in $D$, and $|D \supset t_i|$ the number of documents where $t_i$ appears.

The distance between two documents in the VSM may be used as an indication of how similar their content is. These distance functions are applied to the feature vectors that represent the two documents being compared. The larger the distance the less similar the two documents are.

The rest of this section describes the distance functions selected in this work for similarity measurement. All these distance metrics are well-known in the area of statistical text mining.

Euclidean distance is given by

$$\mathrm{euclidean}(x, y) = \sqrt{\sum_{i=0}^{|S|} \left(\mathrm{w}(x_i) - \mathrm{w}(y_i)\right)^2}, \quad (2)$$

where $|S|$ is the number of dimensions in the VSM (which corresponds to the number of features in each feature vector) and w is the weight of a particular feature in the feature vector.

The Manhattan distance measure results in the sum of the lengths of the projections of the line segment between the points onto the coordinate axes. This function is also known as City Block distance, because it is the shortest distance a car would need to drive in a city organised in square blocks. The measure is defined as

$$\text{manhattan}(x, y) = \sum_{i=0}^{|S|} |\,\text{w}(x_i) - \text{w}(y_i)| \tag{3}$$

The Hamming distance consists of the number of different elements in two strings of equal length. This function is very well known in the field of Information Theory where it represents the number of substitutions (or errors) required to change one string into another. It is expressed by

$$\text{hamming}(x, y) = \sum_{i=1}^{|S|} \delta(x_i - y_i), \tag{4}$$

where the predicate function $\delta$ is defined by

$$\delta(x_i, y_i) = \begin{cases} 1 & \text{if} \quad x_i = y_i \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

The Minkowski distance is a generalisation of the distance between two points in the Euclidean space. This measure is defined by

$$\text{minkowski}(x, y) = \left( \sum_{i=1}^{|S|} |x_i - y_i|^{\lambda} \right)^{\frac{1}{\lambda}}, \tag{6}$$

for $1 \leq \lambda < \infty$. It is interesting to note that when $\lambda = 1$, this function happens to be identical to the Manhattan distance defined in Equation 3. When $\lambda = 2$ it is equivalent to the Euclidean distance detailed in Equation 2. Another special case is when $\lambda = \infty$, which results in

$$\text{minkowski}(x, y) = \max_i |x_i - y_i| \tag{7}$$

A weakness of the standard Euclidean, Manhattan, and Minkowsky distance measures is that if one of the features has a relatively large range, then it can overpower the other features. To prevent this, the data is often normalised. A simple way of normalising the similarity measurement is by previously normalising the feature vectors. This can be expressed by the following equation

$$x_i = \frac{x_i - \mu_i}{\sigma_i}, \tag{8}$$

where each feature $x_i$ in feature vector $x$ is modified by applying its variance and mean. However, feature vectors with very small absolute values will be scaled to have the same variation as feature vectors with initially very large values, which is not desirable in many scenarios. On the other hand, the similarity measure using the Hamming distance does not suffer from these issues and obtaining a normalised measurement is straightforward, being defined by

$$\hat{\text{hamming}}(x, y) = \frac{\text{hamming}(x, y)}{|S|} \tag{9}$$

### C. Implementation

The implementation of the similarity measurement module is based on our own object-oriented text mining application framework called Pimiento [23]. This software component was written using Java Standard Edition (J2SE) and aimed at providing developers with the primary benefits of application frameworks, such as modularity, reusability, extensibility, and inversion of control [24]. Pimiento has an extensible component for each of the text mining domains that it currently tackles: categorisation, language identification, summarisation, clustering, and similarity analysis. It offers numerous features to suit both a production environment where performance is crucial as well as a research context where highly configurable experiments must be executed. It can be used in production systems due to its high scalability based on a cache system that allows for precise control of the amount of memory allocated, and its performance efficiency thanks to a carefully tuned-up code-base. We used the Pimiento's API to generate the documents' VSM framework and the calculation of similarity between document pairs.

Another important implementation aspect of Beagle is regarding the storage infrastructure for the text documents. One of the main challenges that text document repositories face is how to adequately scale to large amounts of information as well as large numbers of users. In most situations, a distributed system will be required to handle the total amount of information in multiple physical storage locations, each one containing many storage objects, and yet treat this set of systems as a single logical unit. This functionality is achieved through our own document repository called Lenteja. Although Lenteja is an independent software component that can be used in other applications and systems, it is one of the key components of Beagle and was developed with the above scenario in mind.

The repository can use two different types of databases to store the unstructured information: a relational database or a document-oriented (i.e. XML) database. If the relational database is chosen, any DBMS with an appropriate JDBC driver can be used, although at this moment we primarily use PostgreSQL – an open source relational database system. In the case of XML databases, we currently support eXist, an open source native XML database. Lenteja can store documents using two different formats: plain text and the OASIS OpenDocument specification. The latter is useful when the input document to be stored comes in a certain format and it has to be imported into a common form while keeping as much of the original format as possible. The current implementation is capable of importing the following document

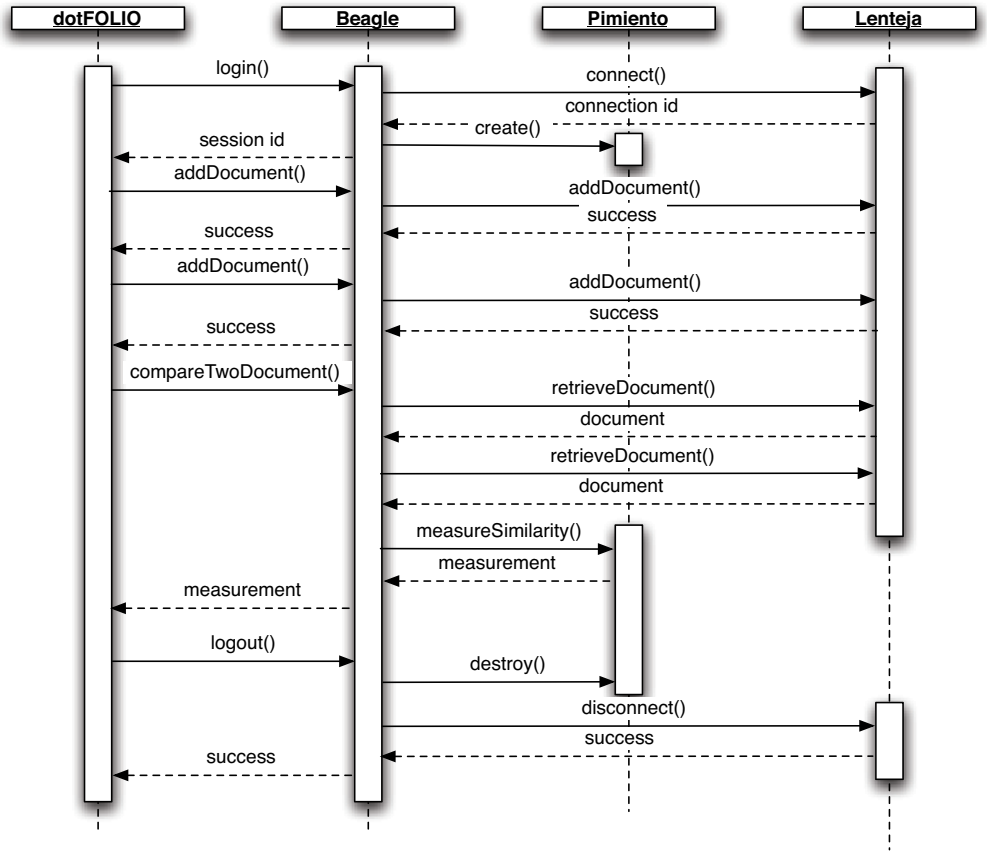



Fig. 2. This sequence diagram shows the iteration among the different software components including dotFOLIO, Beagle, Pimiento, and Lenteja.

types: PDF, RTF, HTML, Microsoft Word, OpenOffice 1.x, and OpenOffice 2.x.

### D. Web Service Interface

Beagle uses a web service to offer its functionality to remote and local applications. Choosing web services as the interoperability mechanism is due to the inherent properties of this technology such as: *i)* its interface contract is platform-independent, *ii)* it can be dynamically located and invoked, and *iii)* is self-contained, so that it maintains its own state. Web services play a major role in the Service-Oriented Architecture (SOA). They are built over well-known platform-independent protocols, including HTTP, XML, UDDI, WSDL, and SOAP. Thanks to these standards, web services are dynamically discoverable and invocable. XML provides a language for platform-independent interface contracts and HTTP provides the interoperable transport mechanism. A notable aspect is that due to web services being self-describing, client applications do not need to know anything about the service except for the format and content of request and response messages. The definition of the message format travels with the message. No external metadata repositories or code generation tools are required. Indeed, the discovery of web services is achieved through standard technologies like UDDI and WSDL.

The web service was developed using Apache Axis, an open source implementation of the W3C SOAP in Java. The Beagle web service interface offers the functionality requirements described in Section IV-A.

- `login`: a new client connects and an unique session identification value is returned. In case of error, `null` is returned instead. This method implies Beagle connecting to the document repository Lenteja and creating a new instance of Pimiento.
- `logout`: a client ends a current session. In case of error, `false` is returned or `true` otherwise. Internally, Beagle disconnects from Lenteja and destroys the instance of Pimiento previously allocated.
- `addDocument`: a current client stores a new document into the document repository. The document is identified by a name chosen by the client. The document content has to be provided as plain text by the client. In case of success, `true` is returned by the web service, or `false` in case of error.
- `deleteDocument`: a current client removes a document from the document repository. The document is identified by its name. In case of success, `true` is returned by the web service, or `false` in case of error.
- `retrieveDocument`: a current client retrieves a document's content from the document repository. The doc-

ument is identified by its name. In case of success, `true` is returned by the web service, or `false` in case of error.

- `compareTwoDocuments`: the similarity of two documents stored in the repository is measured and returned to the client. The two documents are identified by their names. Internally, Beagle uses Pimiento to create the VSM that corresponds to all the documents, so this method can take some additional time when executed for the first time.
- `compareAllDocuments`: one document is compared with all the other documents in the repository. The document is identified by its name. The names of those documents with a similarity above a certain threshold specified by the client are compiled and returned to the client. In case of error, `null` is returned instead. Internally, Beagle uses Pimiento to create the VSM that corresponds to all the documents, so this method can take some additional time when executed for the first time.
- `compareSomeDocuments`: one document is compared with some documents in the repository. The documents are identified by their names. The names of those documents with a similarity above a certain threshold specified by the client are compiled and returned to the client. In case of error, `null` is returned instead. Internally, Beagle uses Pimiento to create the VSM that corresponds to all the documents, so this method can take some additional time when executed for the first time.
- `setLanguage`: a current client specifies the language for the documents. This method will affect posterior similarity measurement when using the comparison methods. The languages supported at this moment are English, Spanish, French, German, and Basque. For other languages, a neutral feature can be used. In case of success, `true` is returned by the web service, or `false` in case of error. After executing this method, Beagle might have to recreate the VSM, hence taking some additional processing time.
- `setProcessing`: a current client sets the configuration for processing documents. This includes whether the words will be stemmed or not, if numerical terms will be taken into account or not, the term weighting method, and the minimum number of terms (as very small documents can easily be very similar to others). In case of success, `true` is returned by the web service, or `false` in case of error. After executing this method, Beagle might have to recreate the VSM, hence taking some additional processing time.

## V. Beagle and dotFOLIO Integration

Figure 2 is a sequence diagram that shows the general behaviour of the system through time as the four main software components involved in this work (i.e. dotFOLIO, Beagle, Pimiento, and Lenteja) exchange messages to control the execution flow.

Beagle is useful for calculating the pairwise similarity measurements for a set of documents. As we will show in the following section, the higher the similarity measurement is, the greater the probability that one of the two documents is a plagiarised work of the other. The similarity measurement alone is not capable of determining the plagiarised document from a pair of similar documents. Further information must be extracted from dotFOLIO so that the system can provide a better recommendation to the teaching staff as to which documents are plagiarised versions of the originals.

For dotFOLIO and Beagle to distinguish between the original document and the plagiarised document from a pair of similar documents, the integrated systems use the time-stamps of the blog entries to determine which entry was posted first. The time-stamps are useful in determining which of the documents from a pair of similar documents is the original, assuming that the document with the earliest time-stamp is the original. Furthermore, dotFOLIO keeps a history of which users have viewed a specific blog entry. This means that we can verify that a plagiarised blog entry was in fact influenced by another blog entry, in the case where the author of the plagiarised blog entry was on record of having viewed the original. By knowing that two documents have been determined as similar by Beagle, we are able to use this result along with the time-stamps and viewing history from dotFOLIO to identify the plagiarised document from the original.

## VI. Experimental Results

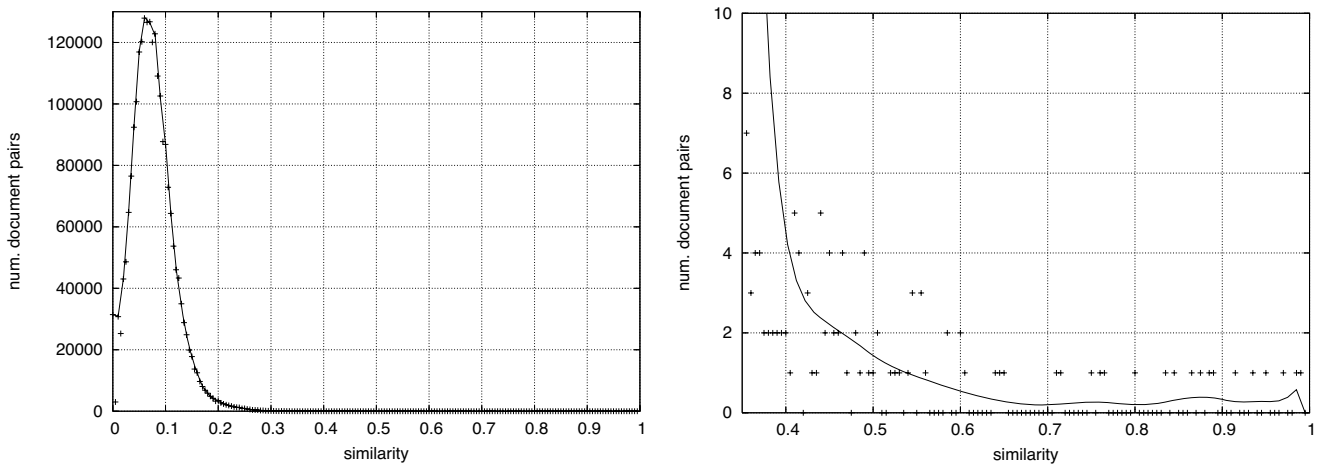
The definition of plagiarism is somehow embedded in the configuration of the similarity threshold. The number of documents that would be flagged as copied would depend on this threshold. If the threshold is too low, it could mean that those somewhat similar postings are actually the result of students' collaboration (a valuable learning outcome) or the fact that the topic requires a very limited vocabulary, so most correct postings would have very similar representations in the Vector Space Model.

If we take the concept of thresholds one step further, we can create categories, each of them defined by the interval or band that lies between two thresholds. An example can be seen in Figure 4. These categories could be application specific descriptions of what the teacher believes is happening in the classroom/online activity. This new point of view supports a fairer diagnostic and might provide better information to the teacher in order to make a prognosis of the different teaching strategies to follow.

In order to estimate how correct Beagle is in detecting possible cases of plagiarism as defined by the teacher, we apply a *precision* measure, represented by $\pi$. It indicates the *positive predictive value* or, in other words, the probability that a document categorised as plagiarised actually was plagiarised. This measure is defined by

$$\pi = \frac{TP}{TP + FP} \tag{10}$$

where $TP$ indicates the number of *true positives*, or how many documents were correctly classified as plagiarised. Accordingly, $FP$ indicates the number of *false positives*.




(a) Overall distribution.

(b) Detailed distribution for similarities over 0.4.

Fig. 3. Distributions of similarity mesurements applied to all possible document pairs.

To calculate the corresponding averaged estimates for $\pi$ there are two different possibilities: *micro-averaging* and *macro-averaging*. Equation 11 defines the micro-average measure $\pi^{\mu}$ as the total sum of true positives over the total sum of true positives plus false positives. Equation 12 defines the macro-average measure $\pi^{M}$ as the sum total of individual precision values over the total number of individual precision values calculated. The macro-average weights equally all the similarity bands found in Table I, regardless of how many findings each contains. The micro-average weights equally all the findings, hence favouring those bands containing more findings.

$$\pi^{\mu} = \frac{TP}{TP + FP} = \frac{\sum_{i=1}^{|R|} TP_i}{\sum_{i=1}^{|R|} (TP_i + FP_i)} \tag{11}$$

$$\pi^{M} = \frac{\sum_{i=1}^{|R|} \pi_i}{|R|}, \tag{12}$$

$R$ is the set of similarity sampling bands and $|R|$ the number of them. For example, in Table I $R = \{[0.4, 0.5), [0.5, 0.6), [0.6, 0.7), [0.7, 0.8), [0.8, 0.9), [0.9, 1)\}$ and $|R| = 6$.

Figure 4 conveys the level of precision for each similarity measurement band using the normalised Hamming distance measure describes by Equation 9. We chose this particular distance measure for providing values between 0 and 1, where a similarity of 0 indicates two dissimilar documents while 1 corresponds to two identical ones. This figure clearly shows that precision increases as the bands of similarity measurements increase. This is further clarified by the increasing micro-averaging and macro-averaging precision values as similarity increases in Table I. The results show that Beagle can be used efficiently to detect plagiarism when a similarity threshold is set at a level so that precision is reasonably high. In the case of Figure 4, a suitable similarity threshold could be set at 0.6, indicating that documents with a similarity measurement greater than 0.6 have a greater chance of being identified by an expert as plagiarised work.

When using a multi-threshold classification, Figure 3(b) could be used to define similarity bands. With the current experimental data, the following similarity bands and corresponding observations were established:

1) *unique* for documents that have less than 0.05 (which could indicate either original or off-topic work),
2) *on topic* for those below 0.2 (the majority of the postings),
3) *collaborative* for those between 0.2 and 0.4,
4) *highly collaborative* for those between 0.5 and 0.6, and
5) *potential plagiarism* for those above 0.6.

This method of classifying relationships between documents could be very valuable in the diagnosis of what is happening in the virtual classroom. Proper evaluations would require time consuming validations across all documents in the collection, which is something that falls out of this research project. If the classification is embedded in a particular system (e.g. dotFOLIO), teachers can validate this classification while supervising the students' contributions.

| Similarity | $TP$ | $FP$ | $\pi$ | $\pi^{\mu}$ | $\pi^{M}$ |
|---|---|---|---|---|---|
| 0.4 to 0.5 | 6 | 35 | 0.15 | 0.15 | 0.15 |
| 0.5 to 0.6 | 8 | 9 | 0.47 | 0.24 | 0.31 |
| 0.6 to 0.7 | 6 | 0 | 1.00 | 0.31 | 0.54 |
| 0.7 to 0.8 | 3 | 2 | 0.60 | 0.33 | 0.55 |
| 0.8 to 0.9 | 6 | 1 | 0.86 | 0.38 | 0.61 |
| 0.9 to 1.0 | 10 | 0 | 1.00 | 0.45 | 0.68 |

TABLE I

TABLE OF PRECISION RESULTS FOR BANDS OF SIMILARITY MEASUREMENTS.

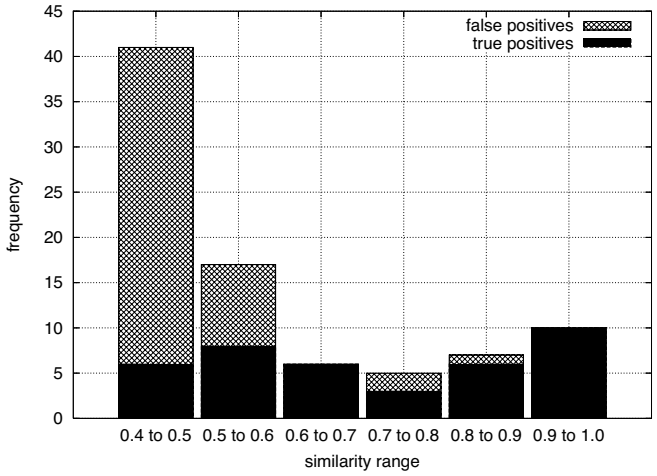



Fig. 4. Frequency of True Positives with respect to False Negatives by similarity range.

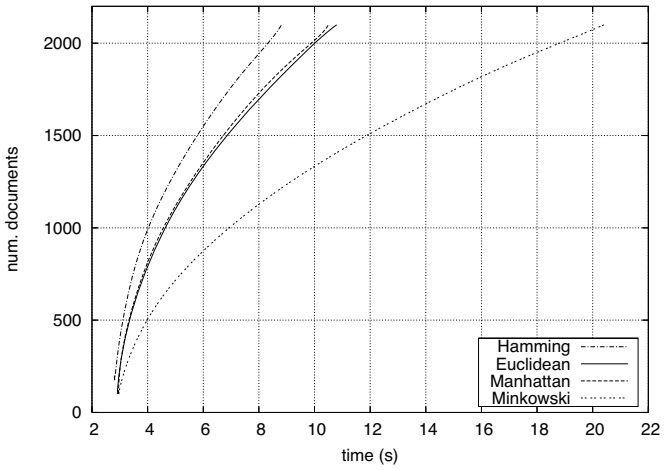



Fig. 5. This graphs shows the execution time that takes to pairwise compare all the documents in a group when applying several similarity measures.

The choice of a distance measure among the several available has an impact on both the precision and the execution performance when analysing document similarities. The execution performance measures indicate the computational requirements that a particular measure impose on a system. This may be important in circumstances where it is crucial to know if the measures and the algorithms used are feasible for a particular real-world application. Since it is not within the scope of this project to manually tag all documents as plagiarised or not, we limited our experiments to only one distance measure, as mentioned above.

Figure 5 shows the time required for computing each of the four existing distance measures on each possible pair of documents in a collection with size indicated in the $y$ axis. For example, a collection of $n$ documents would require $n(n-1)/2$ pairwise comparisons. This measure can help to find out how much time it would take to compare a complete set of postings in a class.

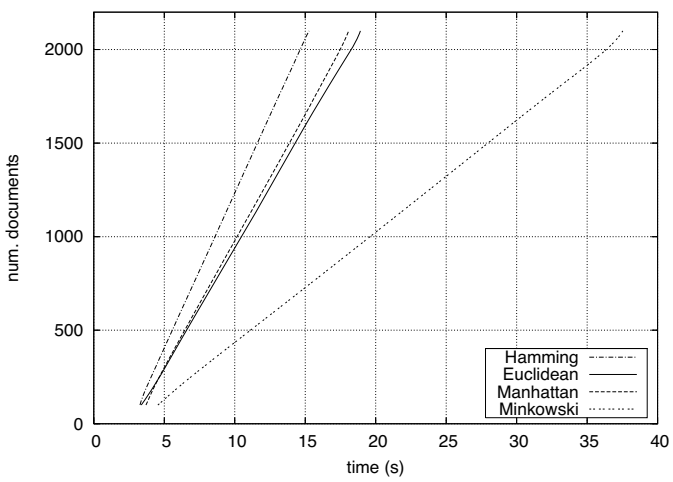



Fig. 6. This graph shows the execution time that takes to compare a growing number of documents with all the existing documents when applying several similarity measures.

Figure 6 shows the time it takes to compare a subset of documents with the complete set that contains all the documents by using the four available distance measures. This comparison would be important in case we need to compare a number of documents with a repository that contains documents from previous years or acquired through other means (e.g. the Internet or encyclopaedias).

## VII. Conclusion

When teachers and students communicate online, they write text that can be efficiently processed using statistical text mining methods. The Vector Space Model provides for a well-known mathematical framework to describe distance-based similarities. This model can be implemented in tools that provide easy integration capabilities.

In this article, we have shown that plagiarism detection is one of the interesting possibilities in which this model can be used. We have evaluated the system on an electronic portfolio application currently being used in a real first year engineering course. The results require the tuning of thresholds for the particular teaching context and group of students, then providing fairly good levels of precision for a relatively large cohort of students. The precision metrics show that the collusion approach is good in detecting possible plagiarism cases. The false positives still exist so the system should only be used as a support tool for academics and not for enforcing disciplinary actions.

The performance of the system have been evaluated, showing that it is feasible to apply this technique to a large number of documents. The execution time on a conventional desktop computer of any of the four available distance measures require less than 20 seconds to process the approximate 2000 postings, which means around 0.01 seconds for each new submission posted.

We have also introduced a novel multi-threshold classification approach that allows teachers to have estimates about the

relationship between documents and produce diagnostic information about the level of collaboration, potential plagiarism and other educational goals.

## REFERENCES


[1] M. Scardamalia and C. Bereiter, "Computer Support for Knowledge-Building Communities," *The Journal of the Learning Sciences*, vol. 3, pp. 265–283, 2004.

[2] D. Laurillard, *Rethinking University Teaching: A conversational framework for the effective use of learning technologies*. London: Routledge Farmer, 2002.

[3] S. Palmer, "Evaluation of an On-Line Reflective Journal in Engineering Education," *Computer Applications In Engineering Education*, vol. 12, no. 4, pp. 209–214, 2004. [Online]. Available: http://www.research.deakin.edu.au/performance/pubs/reports/database/dynamic/output/person/person.php?person_code=palmest

[4] C. Lindahl and E. Blount, "Weblogs: Simplifying Web Publishing," *IEEE Computer*, vol. 36, no. 11, pp. 114–116, November 2003. [Online]. Available: http://csdl2.computer.org/persagen/DLAbsToc.jsp?resourcePath=/dl/mags/co/&toc=comp/mags/co/2003/11/rytoc.xml&DOI=10.1109/MC.2003.1244542

[5] R. A. Ellis and R. R. Moore, "Learning through benchmarking: Developing a relational, prospective approach to benchmarking ICT in learning and teaching," *Higher Education*, no. 51, pp. 351–371, 2006.

[6] N. Carroll and L. Markauskaite, "E-Portfolios and Blogs: Online Tools for Giving Young Engineers a Voice," in *7th International Conference on Information Technology Based Higher Education and Training*. IEEE, July 2006.

[7] S. Downes, "Educational Blogging," *Educause Review*, vol. 39, no. 5, pp. 14–26, September 2004. [Online]. Available: http://www.educause.edu/pub/er/erm04/erm0450.asp?bhcp=1

[8] G. Chester, "Final Report on the JISC Plagiarism Detection Project Final Report on the JISC Electronic Plagiarism Detection Project," Joint Information Systems Committee, Tech. Rep., 2001. [Online]. Available: http://www.jisc.ac.uk/index.cfm?name=plagiarism_detection

[9] B. Leask, "Plagiarism, cultural diversity and metaphor: Implications for academic staff development," *Assessment and Evaluation in Higher Education*, vol. 31, no. 2, pp. 183–199, 2006.

[10] D. M. L. Trevino and K. Butterfield, "Academic Integrity in Honor Code and Non-Honor Code Environments: A Qualitative Investigaton," *Journal of Higher Education*, vol. 70, no. 2, 1999.

[11] Turnitin. [Online]. Available: http://www.turnitin.com/

[12] MyDropBox, "Integration and API." [Online]. Available: http://www.mydropbox.com/services/integration.php

[13] C. J. Neill and G. Shanmuganthan, "A web-enabled plagiarism detection tool," *IEEE IT professional*, no. 5, pp. 19–23, September 2004.

[14] G. R. S. Weir, M. A. Gordon, and G. MacGregor, "Work in Progress – Technology in plagiarism detection and management," in *34th ASEE/IEEE Frontiers in Education Conference*, Savannah, GA, 2004, pp. 18–19.

[15] S. Gruner and N. S., "Tool Support for Plagiarism Detection in Text Documents," in *2005 ACM Symposium on Applied Computing*, 2005, pp. 777–781.

[16] A. Jafari, "The "Sticky" ePortfolio System: Tackling Challenges and Identifying Attributes," *EDUCAUSE Review*, pp. 38–48, 2004. [Online]. Available: http://www.educause.edu/apps/er/erm04/erm0442.asp

[17] H. C. Barrett, "ICT Support for Electronic Portfolios and Alternative Assessment: The State of the Art," in *Proceedings of World Conference on Computers in Education (WCCE2001)*, July 2001. [Online]. Available: http://www.electronicportfolios.org/portfolios/wcce2001.pdf

[18] J. Ittelson, "Building an E-dentity for Each Student," *EDUCAUSE Quarterly Articles*, vol. 24, no. 4, pp. 43–45, 2001. [Online]. Available: http://www.educause.edu/asp/doclib/abstract.asp?ID=EQM0147

[19] P. Jokela, "Peer-to-Peer Learning - an Ultimate Form of e-Learning," in *World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education.*, vol. 2003. AACE, 2003, pp. 1624–1631.

[20] J. Sumner and K. Dewar, "Peer-to-Peer eLearning and the team effect on course completion," in *International Conference on Computers in Education*, vol. 1. IEEE, December 2002, pp. 369–370.

[21] G. Salton, *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Reading, Pennsylvania: Addison-Wesley, 1989.

[22] M. Porter, "An algorithm for suffix stripping," *Program*, vol. 14(3), pp. 130–137, 1980.

[23] J. J. García Adeva and R. A. Calvo, "Mining Text with Pimiento," *IEEE Internet Computing*, vol. 10, no. 4, 2006.

[24] M. Fayad and D. C. Schmidt, "Object-Oriented Application Frameworks," *Communications of the ACM*, vol. 40, no. 10, pp. 32–38, 1997.