

Mining Text with Pimiento

Juan José García Adeva and Rafael A. Calvo

School of Electrical and Information Engineering
University of Sydney, NSW 2006
Australia

{jjga,rafa}@ee.usyd.edu.au

Abstract

Information systems are using an increasing amount of unstructured information in the form of text. Therefore, text mining technologies for information retrieval, filtering, and classification have become increasingly prevalent. This situation has spawned a need to improve the reusability and the ease of integration of these technologies. Different development projects have done this in different ways. In this article we compare some of them and how they can provide textual data mining functionalities to software applications. In particular, we introduce Pimiento, a new Object-Oriented Application Framework (OOAF) for text mining. This framework allows developers to easily create distributed applications that use machine learning and statistical techniques to automatically process documents.

Keywords: text mining, computational linguistics, categorization, clustering, summarization, information extraction, software frameworks.

Introduction

Today, the availability and easy access to information, by both individuals and organizations, is greater than at any other time in history. However, this wealth of information brings important challenges, such as how to effectively sort out what is actually relevant from the overwhelming amount of new information generated every day.

People deal with two main types of information: structured and unstructured. On the one hand, *unstructured information* is usually generated by humans and for humans, although computers require some level of interpretation in order to make unstructured information meaningful. For example, if

a company had a collection of memorandums and wanted to extract those which quoted a quarterly earnings increase of 25% or more, a search engine query looking into the unstructured text would not be enough to find them. On the other hand, there is *structured information*, for which meaning is unequivocally implied by its format. In the earlier example, if the information about earnings had been made explicit as XML or entries in a database, then a relational-type query would make possible to find it.

While there are numerous software tools to capture and process structured information in organizational environments, vast amounts of unstructured information, containing important details about the organization, remain to be explored. The main reason for this situation is the difficulty in analyzing and processing this information. Since the amount of unstructured information in an organization accounts for up to 80% of all its existing information [9], the importance of addressing this challenge is quite clear.

As the importance of unstructured information grows, so does the relevance of textual data mining. Researchers have been developing algorithms for information retrieval (i.e. search engines), information extraction, text categorization, and other specializations of information management since the 1960s. As the research field matures, more commercial offerings become available. The maturity of the field becomes evident when the optimal design for these systems is understood by the stakeholders involved. When this happens, software application frameworks [3] are built. Object-Oriented Application Frameworks (OOAFs) bring together domain-specific design and software components [3].

In this article we look at how an OOAF, improving the reuse of software design and implementa-

tion, can be used to develop text mining tools that application developers can extend and use to build distributed applications. An OOAF is defined as a “reusable design of all or part of a system that is represented by a set of abstract classes and the way their instances interact” [8] and they can also be seen as reusable, semi-complete applications that can be specialized to produce custom-made applications [3].

This article begins by introducing text mining and describing the most common functionalities of text mining frameworks, including Pimiento. We also describe Pimiento’s document repository known as Lenteja, and its interoperability architecture. Using a sample application of plagiarism detection we show how text mining functionalities can be used.

Text Mining

Users want their queries answered accurately, in the right format, the right length, and at the right time. Text mining, sometimes also referred to as intelligent text analysis, text data mining, or knowledge-discovery in text, tackles these issues. Text mining techniques can uncover previously invisible information from existing resources [7], traditionally by applying the principles of computational linguistics. Text mining can be seen as a specialization of data mining. It extracts useful patterns from unstructured text (e.g. books, articles, e-mail messages, web pages, etc.), sometimes converting it first into a structured format, and sometimes processing it without structure in what is called the *bag-of-words* model. Text mining is generally useful in environments where large collections of text documents are handled. One of the well-known premises of using text mining is that the value obtained by mining text documents is directly proportional to the value of those documents. The more important the knowledge contained in the document collection, the more value will be derived.

Text mining frameworks may be classified according to two design criteria: 1) the text mining domain they specialize in (if any) and 2) the design approach to processing data. In this section we describe some of these text mining specializations and provide a comparative summary of some of the frameworks currently available: GATE [2, 1], Weka [10], UIMA [4] and our own framework called Pimiento.

- **Information Extraction (IE):** search engines like

Google can not provide answers to specific questions (e.g. “how many papers has Dr. Who published?”), they can only find documents that are likely to contain the answer. IE tools can extract pieces of information (e.g. the name of a signer, date, or affiliation) thus giving the document some structure. Users then query the relational tables produced allowing for more complex queries. Using UIMA for example, Web-Fountain offers IBM’s customers the possibility to store and analyze massive multi-terabytes of unstructured and semi-structured text. With a more research-oriented history, GATE, a popular open source system built at the University of Sheffield, also offers extensive multilingual extraction techniques. UIMA and GATE are the strongest IE frameworks available to date.

- **Summarization:** users like to receive only as much information about a topic as they have time to read. Summarization techniques distill a source text into a condensed summary. This functionality is usually based on statistical and linguistic methods. Summarization can be extremely useful for helping a user figure out whether a long document is worth reading in full. Only Pimiento seems to offer basic summarization capabilities at this time.
- **Categorization:** with categorization techniques, systems can assign previously unseen documents to the most suitable category available, based on a particular taxonomy (e.g. topic). The categorization system is initially trained with documents whose categories are known. After the machine learning algorithm has been trained, it is able to predict the category of an unseen document. Depending on several factors (e.g. the structure of the collection) the results can be just as accurate as those produced by human categorization. Weka is a research tool with an extensive number of categorization algorithms. Although it does not specialize in text, nor does it have good integration capabilities, it has been used in a number of real-world text categorization applications. Pimiento was originally designed for text categorization [6], hence offering a number of related algorithms and tools.
- **Clustering:** having a taxonomy in which to categorize documents requires sufficient knowledge about the collection of documents. The

goal of clustering is to find structure in this collection by grouping documents into a number of clusters based on their similarity, not on any previous knowledge. Weka provides several clustering algorithms, but again it is not focused on text mining applications. Pimiento and UIMA have some limited clustering capabilities.

The second way of classifying frameworks also applies to other data mining specialties. The text mining software is sometimes *process-centered* and built around the concept of three main data processing activities: preprocessing (including methods such as dimensionality reduction, reading of document formats), solving (training and testing the algorithms) and postprocessing (in the above example, saving the document to the right folder or sending mail to the right person). Although this process-centered approach can be very effective, it poses a problem for users as the methods proliferate. Different preprocessing tools will require different file translation utilities to integrate with each machine learning algorithm implementation. Also, each of these tools requires a different set of utilities to integrate with the postprocessing tools. Packages like Weka [10], used by many researchers in statistical classification, sometimes provide abstract classes and simple mechanisms for extending its functionalities, although they are rarely designed for integrating into real applications. This *data-centered* design is more versatile, with an object-oriented data-model used throughout all the document classification activities.

The design “knowledge” represented in this “improved” way of solving the problem can be extended using software engineering techniques such as the design patterns discussed here. In general, the analysis of design patterns used to solve classification problems, solutions and tradeoffs, is useful for developers of such systems.

As an example of these improvements, we can see that data-centered approaches simplify development since documents can be converted to and from a single representation, reducing the required number of translation utilities from $n(m - 1)$ to $n + m$, where n is the number of input formats, and m is the number of machine learning algorithms. With this approach, incorporating a new interpreter of a document type requires that they are available not only to all the training algorithms but also to all postprocessing mechanisms. New algorithms can be tested more easily and existing ones can be integrated into

applications with less development effort. All four systems described here follow this approach.

General Requirements

We start from the premise that applications should be able to incorporate text mining functionalities without having to worry about the complexity implicit in a text mining engine, the scalable management of text documents, and access control.

Using a text mining platform should not differ from using other functionalities through a conventional API. New applications that want to take advantage of text mining functionalities can easily be designed with the new requirements in mind. Existing applications should be extended to incorporate the new functionalities.

The main general system requirements for a comprehensive text mining platform 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 text mining platform. The architecture should be able to take advantage of the open, dynamic nature of the Internet by supporting rapid integration.
- **Interoperability:** text mining 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 text mining 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:** Text mining 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 text mining platform should not be influenced by the number of concurrent client applications utilizing the system.
- **Security:** the text mining 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.

The Pimiento OOAF

The core of our text mining platform is the object-oriented application framework called Pimiento. This software component was written using Java Standard Edition (J2SE) and aimed at providing developers with the primary benefits of OOAF, such as modularity, reusability, extensibility, and inversion of control [3].

The extensibility of the framework is based on a *black-box* approach [3], where the framework defines interfaces for new components that can be plugged into the framework through object composition. Black-box frameworks are usually structured using object composition and delegation rather than inheritance, as opposed to *white-box* frameworks, which are tightly related to the framework's inheritance hierarchies. In consequence, black-box frameworks are often easier to use and extend than white-box ones [3].

In order to achieve effective reuse, the framework developer must identify those aspects of the target applications that vary from one application to the next, typically called *hot-spots*, and allow explicitly for their variations to be instantiated in applications.

As it can be observed in Figure 1, Pimiento has an extensible component for each of the language technology domains that it tackles: categorization, language identification, summarization, clustering, and similarity analysis. Other hot-spots refer to core tools such as *Language* where the framework manages a stop-word list and a stemmer for each locale (eg. English, Spanish, etc.). Some applications may require more than one of these components.

In some cases, the hot-spot variations are known by the framework developer, so concrete classes can be provided. Application development then becomes a simple matter of selecting the appropriate concrete classes for the application.

Pimiento offers numerous features to suit both a production environment where performance is crucial as well as a research context in which 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.

The framework offer the following functionalities:

- The **text categorization** functionality includes several learning algorithms such as Naïve Bayes (multinomial and complement), Rocchio, Support Vector Machines, and k -Nearest Neighbor. These algorithms can also be used as the base binary learners for ensembles using the decomposition methods One-to-All, Pairwise Coupling, and Error-Correcting Output Codes. The documents to categorize can be written in English, German, French, Spanish, and Basque. A neutral language preprocessor based on n -grams is provided for other languages. Multilingual text categorization is included, so that documents in more than one language can be found in the same corpus. There is also complete evaluation of results including the category-specific measures TP_i , FP_i , FN_i , π_i , ρ_i , F_{1_i} and the averaged measures π^μ , ρ^μ , F_1^μ , π^M , ρ^M , F_1^M , as well as partitioning of the testing space using n -fold cross-validation.
- The **language identification** method we employ is based on computing and comparing language profiles using n -gram frequencies. A n -gram is a chunk of contiguous characters from a word. The number of n -grams that can be obtained from a word of length w is $w + 1$. For example, the word *hello* contains the 3-grams: *_he*, *hel*, *ell*, *llo*, *lo_*, and *o_ _*, with *_* representing a blank. These language profiles are compared to the profile of a document whose language has to be identified, choosing the language that corresponds to the closest. The currently supported languages are English, German, French, Spanish, and Basque.
- The **summarization** functionality uses one of the simplest approaches, based on extracting the most relevant sentences using statistical analysis. This has the advantage of working well with a large number of languages. Other more advanced approaches that create

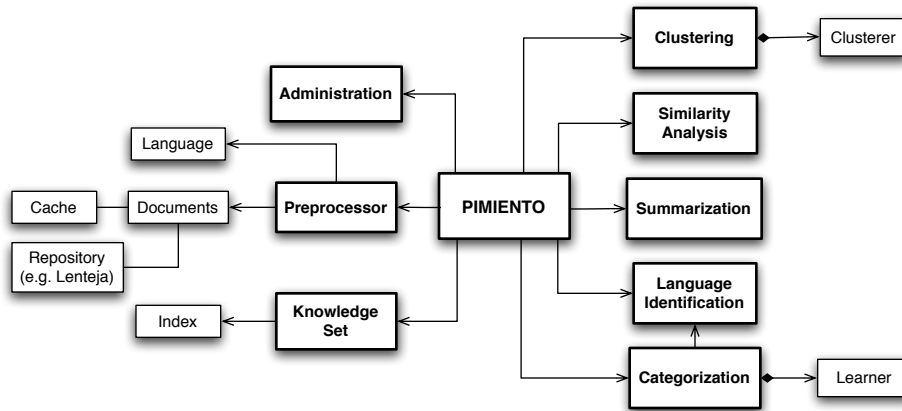


Figure 1: General Class Diagram of Pimiento

new grammatically correct sentences are very challenging to implement in a multilingual environment.

- **Clustering** of documents uses the k -means algorithm, which defines clusters by their centroid. A number of iterations are performed where documents are assigned to their closest cluster. The number of desired clusters k is selected, although there is no guarantee that it will be fulfilled. The distances are measured using similarity functions implemented by the similarity analysis module.
- **Similarity analysis** consists of applying several similarity functions such as Hamming, Euclidean, Manhattan, and Minkowski to measure the degree of resemblance between two text documents. The documents to compare are previously converted into feature vectors using one of the available weighting methods, including term existence, term frequency, and their combination called term frequency/inverse document frequency (TF/IDF).

The Lenteja Document Repository

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.

The storage infrastructure is based on a repository for text documents 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 the text mining platform.

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 (<http://www.postgresql.org>), an open source relational database system. In the case of XML databases, we currently support eXist (<http://exist.sourceforge.net>), an open source native XML database.

Lenteja can store documents using two different formats: 1) plain text and 2) 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 types: PDF, RTF, HTML, Microsoft Word, OpenOffice 1.x, and OpenOffice 2.x.

Application Integration

Figure 2 shows a diagram of the Pimiento interoperability architecture. Applications can interact with Pimiento through a web service layer or, if running on the same hardware, they can interact directly using the framework's API. Users also have access, via a web interface, to the administration functionalities,

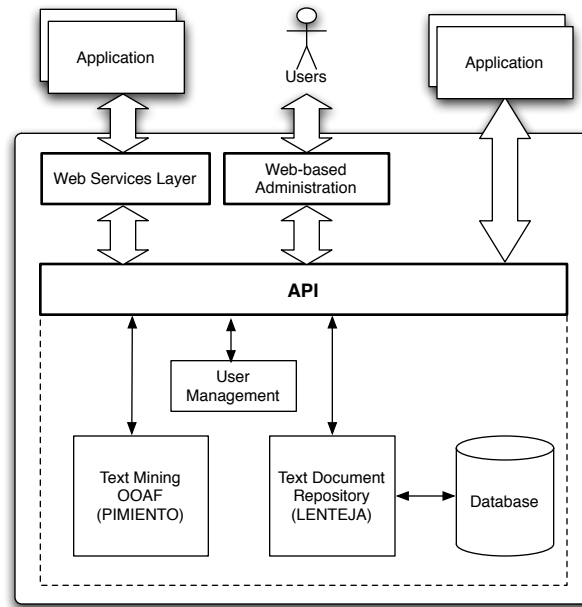


Figure 2: Pimiento Interoperability Architecture

such as adding new user accounts, purging information in the text document repository, viewing the history of present or past tasks, etc. An important strength of this architecture is its simplicity.

Web services encapsulate complexity while allowing the distribution of load and the improvement of scalability, which are common requirements for integrating text mining techniques into real-world applications. A web service has three main properties: *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 a 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, Pimiento's client application does 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.

Plagiarism Detection Sample Application

As an example of how Pimiento can be applied we describe here a plagiarism detection feature added to dotFolio (<http://www.dotfolio.org>) – an e-portfolio system developed at the University of Sydney, Australia.

Broadly speaking, plagiarism is one type of intellectual property violation that consists of using the ideas, words, or findings from other people without acknowledging them. In universities, it is an important type of academic dishonesty, where a student copies material from another person's work and claims it as his own. Exhaustive manual plagiarism detection is time consuming and sometimes impractical.

A plagiarism detection system attempts to automatically find plagiarized material submitted by students. Some plagiarism detection systems compare each document submitted by a student with external repositories such as libraries, endorsements systems, or even the web. The system presented here uses an internal approach (also known as collusion): whenever a student submits an assignment

in the dotFolio system, it is compared to those previously submitted by other students. The system takes advantage of the Similarity Analysis functionality of Pimiento to measure the similarity between two text documents using different functions.

The dotFolio system and the plagiarism detection tool are deployed in different servers and the integration between these two components is based on a web service. The plagiarism detection system uses Pimiento and Lenteja locally. The process is simple: 1) a student submits an assignment, 2) dotFolio sends the document and a default similarity threshold (between 1 for exact copies and 0 for completely different documents) to the plagiarism detection tool, 3) Pimiento measures the similarity with all the existing documents in the repository 4) a list of documents with a similarity below the given threshold is returned to the dotFolio system, 5) based on the teacher's preferences, the dotFolio system decides what to do with probable plagiarism cases and 6) the new document is added to the document repository for future use.

Concluding Remarks

Research progress on text mining algorithms is currently available through software systems that can be integrated into existing infrastructures and document repositories using web services. We have described some popular frameworks, comparing them according to their specialization and architectural design. We also looked at the Pimiento framework in greater detail, describing some of its functionalities, as well as its architecture. The Pimiento framework is currently being integrated to real applications based on a distributed infrastructure that uses web services.

References

- [1] K. Bontcheva, H. Cunningham, V. Tablan, D. Maynard, and H. Saggion. Developing reusable and robust language processing components for information systems using gate. In *Proceedings of the 3rd International Workshop on Natural Language and Information Systems (NLIS'2002)*, New York, 2002. IEEE Computer Society Press.
- [2] H. Cunningham, K. Bontcheva, V. Tablan, and Y. Wilks. Software infrastructure for language resources: A taxonomy of previous work and a requirements analysis. In *Proceedings of the Second Conference on Language Resources Evaluation*, Athens, 2000. European Language Resources Association, Paris.
- [3] Mohamed Fayad and Douglas C. Schmidt. Object-oriented application frameworks. *Communications of the ACM*, 40(10):32–38, 1997.
- [4] D. Ferrucci and Lally A. Building an example application with the unstructured information management architecture. *IBM Systems Journal*, 43(3), 2004.
- [5] J. J. García Adeva and Rafael A. Calvo. A Decomposition Scheme based on Error-Correcting Output Codes for Ensembles of Text Categorisers. In *Third International Conference on Information Technology and Applications*, volume Vol. I, pages 375–378. IEEE Computer Society, 2005.
- [6] Marti A. Hearst. Untangling text data mining. In *Proceedings of the 37th conference on Association for Computational Linguistics*, pages 3–10, College Park, Maryland, 1999. Association for Computational Linguistics.
- [7] Ralph E. Johnson. Frameworks = (components + patterns). *Commun. ACM*, 40(10):39–42, 1997.
- [8] Cathleen Moore. Diving into data. *InfoWorld*, October 25, 2002.
- [9] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques (Second Edition)*. Morgan Kaufmann, June 2005.

Biographies

Juan José García Adeva received his BEng in Computer Engineering from the Mondragón Engineering School at the University of the Basque Country, Spain, and a MSc by research in Computer Science from the University of Essex, UK. He has worked for several years on different topics of Artificial Intelligence at research centers in Spain, the UK, and the US. He is currently working towards his PhD in the School of Electrical and Information Engineering at the University of Sydney, Australia. He is a member of the IEEE.

Rafael A. Calvo is Senior Lecturer, Director of the Web Engineering Group and Associate Dean of eLearning at the University of Sydney's Faculty of Engineering. He holds a PhD in Artificial Intelligence applied to automatic document classification and has taught at several Universities, high schools and professional training institutions. He has worked at Carnegie Mellon University (USA) and Universidad Nacional de Rosario (Argentina), and as an Internet consultant for projects in Australia, Brazil, USA, and Argentina. Rafael is author of a book and over 50 other publications in the field and he theme editor for the Journal of Digital Information.