Working Paper Series
ISSN 1177-777X

Text Categorization and Similarity Analysis: Similarity measure, Literature review

Michael Fowke[1], Annika Hinze[1], Ralf Heese[2]

Working Paper: 11/2013
December 2013

[1]Department of Computer Science
The University of Waikato
Private Bag 3105
Hamilton, New Zealand
[2] Pingar International Ltd.
152 Quay St, Auckland, New Zealand

Text Categorization and Similarity Analysis: Literature review

Michael Fowke[1], Annika Hinze[1], Ralf Heese[2]

[1]University of Waikato, Hamilton, New Zealand
[2]Pingar International Ltd, Auckland, New Zealand

1. Introduction

Figure 1 shows a situation that has become increasingly common. With document organisation like this it is almost impossible to find the file that you want, and definitely not quickly.

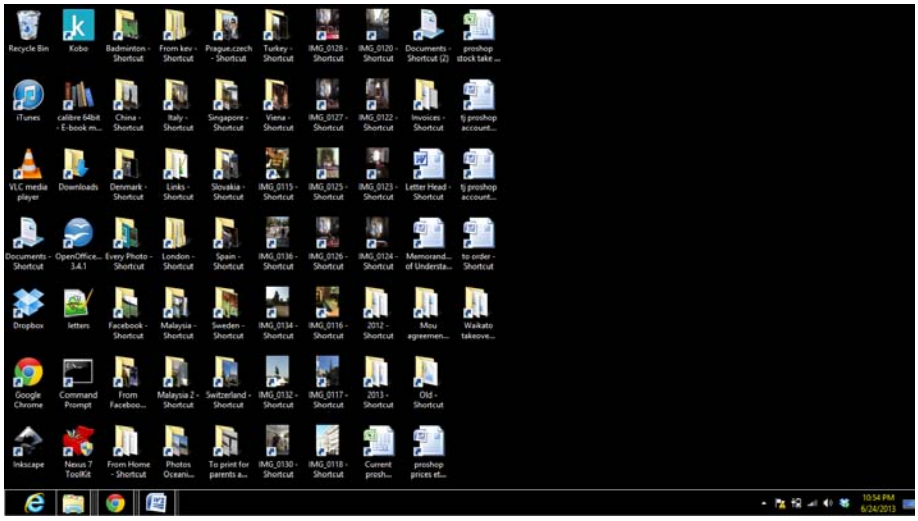


Figure 1: Example of a very crowded desktop

Document classification and provenance has become an important area of computer science as the amount of digital information is growing significantly. Organisations are storing documents on computers rather than in paper form. Software is now required that will show the similarities between documents (i.e. document classification) and to point out duplicates and possibly the history of each document (i.e. provenance). Poor organisation is common and leads to situations like above. There exists a number of software solutions in this area designed to make document organisation as simple as possible. I'm doing my project with Pingar who are a company based in Auckland who aim to help organise the growing amount of unstructured digital data. This reports analyses the existing literature in this area with the aim to determine what already exists and how my project will be different from existing solutions.

Most computer users end up with a number of copies of documents for a variety of reasons. Three main reasons were identified by Karlson et al. [1]:

1. Content preservation: People tend to make copies to back up their work and save different versions as checkpoints. They then rely on the date modified timestamps to find the current version.
2. Sharing across devices: People often end up with many copies when they are sharing documents across devices such as laptop, desktop and tablet.
3. Sharing between people: When documents are shared between multiple users then numerous copies are usually created to keep each contributors version of a document separate.

Figure 2: Document sharing amongst workers (image adopted from www.javla.com)

1.1. Focus of this project

The aim of the project is to implement a tool that will aid users in finding and organising documents. Software like this is required due to the increasing congestion in file systems. I hope to create a system that will make organisation clearer by showing which files are related and also which are versions of another file. This information will make it easier for a user to discover and locate documents relevant to the user's task.

1.2. Approach

I am provided with a document corpus from Pingar which I will test the software on. Pingar are also providing me with two of their semantic technologies: Pingar API for extracting entities from a document collection and a Taxonomy generator for finding relationships between the extracted entities. I can use this information along with the original document text to determine inter-file relationships.

1.3. Structure of this document

This report reviews the existing literature and aims to show the differences between each of the methods and then decide which approach is best for my implementation. I start off by breaking the existing work into two sub groups and show the differences between each group. Each of the approaches within these groups is discussed and applied to a real situation to assess its appropriateness. The overall design of my project is then decided based on my findings from the related literature. The

next stage is to break the software design into smaller pieces and determine what similarity measure will be used to assess document similarity. Finally a conclusion is given on the similarity measure and at this point enough information is known to begin implementation of the software.

2. Background

2.1. Scenario

I will be using a running example throughout this literature review to show how each of the implementations would classify a document. The example involves classifying documents into those that are related to Sir Edmund Hillary and/or climbing. By describing how each solution tackles the problem it should make it clear how they work and identify the parts that work well and those that do not.

2.2. Semantic technology

Semantic technology plays a big part in the project and is the part assisted by Pingar. Semantic technology includes named entity extraction and taxonomy creation. Pingar software will output a named entity summary and taxonomy from the input documents. The taxonomy sorts the content into subcategories and gives the relationship between entities in the document using a tree like structure. Named entities are phrases that contain the names of persons, organizations and locations [2]. In our example Edmund Hillary is an entity and he is a climber and a climber is an explorer which are also both entities. The entities are the objects and the taxonomy gives the semantics or relationships between them (Figure 3).

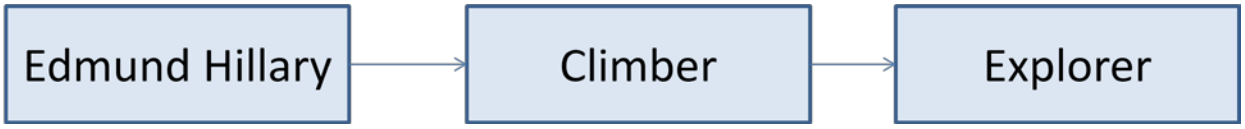
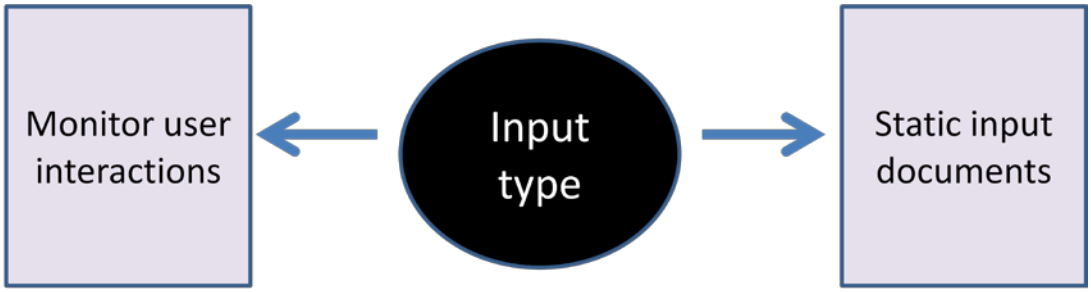


Figure 3: Visualisation of taxonomy example

3. Related work: Methodology

I reviewed the existing solutions and analysed them based on their input, output, approach and measurement used. It became clear that the best way to sort the implementations into groups was to separate on the input available for classification. At a certain point in time a documents history may or may not be known. A file may be viewed for the first time in which case nothing is known about its history (static input). Alternatively the complete history of a file may be known (monitors user

interaction). The classification of the document depends greatly on how much information is known about the history of the document.



Monitoring interactions	Static documents
Aware of actions on documents	No knowledge of actions
Can establish history	Difficult to establish history
Software required on users computer	No software required until classification

Figure 4: Summary of inputs to document classification

Figure 4 shows the properties of each of the two groups of implementations. Essentially one group is watching the user as they produce and interact with a document. The other group is viewing a document for the first time during classification.

4. Monitoring user interactions

Many of the solutions involve monitoring or watching the user as they create the documents. This method can uncover a great deal of information about the history/similarity of documents if the software can observe exactly how documents interact. Provenance is a key word in all the examples that use this approach. Provenance is described as the process of tracing and recording the origins of data and it's movement between databases [3]. The majority of solutions I looked at fall into this group.

2.3. Patent: Tracking document usage

An example of this is the patent by Johnson et al. [4]. This is an application that takes multiple documents as input as well as a tracking module within each document. The application outputs a history and usage of the document. The approach to identifying a documents history and interactions with other documents is simple as the tracking module gives a detailed listing of anytime the document is interacted with in any way. The accesses of a document can be varied and include copying, printing,

attaching the document to an email etc. A useful piece of information stored by the tracking module is an id of each computer that accesses the file. The system can then compile the history of computers that accessed the document and the actions they carried out on it.

This solution doesn't group files by topic as it's working on identifying interactions between files. For our example on Edmund Hillary it will find a document to be related to one on Hillary only if it can determine that it's derived from it.

2.4. Timewarp

A second example is the Timewarp Software [5]. This system is different in that the input is a document with numerous versions created by different users. The output produced by this system is an integrated output document built from the combined input documents. This system uses the approach of storing the input documents as timelines rather than static documents. A timeline contains a lot more information on the timestamps for each part of a document and will help document integration considerably. Again this system relies on having more information in the input than just the static documents. Timewarp generates the input by making the timeline explicit in the interface the user sees when creating the document (see Figure 5). This approach means that the user has to change the way they would work on a document from the regular approach. Rather than maintaining different versions in a folder, Timewarp shows a graph of the different versions over time. It's still simple enough for the user to use as they just need to work on the most recent version as shown by the last node in the directed acyclic graph.

Similar to the tracking document usage patent, it's not looking to classify documents or join documents on topic. In this example it will join pieces of documents on Edmund Hillary together based on the timestamps for when they were modified.

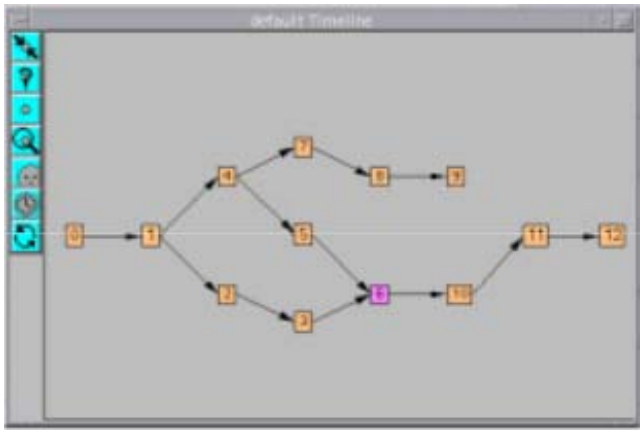


Figure 5: User view when using timewarp, Edwards et al. [5]

2.5. H.P. Trust cloud

Another example of classification being done by observation is that used by HP in their trust cloud [6]. There are a number of issues currently regarding file security in cloud based storage systems and HP have implemented a method to track all file accesses so that cloud users can be confident that their content is only being viewed by themselves. The output of this software is a history of all file accesses. Again the system monitors all file interactions and movements and in combination with a provenance layer it reasons about the origin of resources.

For our example on Hillary, this solution doesn't know anything about the content of the documents. The aim is to track the file accesses and movements and is aware of the locations of a document. It would be able to tell that two documents on Hillary were related only if they were derived from each other.

2.6. Digital thread

Digital thread is a system created by next page [7] for version control and document organisation. The benefit of this system is that the user is free to store documents on their personal computers however they choose and the Digital Thread will still be able to give a comprehensive view of where a document has been and its history. Input documents can be in emails, hard drives, or almost any type of storage. The system uses a digital thread to track a document so it can be easily followed despite name changes and other modifications. It uses patented algorithms to analyse a documents DNA to determine how documents are related.

Like many of the other examples already given, this system isn't concerned with what's within the document. It doesn't even need to know anything about the document format. Digital thread is watching

the file being moved around/ renamed and interacted with from outside the file. It will find documents on Hillary to be related if one is derived from another.

2.7. Copy aware computing Ecosystems

One paper introduces the idea of using metadata in the classification of documents. This includes information on the date of creation, file size, content type, emails attached to etc. This is all information that is associated with a document without actually knowing much about the content in the document. Karlson et al. describe it as a non-keyword approach [1]. This system does observe user interactions as well as use metadata but it would be possible to carry out document classification just based on the metadata if it wasn't practical to observe the user as they interact with the document. The software outputs a version set which is a set of digital items that users conceptualize as a single entity. It also offers graphical representations of the different versions of a document. The software appears to mainly use the metadata that was described above but it does also use information of the users activities in respect to the documents. It aims to find interfile provenance relationships. Provenance involves the ownership of data and the data's usage. Karson et al. state that recording when a person uses the copy/paste command was useful in establishing related files but not useful in determining different versions.

As stated above, this system classifies without knowing about the content of the documents. It works by finding relationships between files using knowledge such as emails a document is attached to. It will find documents on Hillary to be related if they were attached to the same email or share some other forms of connection. It won't simply find two documents with similar topics to be related.

2.8. Tasktracer

Task tracer is a different kind of software to the others mentioned in this section. It involves observing the resources a person is using and then each is assigned to a certain task that the user is working on, Dragunov et al. [8]. This is different in that it isn't finding similarities between documents but it is still watching the documents that a person is using and then assigning them to certain tasks using an algorithm. The application monitors user interactions and outputs a set of tasks, each with a number of associated interactions. This software uses machine learning methods to classify the documents into tasks. Other literature also states that machine learning is the most common method. There is a later report on the task tracer that further explains it's function [9]. This paper mentions that as well as classifying resources to tasks, a task can be automated by making predictions on how a task will be carried out. It again mentions using provenance data in the algorithms and finding most frequent paths

in a graph representing user's actions. The group from Oregon University who created the task tracer bring up the point that it's necessary to find the balance between granularity of data collection and necessary level of inference. This is an issue that would be present in all the methods that involve observing the user as too much data collection would cause the software to run slowly. I have included this software in the group that monitors user interactions as the system is aware of users actions as they happen

This solution isn't looking to find similar documents but rather group resources together that are required for a certain task. Two documents on Hillary would be grouped together if it observes that both are used when a user is working on a single task.

2.9. Document DNA

The final example of a system that monitors user interactions is one by Michael Rinck who is a PHD student at Waikato University. His software takes multiple documents as input with each having attached DNA which accumulates as a person interacts with the document. The output of his software is a summary of document connections and similarities [10]. The software observes a document being created and the commands that are applied to it. Each document has a DNA similar to that of a living organism and each command will add information to the DNA (Figure 6). Rinck uses a method to determine how far apart two documents are based on the number of differences in their DNA. The report states that the "what you see is what you get" approach to documents and printing was limiting advancements and that generating additional information on a document is providing greater opportunities for document tracking and understanding. Rinck states that a document is only monitored in a few certain programs [11]. These include Microsoft word and a few other Microsoft programs as these were the most common and also easiest to track user interactions.

Like the other methods in this group, this doesn't look specifically at the topic of a document but rather interactions and in particular inter-document relationships. It will find two documents on Edmund Hillary to be related if they have content that comes from the same source. Whereas some of the methods would only find connections if one document is a copy of another, this will find connections if one contains a segment copied from the other.

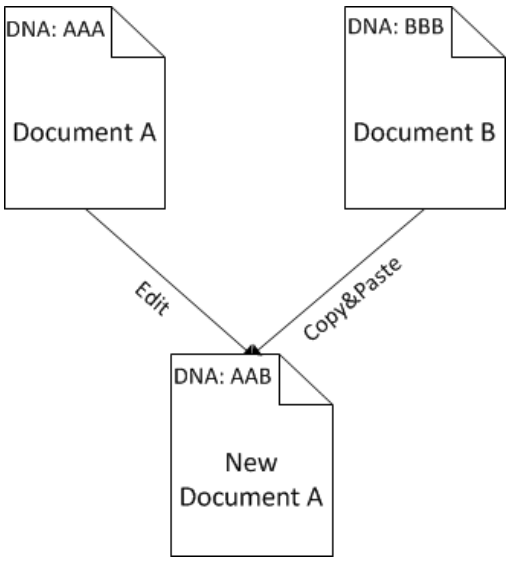


Figure 6: Attaching DNA to a document, Rinck [11]

2.10. Summary of methods that observe user interactions

The following table summarizes the input, output and method of existing implementations for this group.

Approach	Input	Output	Method
Tracking patent	Set of documents each with a tracking module	History + usage	Analyse tracking module and in particular ID of computers that have accessed document.
Timewarp	Set of documents done by several people in timeline form.	Single document combined from input documents.	Uses timelines to piece together different documents.
H.P. Trust cloud	All documents stored in the trust cloud and interactions.	List of any accesses for each document.	System monitors interactions and reasons about origins.
Digital Thread	Input documents each with a digital thread attached.	History of a document and connections between documents.	Uses digital thread to follow file renames etc. Uses patented algorithms to reason

			about interactions.
Copy aware Ecosystem	Document metadata as well as interactions.	Version set containing the different versions of a document in one object.	Analyses interfile provenance relations using metadata as well as user interactions.
Tasktracer	Resources used.	Number of tasks each with all the associated resources.	Observes user interactions to put resource with user specified tasks. Uses provenance data to find most frequent paths.
Document DNA	Input documents and the actions performed on them.	Document connections and similarities.	Analyses the DNA of a document and uses algorithms to determine how far apart two documents are.

Figure 7: Summary of approaches for observing user interactions

Figure 8 shows the major similarities and differences between the solutions in the monitoring user interactions group and is a summary of what was in Figure 7.

Similarities	Differences
All aware of document history	Some focus on actions applied to a single document.
Most output version history.	Some focus on relationships from actions applied between documents.
	Some focus on content changing within single document.

Figure 8: Similarities and differences of methods

5. Working with static input documents

The second group of implementations involve content analysis on static documents. These implementations have no information on the interactions a person made with the documents but only a final set of documents that the user wishes to classify. Content classification is often used as this is some of the only information known for the documents. This group of solutions can usually show the similarities between documents but will find it difficult to show that one document is derived from another unless further information is available.

2.11. Classification based on annotations.

One example of this is documented in a patent by Schilit et al. [12]. This patent describes a system where the input is a collection of documents and the output is a number of groups to which the input documents have been assigned. This input and output is common for this group of document classifying software. This particular implementation uses annotations that users have made to the documents to try and find similarities between documents and eventually put them in appropriate groups. Annotations can be highlighting certain passages, a reaction to a certain part or a number of other possibilities, see Figure 9. Annotations can also be comments made on existing annotations such as writing "good idea" next to an existing annotation. The software extracts all of the annotations that a person has made to each document and then looks to analyse these. The annotations are grouped based on their proximity in time and space. The system also provides an organised list of the annotations made to the documents with links to the text so the user can follow the annotations to their location in the original documents. This system is reliant on there being a number of detailed annotations made throughout all of the documents in the collection.

Back to our example on finding documents related to Edmund Hillary. This solution is looking within the document but at annotations rather than the text itself. For two documents on Hillary to be grouped they would have to be identical documents as this is only looking at the positioning of annotations. If two documents are similar but not identical, the annotations will be in difference places.

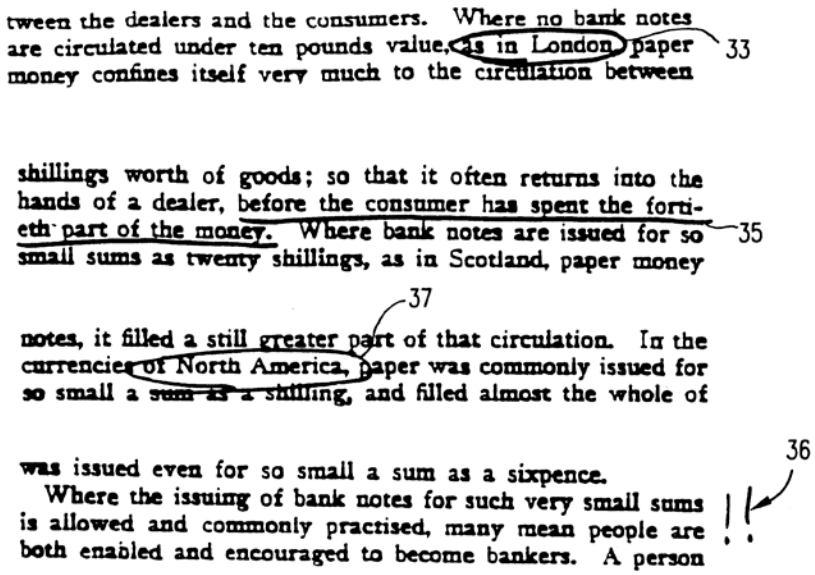


Figure 9: Annotation based classification, Schilit et al [12]

2.12. Classifying on sensitivity attributes.

Another implementation is that described in the patent by Kasiraj et al. [13] . This system takes a collection of documents as input each with a classification relationship attribute which contains sensitivity values as well as a relationship between the documents. The output is a collection of documents treated as a single entity with a single document classification value. The system can then decide if a certain individual is allowed access to the collection based on their own credentials. A documents classification relationship value contains a sensitivity value that shows the credentials a person requires to gain access to the document. The relationship also has information that links the document to a particular set. The software contains algorithms that calculate the pairings of documents based on their sensitivity attributes and relationships to give a single classification. The software is not so much concerned with outputting groups of similar documents but rather how a final collection of documents relates in terms to sensitive information to determine its access rights.

This implementation isn't concerned with finding related documents but rather assigning a value for a group of documents. It isn't so relevant for our example with Hillary. It is only relevant for this study as it shows how they've worked with finding a relationship between documents in terms of static values. For us to apply this to our project, each document would have to have a value for its relationship with

other documents and we might be able to use a similar method to assign a content relationship value for the combination of two documents.

2.13. Classifying by chunking byte code of text

HP proposed another method of text categorization [14]. This method again looked at only classifying the final documents rather than observe the user interactions. They have millions of technical documents and they require a method to find related or duplicate documents when making revisions. They state that they chose to look only at the content of the document rather than metadata associated with the document as the metadata may be incomplete so content analysis was more reliable. The input is millions of support documents and the output is a set of related documents. The approach they use is to break the byte code stream from the documents into chunks using a content based chunking algorithm, see Figure 10. They then compute a hash value for each of the chunks in all of the documents and the hashing algorithm is made sufficiently complicated so that two different chunks will not have the same hash value. Comparisons are made on the documents by finding all that share common hash values which indicate common text. In this example they are looking for chunks that are identical so the hashing algorithm is simpler and doesn't allow for any variation. This is an example of a purely content based method of classifying a large set of documents. A graph is constructed to determine which files are similar, see Figure 11. Each file is shown on a graph along with a node for each hashed chunk from the database of files. A line is drawn if a file contains a certain chunk. In Figure 11 each of the files has a duplicate in another directory so there are more connection lines than necessary. Two files are considered related if the total number of shared chunks is over a certain threshold percentage of the size of the files.

For our example on Hillary, if two documents share similar phrases on Hillary, these will have the same hash value and thus will appear related. With hashing, the hash value will be different with even a single different symbol however there are algorithms that allow for slight differences in the input to hash. This solution will also be useful for finding documents that are derived from each other, if two documents share identical sentences and phrases, it is likely one is a version of the other. This solution can be used to find similarity as well as versions potentially.

ITSM is shipped with the Inventory Reconciliation Manager (IRM), which allows you to build interfaces with the ITSM Configuration Manager Database (CMDB). The extractor, necessary to extract data from DTA, can be built with the Extractor Developer Kit (EDK) also delivered with the product. The IRM is delivered with an example extractor for DTA and for NNM, Asset Manager and Microsoft SMS. Next to the IRM, ITSM delivers the Event Interface Developers Kit (EIDK), enabling to build interfaces with the ITSM helpdesk for automatic upload and maintenance of service calls. The EIDK is delivered with example interfaces for NNM, DTA, ManageX, Tivoli TEC and Remedy ARS.

Figure 10: Example of text broken into chunks, Kasiraj et al. [13]

Figure 11: Graph of shared chunks between files, Kasiraj et al. [13]

2.14. Classification using word frequencies

Stanford University reported on a system they have used to detect plagiarism [15]. They have a number of large databases containing students work as well as reports submitted to journals. They needed a way to efficiently check if a document had been plagiarised from any of the documents in the databases. The input is the suspicious document and the large databases of reports. The output is a decision on whether the document had been plagiarised. The approach they took was to compare word frequencies of the suspicious document and any document in the databases. They initially looked at counting common sentences but this computation took far too long. Using word frequency and some

fairly complex statistics they can get a good measure of the similarity of two documents. Again this method only uses the static final documents and has no knowledge of how the documents were created. They point out that two documents can have similar word frequencies and not be copied from one another. Even using complex statistics it is still necessary for a human to check the suspected document and the original against each other to make a judgement on whether the document was plagiarised.

Two documents on Edmund Hillary will be considered similar if they contain similar word counts of key words such as Hillary, Everest, climber etc. So this method is definitely useful for finding similarities in topic. Surprisingly this method is also able to identify versions. If you look at word frequencies for all the words in the document and not just the main words it's possible to determine if documents are likely to be copies. The method uses some fairly complex statistics for this. The time taken is an important criteria of the methods and will be considered when choosing the most appropriate.

2.15. Clustering documents using Wikipedia

Anna Huang recently completed her PHD at Waikato university working with Pingar. Huang wrote about clustering documents using Wikipedia [16], [17]. The input is a set of documents and she aims to cluster them into groups based on content. She uses Wikipedia as a tool to analyse the documents and Wikipedia is becoming an increasingly useful resource for classifying documents as it is a huge, well structured collection of information. She starts by creating a concept-based document representation by mapping the terms and phrases within documents to their corresponding articles (or concepts) in Wikipedia. Each of the key terms within the document is extracted and mapped to a page in Wikipedia. She uses a similarity measurement that evaluates the semantic relatedness between concept sets for two documents. Wikipedia has a set of related pages for each page and Huang uses these to establish relationships between concepts in documents. Huang states that previous work considers the overlap of concepts in two documents but not the relationship between concepts i.e. the semantics. She discards some concepts to be left with only those related to the central theme of the document.

Back to our Hillary example. This method will identify the key concepts such as Hillary and climbing and establish the connection between them. It will then compare documents to see how many concepts overlap as well as the relationships between them. Two documents containing related information on Hillary would show up as having many shared entities and also will show the relationship between

these entities. It will find the similarity of documents but may not be able to find versions of the same document.

2.16. Static input summary

There are less existing implementations in this group than there are for the group that observes interactions. One of the main features is that these solutions are less intrusive. They require no software to be present on the user's computer watching them prepare documents as they are only getting the documents when they are to be classified. Figure 12 summarizes each of the approaches.

Approach	Input	Output	Method
Annotations	Documents plus annotations	Grouped documents plus list of all annotations and link to the text	Analyses the annotation in time and space.
Sensitivity Attributes	Documents with Sensitivity attributes	Group of documents with new sensitivity attribute	Calculates based on the sensitivity attribute and relationship to other groups of documents
Hashed Chunks	Documents	List of near copies of documents	Finds documents that share a certain number of hashed chunks
Word Frequencies	Suspicious document plus database of documents	Whether document has been copied off an existing document	Analyses the word frequencies of suspicious document compared to existing documents
Clustering using Wikipedia	Documents	Related groups of documents	Uses entity extraction and links entities to Wikipedia. Uses Wikipedia related pages to calculate relationships.

Figure 12: Summary of methods that use static input documents

6. Overview of Design

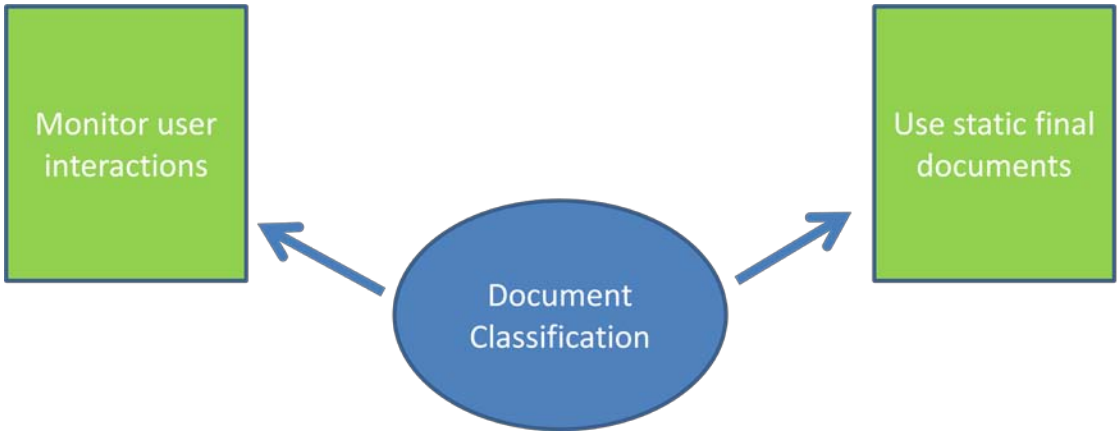


Figure 13: Subgroups of document classification

After looking at the solutions currently available (groups shown in Figure 13) and communicating with Pingar we came to the conclusion that we would use a system that uses only static input documents.

In collaboration with Pingar we finalised the specifications and structure of the project, see Figure 14. We would be using content analysis to compare documents rather than observing document interactions like a number of the other methods. The input of the program is an archive of static documents that can be in a number of formats and we have no information on a documents history other than its last modification and creation dates. Pingar software will initially be used to analyse the documents. The Pingar software will extract named entities and also produce a taxonomy from the documents. The taxonomy shows the relationships between the extracted entities (e.g. a hierachical structure). The software also gives information on which documents an entity occurs in. Named entities are phrases that contain the names of persons, organizations and locations [2]. So in our example Edmund Hillary is an entity and he is a climber and a climber is an explorer which are also both entities. The entities are the objects and the taxonomy gives the semantics or relationships between them. The named entities and associated taxonomy will then be fed into the software that I will create and the output of the system will the similarities between the input documents. I will consider the original text as well as the summarised output from Pingar software. The blue rectangles are software and include the Pingar software as well as the software I will create. The green ovals show the output from the Pingar software. These outputs are both required to be useful and are then fed into my software. The text without colour show the input and output of the entire system. The red rectangle is the similarity measure which will be discussed shortly. As we don't have any modification information

on the input documents it is difficult to find relationships between the documents but this will be attempted depending on time available.

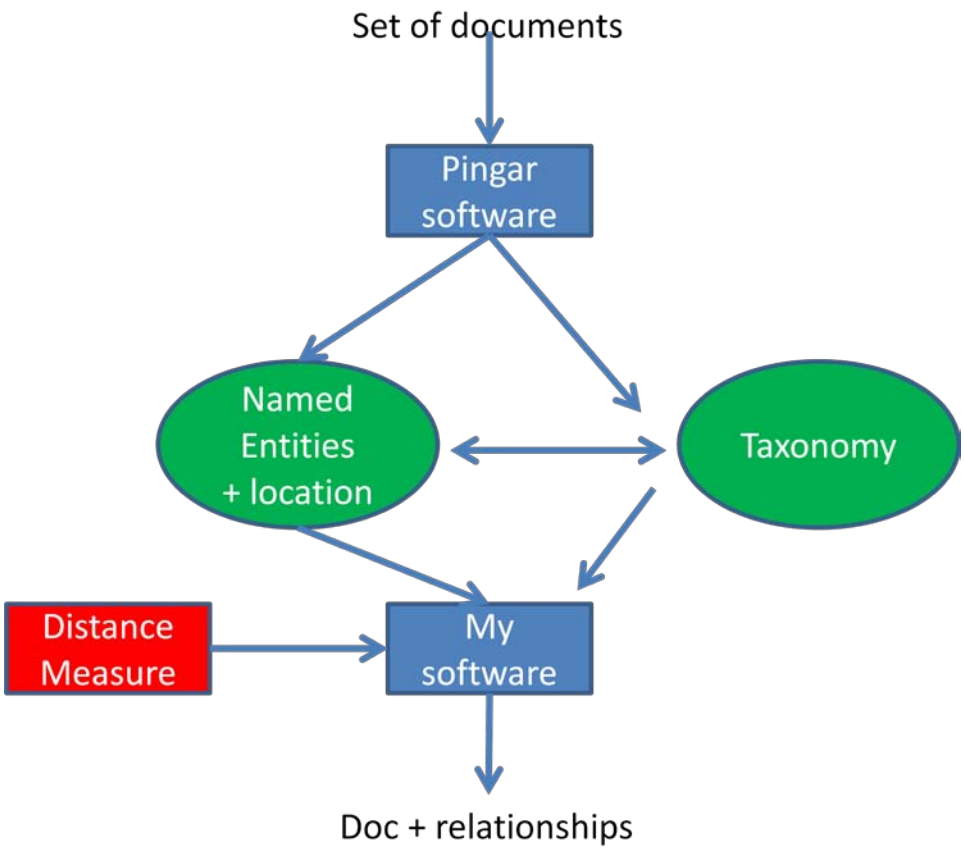


Figure 14: Overview of the design with Pingar

7. Similarity Measure

The next stage of literature review involves analysing existing software to see what similarity measure they use to decide how similar 2 documents are. Earlier I was looking at the static approaches to compare them to the approaches that monitor user actions. Now I will be looking at the approaches more carefully and in particular at the similarity measure they use. This similarity measure will be fed into my document classification software to use in the comparisons to other documents. Sebastiani states "The dominant approach to document classification is based on machine learning techniques and classification is usually done automatically by the learner using one of the available machine learning packages" [18]. I'll look at whether machine learning techniques are relevant as well as other methods. Each of the approaches in the static input method has a similarity measure and in this section I'm considering all of these and also introducing a new hashing technique. I aim to find which similarity

measure I should use. I also have to determine what criteria I use to decide whether documents are related. Obviously documents should be grouped if they are all talking about Hillary, but are two documents related if one is talking about Hillary and one about climbing in general.

To further illustrate how each comparison works I'm going to refer to 4 very brief sections of text which I will treat as separate documents.

1. Edmund Hillary is a great and well respected New Zealander.
2. Edmund Hillary is a well respected New Zealander
3. New Zealander Edmund Hillary is well respected worldwide.
4. Everest is the world's tallest mountain and is part of the Himalayas.

8. Measurements

8.1. Shared Hash Chunks

The HP method of chunking the document text [14] uses one measure for determining similarity. HP find the number of shared hashed chunks that two documents have, Figures 10 and 11. The documents are considered similar if the number is higher that a certain threshold percent of the size of the document. This will work if two documents have common phrases. An issue is that a hash value will be different if even a single symbol changes so the phrases will have to be identical. Two documents could be on very similar topics but with no shared phrases so they won't show any similarity. Using the 4 example texts, it's probable that none will be shown as related as none share common chunks thus they will hash differently. Later on I'll introduce hash calculations that allow for a bit of variation. This option won't work so well with the named entity input I'm getting from the Pingar software however I can use this method on the original text as further evidence that documents are related. The hashing method used is a minhash method using only a single hashing algorithm.

8.2. Word Frequencies

The method of text classification proposed by Stanford University introduces another similarity measure [15]. Two documents are compared based on their word frequencies. Some complex statistics are used to determine the chance that two documents are related based on how similar the word frequencies are. A human is required to do a final check on the two documents as the system is known to give false positives. This method will show that texts 1, 2 and 3 are related due to their similar word frequencies for words like New Zealander, Edmund Hillary and well respected. This method could be

taken further to find document versions if word counts for some of the smaller words are considered. Texts 1, 2 and 3 have similar word frequencies for smaller words such as "is" and "a" so it can be shown that they are versions of each other as well as semantically similar. One issue with this method is that it's not using all the information we have available. We are getting a taxonomy and named entities as input so we have the semantics/relationships between entities. This word frequencies example doesn't use the relationship between words.

8.3. Bag of Concepts

This method was developed by Anna Huang [16], [17]. It finds the key concepts in a document and maps each one to a Wikipedia page. She can identify the relationships between concepts with the help of Wikipedia which provides related pages. She then uses machine learning algorithms to cluster the documents. With our example texts, the first 3 will be mapped to the wikipedia page for Edmund Hillary and New Zealand. It will then be shown that the texts are related. This method can be smarter than the previous two and determine that "Everest" is related to "Edmund Hillary". With this semantic knowledge it can possibly determine that the 4th text is related to the first three as well.

8.4. SimHash

SimHash is a variant on the minhash similarity estimate. Given two sets, minhash [19] outputs a value between 0 and 1 to show the similarity of the two sets, 0 for disjoint sets and 1 being identical sets. In our project each set will be a chunk within a document. Simhash is introduced with the formula $Prh \in F[h(x) = h(y)] = sim(x, y)$ [20]. F is a family of hash functions and each is applied to each of the sets (chunks within documents being compared). Each set will have a number of values from this, one for each of the hashing functions. This output is represented by a vector and the vector for 2 sets is compared for similarity. This should be more accurate than the minhash as it's using a number of different hashing functions however it will be more computationally expensive. This method should be able to handle slight variations in sentences and still find them to be related thus showing related documents. This method will be better at determining document versions than the minhash method used in the shared hash chunks. This method can likely show that the first 2 texts are versions of each other as when they are put through a number of different hashing algorithms they will have some shared hash values. Texts 1 and 2 are identical except text two has a words "great and" removed. Simhash should be able to identify this and decide they are related where minhash probably wouldn't.

Similarity Measure	Semantic relationship	Document versions
Shared Hash Chunks	Only if the texts are identical, not so good at finding related content.	Only if texts are identical. Will show two texts are versions of each other if they share exact phrases
Word Frequencies	Shows relations if similar word frequencies. Doesn't consider semantics so needs to be same terms and not related terms.	Will show versions if smaller words are considered as well as main terms.
Bag of concepts	Does well in this area. Will show documents are related if contain the same named entities or ones that are semantically similar.	Not so good at finding versions.
Simhash	Only if the documents are very similar. Not so good at finding related content.	Good at finding versions due to a number of hashing functions being used.

Figure 15: Summary of similarity measures

9. Conclusion

It is obvious from above that each of the methods that uses static document input has an associated similarity measure. It is not as important for the implementations that observe the user interacting with documents as they are aware of the actions and can easily decide on groupings. With consultation with Pingar we have decided what we should aim for with the project. The first stage is to find documents that are semantically similar. This means documents on similar topics. We will also attempt to find documents that are related in terms of version. This last part will be difficult and we may run out of time.

Combining more than one of the similarity measures would be the best way to achieve both of these goals. I plan on using a form of entity frequency method that uses the output from the Pingar software to find related documents. I will follow this with a simhash algorithm on the original text to find versions of the same document. The specifics for on the hashing algorithm will be covered in the future research report.

10. References

[1] Karlson, A. K., Smith, G., & Lee, B. (2011, May). Which version is this?: improving the desktop experience within a copy-aware computing ecosystem. In *Proceedings of the 2011 annual conference on Human factors in computing systems* (pp. 2669-2678). ACM.

[2] Tjong Kim Sang, E. F., & De Meulder, F. (2003, May). Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In*Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4* (pp. 142-147). Association for Computational Linguistics.

[3] Buneman, P., Khanna, S., & Tan, W. C. (2000). Data provenance: Some basic issues. In *FST TCS 2000: Foundations of Software Technology and Theoretical Computer Science* (pp. 87-93). Springer Berlin Heidelberg.

[4] Johnson, B. L., Schroath, L. T., Anderson, B. J., & Herrmann, W. I. (2002).*U.S. Patent Application 10/236,441*.

[5] Edwards, W. K., & Mynatt, E. D. (1997, March). Timewarp: techniques for autonomous collaboration. In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems* (pp. 218-225). ACM.

[6] Ko, R. K., Jagadpramana, P., Mowbray, M., Pearson, S., Kirchberg, M., Liang, Q., & Lee, B. S. (2011, July). TrustCloud: A framework for accountability and trust in cloud computing. In *Services (SERVICES), 2011 IEEE World Congress on* (pp. 584-588). IEEE.

[7] Next Page. (2011). NextPage Digital Thread. Retrieved from http://www.nextpage.com

[8] Dragunov, A. N., Dietterich, T. G., Johnsrude, K., McLaughlin, M., Li, L., & Herlocker, J. L. (2005, January). TaskTracer: a desktop environment to support multi-tasking knowledge workers. In *Proceedings of the 10th international conference on Intelligent user interfaces* (pp. 75-82). ACM.

[9] Shen, J., Fitzhenry, E., & Dietterich, T. G. (2009, February). Discovering frequent work procedures from resource connections. In *Proceedings of the 14th international conference on Intelligent user interfaces* (pp. 277-286). ACM.

[10] Rinck, M. (2012). Full Research Proposal Connecting Information: Detecting and tracing object evolution. Manuscript Title. Waikato University.

[11] Rinck, M. (2013). Document DNA: How to Track Re-used Content Across Documents. Manuscript Title. Waikato University.

[12] Schilit, W. N., Price, M. N., Golovchinsky, G., & Wilcox, L. D. (2001). *U.S. Patent No. 6,279,014*. Washington, DC: U.S. Patent and Trademark Office.

[13] Kasiraj, C., Taylor, J. L., & Wolf, T. J. (1993). *U.S. Patent No. 5,204,812*. Washington, DC: U.S. Patent and Trademark Office.

[14] Forman, G., Eshghi, K., & Chiocchetti, S. (2005, August). Finding similar files in large document repositories. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining* (pp. 394-400). ACM.

[15] García-Molina, H., Gravano, L., & Shivakumar, N. (1996, December). dSCAM: Finding document copies across multiple databases. In *Parallel and Distributed Information Systems, 1996., Fourth International Conference on* (pp. 68-79). IEEE.

[16] Huang, A., Milne, D., Frank, E., & Witten, I. H. (2008, December). Clustering documents with active learning using Wikipedia. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on* (pp. 839-844). IEEE.

[17] Huang, A., Milne, D., Frank, E., & Witten, I. H. (2009). Clustering documents using a Wikipedia-based concept representation. In *Advances in Knowledge Discovery and Data Mining* (pp. 628-636). Springer Berlin Heidelberg

[18] Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, *34*(1), 1-47.

[19] Broder, A. Z. (1997, June). On the resemblance and containment of documents. In *Compression and Complexity of Sequences 1997. Proceedings* (pp. 21-29). IEEE.

[20] Charikar, M. S. (2002, May). Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing* (pp. 380-388). ACM.