Automatic construction of labeled clusters of named entities for information retrieval

Student: Teffera, Henock Tilahun

Student number: s1951866

Program: European Master's in Language and Communication Technology

supervisors: Dr. Gosse Bouma, University of Groningen

Dr. Valia Kordoni, University of Saarland

Prof. Hans Uszkoreit, University of Saarland

Date: July 07, 2010

Address: Van Houtenlaan 27, Groningen, Netherlands | **Email**: henocktilahun@gmail.com |**Tell**: +31-684457384

Contents

List of figures

List of Tables

Declaration

I hereby confirm that the thesis presented here is my own work, with all assistance acknowledged.

Groningen, 07 July 2010 **Signature:**

Acknowledgments

First of all I would like to thank the almighty God for letting pass all the challenges and worries of these two years. Thanks be to God for his indescribable gift!

This thesis would not have been possible without the guidance and support of my supervisors. I would like to pass my deep gratitude for Dr. Gosse Bouma, who has shown me all the ways starting from the proposal till the end, and PD.Dr Valia Kordoni, who has been given me valuable comments as a supervisor and whole hearted advice and assistance as LCT coordinator. I would also like to thank the LCT program coordinators as a whole for making my dream come true and allow me to taste the pure joy of undertaking scientific and technological explorations.

I am indebted to many of my classmates (Israel, Nick, Jon, Olga, Eliza, Gebre…) for supporting me with ideas, editing my work, evaluating my result and making the life easier. I wish you all great success in the life ahead of you.

I owe my deep gratefulness for my family and friends who shared my burden, helped me with idea and kept me in their prayers. Enatye, your advice and prayer has brought me to this point, Thank you so much.

Abstract

Within the general goal of information retrieval, i.e. finding the documents that are relevant to a user's information need, there is a problem in terms of the users' input and search engines' output. Users submit incomplete and not very informative queries and the search engine returns a list of documents which are considered to be most relevant because only the first few of them are checked by the users. One way of dealing with these problems is to organize documents based on their semantic similarity and allow users to navigate by means of category labels as well as by looking for similar pages.

In this study we have tried to harvest labeled clusters of semantically similar named entities which can be used as a first step for web document clustering. We first collect ~44,000 named entities from a thesaurus which is constructed by Dekang Lin applying a word similarity measure based on their distributional pattern. Using their similarity metrics and CLUTO clustering software, we create 2000 semantically similar clusters of the named entities. Then we collect ~305,500 label-instance pairs from the 2007 English Wikipedia dump and implement a labeling algorithm presented by Benjamin Van Durme and M.Pasça (2008) to assign a label to the clusters. This automatic lableing task is able to assign a label which describes the majority of the named entities in 924 of the clusters, which is 46.2% of the total clusters.

Finally we evaluate both the clustering and labeling tasks taking 86 randomly selected clusters and on the bases of two native English speaker evaluators' subjective judgment. According to these evaluators, the clustering task has a purity score of 0.7 and 55% of the labels are acceptable with different degree of accuracy. To check inter-evaluator agreement, we give them 20 similar labeled clusters and they get 0.6 and 0.5 kappa score for clustering and labeling result evaluation.

Chapter 1

1 Introduction

These days, people are overwhelmed by the large amount of information on the web. The information overload problem is worsening at an unprecedented speed. The number of pages available on the Internet almost doubles every year. To help alleviate the information overload problem and help users find the information they need many search engines have emerged (e.g. Google, AltaVista, etc.). These search engines commonly build a very large centralized database to index a portion of Internet and help to reduce the information overload problem by allowing a user to do a centralized search. However, they also bring up another problem: too many web pages are returned for a single query. To find out which documents are useful, users have to sift through hundreds of pages to find out that only a few of them are relevant. (Dragomir R. Radev, Weiguo Fan and Zhu Zhang, 2001). One way to tackle this problem is to cluster the search result documents based on their semantic similarity so that users can scan a few coherent groups instead of many individual documents.

In this paper, we present how we construct semantically similar clusters of labeled named entities which can be used for similar web document clustering and the development of a recommendation engine. The paper is divided into three chapters and they are organized as follows. The first chapter gives a general highlight of information retrieval, explains the problem of traditional web search information retrieval, the objective of the study and related works. The second chapter gives an overview of clustering and the selected cluster labeling approach; discusses the experiment conducted on both the clustering and labeling of named entities in detail and reports the result obtained. The third chapter points out types of cluster evaluation methods, reports the evaluation results, a general conclusion based on the result obtained, and our plan for the further study.

1.1 Information Retrieval

Information retrieval (IR) is a technique of finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers). The term “unstructured” refers to data which does not have clear, semantically overt and easy-for-computer structure. The field of information retrieval also covers supporting users in browsing or filtering document collections or further processing a set of retrieved documents. Given a set of documents, clustering is the task of coming up with a good grouping of the documents based on their contents. It is similar to arranging books on a bookshelf according to their topic. Given a set of topics, standing information needs, or other categories, classification is the task of deciding which class(es), if any, each set of documents belongs to. It is often approached by first manually classifying some documents and then hoping to be able to classify new documents automatically (Christopher D. Manning, Prabhakar Raghavan & Hinrich Schütze, 2009).

Information retrieval systems can be distinguished by three prominent scales at which they operate; *web search*, *personal information retrieval* and *domain-specific* search. In *web search*, the system has to provide search over billions of documents stored on millions of computers. Distinctive issues are need to gather documents for indexing, being able to build systems that work efficiently at this enormous scale, and handling particular aspects of the web, such as the exploitation of hypertext and not being fooled by site providers manipulating page content in an attempt to boost their search engine rankings, given the commercial importance of the web. At the other extreme is *personal information retrieval*. In the last few years, consumer operating systems have integrated information retrieval. Email programs not only provide search but also text classification: they at least provide a spam filter, and commonly also provide either manual or automatic means for classifying mail so that it can be placed directly into particular folders. Distinctive issues here include handling the broad range of document types on a typical personal computer, and making the search system maintenance free and sufficiently lightweight in terms of startup, processing and disk space usage that it can run on one machine without annoying its owner. In between the two scales of information retrievals, there is the space of *enterprise, institutional and domain-specific* search. In this scale retrieval might be provided for collections such as a corporation's internal documents, the database of patents, or research articles on biochemistry, where the documents will typically be stored on centralized file system and one or a handful of dedicated machine will provide search over the collection (Christopher D. Manning, Prabhakar Raghavan & Hinrich Schütze, 2009).

1.2 Problem in web search scale Information Retrieval

Within the general goal of, mainly web search scale, information retrieval, i.e. finding the documents that are relevant to a user's information need, Pasça observed a complication in terms of the user input and output. From the input side, users submit queries that contain a small number of words which are not very informative for the search engine. On the output side, the application provides only a limited number of documents which are considered to be the most relevant because users check only a few returned results. These challenges make the information retrieval task as hard as searching for a needle in a haystack, considering billions of web documents as the haystack and users' needs as a needle (Pasça, 2004).

Jansen has explained the behavior of most web searchers as they are adhered to the principle of least effort, which postulates that there are "useful" behaviors that are quick and easy to perform. The very existence of these quick, easy behavior patterns then cause individuals to choose them, even when they are not necessarily the best behavior from a functional point of view. However, they are good enough, and people will generally expend the least amount of effort to achieve what they want. He also noticed that the ranking algorithms of the Web search services are supportive of the typical usage pattern of Web searcher and adhere to the following rule: place at the top of the results list, those documents that contain all the query terms and that have all the query terms near each other (J.Jansen, 2000).

For web search engines working in large document collections, the resulting number of documents that matches a query can far exceed the number a human user could possibly shift through. Accordingly, it is essential for them to rank-order the documents matching a query. To do this, the search engines compute, for each matching document, a score with respect to the query at hand. The simplest approaches to compute a score between a query term *t* and a document *d* are term frequency and document frequency. Term frequency, denoted as $tf_{t,d,}$ is the number of occurrences of term *t* in document *d.* On the other hand, document frequency is the number of documents in the collection that contain a term *t.* Term Proximity is also used to measure that two terms in a query must occur close to each other in a document, where closeness may be measured by limiting the allowed number of intervening words or by reference to a structural unit such as a sentence or paragraph (Christopher D. Manning, Prabhakar Raghavan & Hinrich Schütze, 2009)

After looking at the unstructured text in documents and users' queries, Pasca realized that terms and phrases are not equal and the occurrences of named entities signals important information beyond measurements made on document frequency and term proximity. Users often search for list of names like *movies, countries*; popular names like *Paris, Harry Potter.* In other cases, users might search for unfamiliar rather than popular names, like *a reptile; a modern programming language or a Greek dragon,* to address more basic information needs. He explained the unstructured collection of documents as a goldmine that "hides" valuable information nuggets about various names, including nuggets that encode their categories as shown in Figure 1 below (Pasça, 2004).

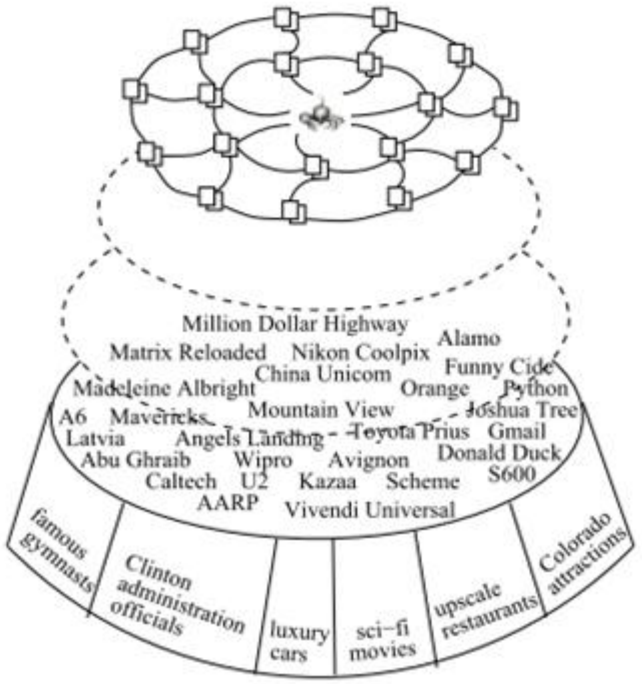


Figure 1: Names (middle layer) and their categories (lower layer) are hidden within the unstructured text of Web documents (Pasça, 2004)

On the above figure the names and their categories found in the middle and lower layer are hidden within the unstructured text of web documents at the upper layer. To “unhide” these information nuggets about named entities and their categories, Pasca presents a categorization approach for acquiring named entities in arbitrary categories from web documents (Pasça, 2004). Text categorization and clustering are two processes of deciding to which category a given text belongs with the following basic difference: In categorization, also called classification, the number of classes is already known and the goal is to sort the texts based on a well known structure; in clustering, the structures are not previously known and they will be discovered as part of the clustering process. When we make a computer discover the number of classes and decide to which preferred class a given text belongs we call the process automatic clustering. There are several objectives to consider when creating a partition of the text using categorization or clustering techniques. We might, for example, want to have sets of texts divided into groups of different levels of readability so that we can choose which text to read depending on our reading performance. We can also categorize texts according to their genres in different ways (Rosell, 2009).

1.3 Research goal

As we have discussed in the previous section, important information about named entities and their categories are hidden in an unstructured web document collection. In this study we try to show how to utilize this hidden valuable information for information retrieval. We will implement clustering approach to create a division among a collection of named entities based on semantic relatedness using a collection of named entities from a freely available thesaurus and their similarity matrix as a feature. We then automatically assign a class label for the clusters from parsed Wikipedia text using a TF/IDF like method introduced by M.Pasca and Van Durme (2008). We expect that the output of this research will contribute to the improvement of web search scale information retrieval as a first step for automatically detecting similarity between web pages and assigning class labels to web pages.

The overall aim of this study is trying to address the following research questions:

- How accurately can we cluster named entities taken from a thesaurus based on their semantic similarities?
- How accurately can we assign a class label to the derived clusters using parsed Wikipedia collection as a knowledge base?

1.4 Related works

Currently there is much interest in the automatic acquisition of lexical syntax and semantics, with the goal of building up large lexicons for natural language processing (Hearst, 1992). Many natural language processing applications, for instance information extraction and information retrieval, need semantically categorized collections of named entities as a resource. Traditionally these resources are constructed manually, like semantic lexicons and gazetteers, are limited to a small domain, and do not contain sufficient information relevant for the intended application

(Pasça, 2004). The following selected studies on the use of labeled named entities for natural language processing applications explain the relevance of automatic construction of labeled clusters of named entities.

Pasça has used categories of names acquired offline from the web to improve the search results for users' queries in an information retrieval application. He points out three areas where the resource can make enhancements to extracting relevant documents. The first enhancement is in processing list-type queries; when a user is interested in finding lists of items and the input query matches a known category, the engine returns the top names as representative elements of that category. The second improvement is in retrieval of siblings; when the user doesn't know the exact name to be queried, but is familiar with other similar names, categorized named entities guide the user in retrieving the relevant document by attaching the name into a set of possibly known names. The third use he has shown is in a query refinement application in which the search engine suggests related queries based on the analysis of a user query in relation to the extracted names and categories. Query refinement suggestions have the potential of affecting both names and categories. If the input query is a known name or category a set of related queries is generated and offered to the user as suggestions for refinement. This is in contrast with list-type queries and retrieval of siblings, which operate on input that matches either categories or names (Pasça, 2004).

Pantel and Ravichandran proposed an algorithm for automatically inducing names for semantic classes and for finding instance/concept (is-a relationship) pairs. They use the labeled semantic classes for question answering (answering definition questions) and passage retrieval for question answering tasks to approximate the recall of their automatic extraction approach. The result shows that integrating automatically labeled semantic classes gives a significant increase in performance on both tasks (Pantel, P. and D.Ravichandran, 2004).

These selected studies show that the output of this research can have a significant contribution in many areas of natural language processing. We are also planning to extend this research (not as part of a Master's thesis) by performing task based evaluation on a collection of web documents to automatically detect whether two pages describe semantically similar entities, which may be a good contribution to information retrieval research.

1.5 Background

To achieve our research goal we need to perform two steps; clustering named entities based on their semantic similarity and assigning class labels to the clusters. There are related studies which try to implement different approaches for these tasks. In this section, we briefly discuses some of these considerable approaches.

1.5.1 Clustering named entities

Different studies have been conducted which tried to locate the optimal and robust approach for finding similarity between named entities and categorizing them based on their semantic similarity. There are also many ongoing research projects which use clustered named entities as a resource for different applications. Traditionally recognition of named entities and their semantic categories within unstructured texts depends on the information found from semantic lexicons and gazetteers. Due to the restricted domain coverage of already available lexicon resources and the effort required to assemble them, the categorization task is limited to a domain for which large coverage resources already exist. Pasca uses a simple minimally-supervised method to acquire named entities in arbitrary categories from web documents to satisfy the diverse needs of users and overcome the overlapping of categories occurring in traditional approaches. The resulting categories provide enhancements to the search results of novel web search applications in addition to improving the existing knowledge sources (Pasça, 2004).

Wang and Cohen propose a language independent method which categorizes similar named entities from semi-structured web documents using a set expansion approach. In this approach the user provides a query consisting of a small number of seeds and the answer will be a list of other probable elements of the set. To find lists of entities on semi-structured pages, they use a novel technique which automatically constructs a collection of page-specific extraction rules, called wrappers, for each page that contains the seeds. They also use another novel approach to rank the list of entities returned by the wrapper. A graph is built containing all the constructed wrappers and all extracted candidate entities so that candidates are ranked according to a certain similarity measure in the graph (Wang, R.C. and W.W. Cohen, 2007).

In this study, we use a preexisting clustering approach implemented by Karypis (2003). He developed a software package called CLUTO for clustering low- and high-dimensional datasets and for analyzing the characteristics of various clusters. We discuss this freely available application in section 2.1.4 in detail.

1.5.2 Labeling clusters of named entities

As it was the case for clusters, different approaches have been proposed to assign a label for semantically similar clusters of named entities from unstructured collections of documents. Pantel and Ravichandran proposed a top-down approach for labeling semantic classes and for leveraging them to extract is-a relationships. They use concept signatures, templates that describe the prototypical syntactic behavior of instances of a concept, and extract concept names by searching for simple syntactic patterns. Their algorithm takes a list of semantic classes in the form of a cluster of words as an input and provides a ranked list of concept names for each semantic class, (Pantel, P. and D.Ravichandran, 2004).
In their open-domain class and class attribute acquisition study, Pasca and Van Durme introduced a method that mines a collection of web search queries and a collection of web documents to acquire open-domain classes in the form of instance sets associated with class labels as well as large sets of open-domain attributes for each class. For instance, open-domain classes like (*whales, seals, dolphins* and *sea lions*) can have a label *"marine animals*" and open-domain attributes such as (*circulatory system*, *life cycle*, *evolution*, *food chain* and *scientific*

name) for each class. The attributes capture properties that are relevant to the class. The extraction of attributes exploits the set of class instances rather than the associated class label, and consists of four stages. The first step is identification of a noisy pool of candidate attributes, as remainders of queries that also contain one of the class instances. The second step is construction of internal search-signature vector representations for each candidate attribute, based on queries that contain a candidate attribute and a class instance. The third step is construction of reference internal search signature vector representation for a small set of seed attributes provided as input. A reference vector is the normalized sum of the individual vectors corresponding to the seed attributes. The final step is ranking of candidate attributes with respect to each class by computing similarity scores between their individual vector representations and the reference vector of the seed attributes. The result of the four stages is a ranked list of attributes for each class (M.Pasça and Benjamin Van Durme, 2008).
We will mainly depend on the approach presented by Van Durme and M.Pasca (2008), which tries to extract large numbers of semantic classes along with their corresponding instances. This study tries to apply minimal constraints on the set of related terms to give rise to more strongly coherent classes. Durme and Pasca introduce a simple TF/IDF-like algorithm for deriving labeled classes of instances from unstructured open domain text. The algorithm takes a large collection of instance and class label pairs and a cluster of semantically related phrases. It then checks the viability of a label based on intra-cluster and inter-cluster constraints. An experiment was conducted on approximately 100 million English web documents and clusters of related terms collection constructed using a distributional similarity matrix following the approach of (Lin,D. and Pantel,P., 2002). The result shows the method is successful in separating high quality class/instance pairs from the lower average quality input data. A separate experiment on the attributes extraction task was conducted, in which they acquire a list of ranked class attributes from query logs based on a set of instances and a set of seed attributes input for each class. The result showed the large set of classes acquired from text produced attributes with an improved accuracy level. The simplicity of the algorithm they present for deriving labeled classes of instances from unstructured open domain text convinced us to follow their approach for our study.
In the next chapter, we discuss how we use the preexisting clustering approach to cluster collection of named entities taken from automatically constructed thesaurus. We also show how to extract a large collections of {Label, Instance} pairs from a parsed Wikipedia files. Finally, we present how we use these semantically similar clusters of named entities and {Label, Instance} pair collections as an input for the algorithm and extract a class label for the clusters.

Chapter 2

2 Experiments and result

As we have mentioned in the previous chapter this study has two subtasks; extract clusters of named entities from a freely available thesaurus based on semantic similarity and assigning a class label to the clusters. In this chapter, we present details of our experiment for extracting the named entity clusters and their labels together with the approach we have followed. The chapter is divided into two sections. The first section deals with clustering of ~44,000 named entities taken from a dependency based thesaurus assembled by Dekang Lin(1997). The second section explains how we extract ~305,500 instance-label pairs from parsed Wikipedia files and use them to find a label for the clusters.

2.1 Clustering named entities

2.1.1 Introduction to clustering

Zhao and Karypis define Clustering as the task of organizing a set of objects into meaningful groups which can be disjoint, overlapping or organized in some hierarchical fashion. Clustering Algorithms can be classified into different classes based on the methodology they follow, the structure of the final solution, the characteristics of the space in which they operate and the type of cluster they discover (Ying Zhao and George Karypis, 2003).

Based on their underlying methodology clustering algorithms can be p*artitional* or *Agglomerative*. Partitional algorithms find the clusters by partitioning the entire dataset into either a predetermined or an automatically derived number of clusters. They compute a k-way clustering of a set of objects either directly or via a sequence of repeated bisections. Partitional clustering can be considered as an optimization procedure that tries to create high quality clusters according to a particular criterion function. These criterion functions reflect the underlying definition of the "goodness" of clusters (Ying Zhao and George Karypis, 2003). An agglomerative algorithm is a greedy algorithm that starts with a separate cluster for each object. In each step, the two most similar clusters are determined and merged into a new cluster. The algorithm terminates when one large cluster containing all objects of a group average similarity function has been formed, which then is the only remaining cluster in the set of clusters (Christopher D. Manning and Hinrich Schutze, 1999).

Depending on the structure of the final solution clustering algorithms can be *Hierarchical* or *Non- hierarchical (Flat)*. Hierarchical clustering algorithms create clusters having a hierarchy between them with the usual interpretation that each node stands for subclass of its mother node. The leaves of the tree are the single objects of the clustered set. Each node represents the cluster that contains all the objects of its descendants. Non-hierarchical (Flat) clustering algorithms on the other hand create a certain number of clusters and the relation between clusters is often undetermined. Most of these algorithms are iterative, i.e. start with a set of initial clusters and improve them by iterating a reallocation that reassigns objects. Hierarchical clustering algorithms are preferable for detailed data analysis because they provide more information than flat clustering. Non-hierarchical clustering algorithms on the contrary are preferable if efficiency is a consideration or data sets are very large (Christopher D. Manning and Hinrich Schutze, 1999).

Based on the characteristics of the space in which they operate clustering algorithms can be *feature-based* or *similarity-based.* In feature-based clustering algorithms, each object is represented as a multi-dimensional feature vector and the clustering solution is obtained by iteratively optimizing the similarity between each object and its cluster centroid. On the other hand, similarity-based algorithms compute the clustering solution by first computing the pairwise similarities between all the objects and then use these similarities to drive the overall clustering solution. The advantage of similarity-based methods is that they can be used to cluster a wide variety of datasets, provided that reasonable methods exist for computing the pair-wise similarity between objects. Conversely, they have two limitations. First, their computational requirements are high as they need to compute the pair-wise similarity between all the objects that need to be clustered. As a result such algorithms can only be applied to relatively small datasets. Second, they tend to produce sub-optimal clustering solutions especially when the similarities are low relative to the cluster sizes because they don’t use the object’s feature space and rely only on the pairwise similarities. Since feature-based clustering algorithms do not require the pre-computation of pairwise similarities and their clustering decisions are made in the object’s feature space, they can overcome both of these limitations (Ying Zhao and George Karypis, 2003).

Considering the type of cluster they discover, we can classify clustering algorithms as *Globular* and *Transitive.* The differences between these clustering algorithms are the dimensions of their feature space and the relationships between the cluster’s objects. With regard to dimensions, Globular clustering algorithms create clusters containing objects that exhibit a strong pattern of conservation along a subset of their dimensions, i.e. there is a subset of the original dimensions in which a large fraction of the objects agree. However, in clusters formed using a transitive clustering algorithm there will be sub-clusters that may share a very small number of the subspace’s dimensions. In terms of the object-to-object similarity graph, Globular clustering algorithms create clusters that tend to contain objects in which the similarity between all pairs of objects will be high. On the contrary, Transitive clustering algorithms create clusters with a lot of objects whose direct pairwise similarity is quite low (Ying Zhao and George Karypis, 2003).

Clustering algorithms can also be classified as *vector clustering* and *graph clustering*. In vector clustering entities are represented by vectors, describing how each entity scores on a set of characteristics or features. The dissimilarity between two entities is calculated as a distance between the respective vectors describing them. Graphs are objects that have a much more combinatorial nature which can be expressed by notions such as degree, path, cycle, connectedness, etc. The graph model and the vector model do not exclude one another, but one model may inspire methods which are hard to conceive in the other model, and less fit to apply there. Vector methods have little to offer for the graph model, except that a generic paradigm such as single link clustering can be given meaning in the graph model as well.

Graph partitioning is a field where methods are studied to find the optimal partition of a graph given certain restrictions. Cluster analysis in the setting of graphs is closely related to this field. The generic meaning of "*clustering*" is in principle exactly that of "*partition*". The difference is that the semantics of "*clustering*" change when combined with adjectives such as "*overlapping*" and "*hierarchic*". A partition is strictly defined as a division of some set *S* into subsets satisfying two conditions; all pairs of subsets are disjoint and the union of all subsets yields *S*. In graph partitioning the required partition sizes are specified in advance. The objective is to minimize some cost function associated with links connecting different partition elements. Thus the burden of finding natural groups disappears (Dongen, Stijn Marinus van, 2000).

2.1.2 Uses of clustering in statistical NLP

There are two main uses of clustering in statistical natural language processing; for exploratory data analysis (EDA) and for generalization. Exploratory data analysis is the examination of data with minimal preconceptions about its structure through which it is hoped that relationships and patterns, at least some of which are unanticipated, will be uncovered (J.O.Ramsey and David Wiley, 1978). It is an important activity in any pursuit that deals with quantitative data. It is also the first step whenever we are facing a new problem and want to develop a probabilistic model or just understand the basic characteristics of the phenomenon. Clustering is a particularly important technique for EDA in statistical natural language processing because there is often no direct pictorial visualization for linguistic objects. Other fields, in particular those dealing with numerical or geographic data, often have an obvious visualization. Any technique that lets one visualize the data better is likely to lead to a new generalization and to stop one from making wrong assumptions about the data. When used for EDA, clustering is one of a number of techniques that one might employ for displaying a set of objects in a two-dimensional plane. Compared to other techniques clustering has the advantage that it can produce a richer hierarchical structure. It may also be more convenient to work with since visual displays are more complex.

The second main use of clustering in natural language processing is for generalization where we group objects into clusters and generalize from what we know about some members of clusters to others. For example, suppose we want to know the correct preposition to use with the noun Friday for translating a text from French into English provided that we have an English training text that contain the phrases on Sunday, on Monday and on Tuesday, but not on Friday. We can infer that 'on' is the correct preposition to use with Friday by clustering English nouns into groups with similar syntactic and semantic environments (Christopher D. Manning and Hinrich

Schutze, 1999). In this study we examine the benefit of clustering for assigning a general semantic category to named entities and assigning a generic name to these categories.

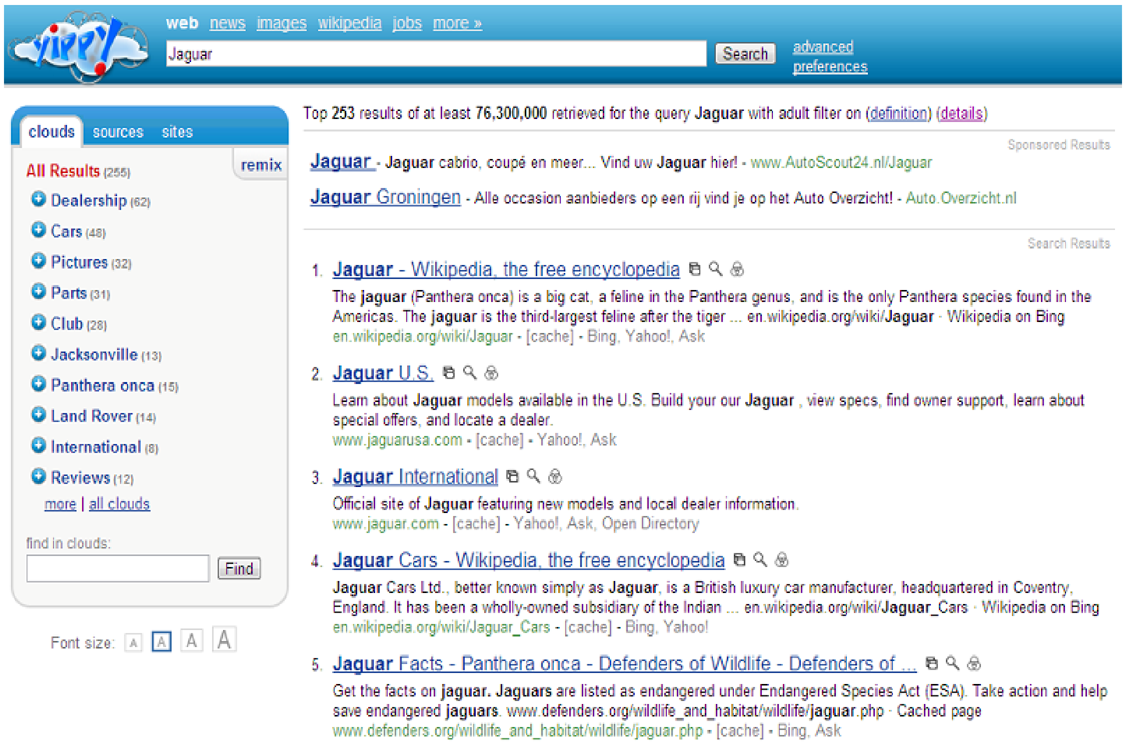
2.1.3 Clustering in information retrieval

Christopher D. Manning, Prabhakar R. and Hinrich S. (2009) present the cluster hypothesis, i.e. "*Documents in the same cluster behave similarly with respect to relevance to information needs*", as the fundamental assumption we make when using clustering in information retrieval. If there is a document from a cluster that is relevant to search request, then it is likely that other documents from the same cluster are also relevant. This is because clustering puts together documents that share many terms. They also describe some of the applications of clustering in information retrieval. These applications are; *Search result clustering*, *Scatter-gather*, *Collection clustering*, *Language modeling* and *Cluster-based retrieval*.

In the first application, search result clustering, search results refer to the documents that were returned in response to a query. The default presentation of search results in information retrieval is a simple list. Users scan the list from top to bottom until they have found the information they are looking for. Instead, search result clustering clusters the search results, so that similar documents appear together. It is often easier to scan a few coherent groups than many individual documents. This is particularly useful if a search term has different word senses. The example shown in figure 2 is the clustered results returned by Yippy search engine (http://clusty.com/)[1]. The clustered results panel can be a more effective user interface for understanding what is in the search results than a simple list of documents.

The second application, Scatter-Gather, has also the goal of creating a better user interface. It clusters the whole collection to get groups of documents that the user can select or gather. The selected groups are merged and the resulting set is again clustered. This process is repeated until a cluster of interest found. Cluster-based navigation is an interesting alternative to keyword searching, the standard information retrieval paradigm. Figure 3 below shows automatically generated hierarchical clusters tree.

[1]Clusty.com is the first full-service search engine site powered completely by Vivismio's breakthrough clustering technology. Instead of simply presenting long list of results, Clusty.com groups search results into folder topics, giving users a quick overview of the main themes in the results and letting them focus on topics of interest. Designed to relieve consumers of information overload and provide easy access to information that is usually overlooked with current search engines, Clusty changes how consumers do general web searches as well as shopping, blogs, gossip, images, Wikipedia and people searches (Baker).



Top 253 results of at least 76,300,000 retrieved for the query **Jaguar** with adult filter on (definition) (details)

All Results (255)
- Dealership (62)
- Cars (48)
- Pictures (32)
- Parts (31)
- Club (28)
- Jacksonville (13)
- Panthera onca (15)
- Land Rover (14)
- International (8)
- Reviews (12)

more | all clouds

find in clouds: Find

Font size: A A A A

Sponsored Results

Jaguar - **Jaguar** cabrio, coupé en meer... Vind uw **Jaguar** hier! - www.AutoScout24.nl/Jaguar

Jaguar Groningen - Alle occasion aanbieders op een rij vind je op het Auto Overzicht! - Auto.Overzicht.nl

Search Results

1. **Jaguar** - Wikipedia, the free encyclopedia
   The **jaguar** (Panthera onca) is a big cat, a feline in the Panthera genus, and is the only Panthera species found in the Americas. The **jaguar** is the third-largest feline after the tiger ... en.wikipedia.org/wiki/**Jaguar** - Wikipedia on Bing
   en.wikipedia.org/wiki/Jaguar - [cache] - Bing, Yahoo!, Ask
2. **Jaguar** U.S.
   Learn about **Jaguar** models available in the U.S. Build your our **Jaguar** , view specs, find owner support, learn about special offers, and locate a dealer.
   www.jaguarusa.com - [cache] - Yahoo!, Ask
3. **Jaguar** International
   Official site of **Jaguar** featuring new models and local dealer information.
   www.jaguar.com - [cache] - Yahoo!, Ask, Open Directory
4. **Jaguar** Cars - Wikipedia, the free encyclopedia
   Jaguar Cars Ltd., better known simply as **Jaguar**, is a British luxury car manufacturer, headquartered in Coventry, England. It has been a wholly-owned subsidiary of the Indian ... en.wikipedia.org/wiki/**Jaguar**_Cars - Wikipedia on Bing
   en.wikipedia.org/wiki/Jaguar_Cars - [cache] - Bing, Yahoo!
5. **Jaguar** Facts - Panthera onca - Defenders of Wildlife - Defenders of ...
   Get the facts on **jaguar. Jaguars** are listed as endangered under Endangered Species Act (ESA). Take action and help save endangered **jaguars**. www.defenders.org/wildlife_and_habitat/wildlife/**jaguar**.php - Cached page
   www.defenders.org/wildlife_and_habitat/wildlife/jaguar.php - [cache] - Bing, Ask

Figure 2: Yippy search engine result for the keyword "Jaguar"

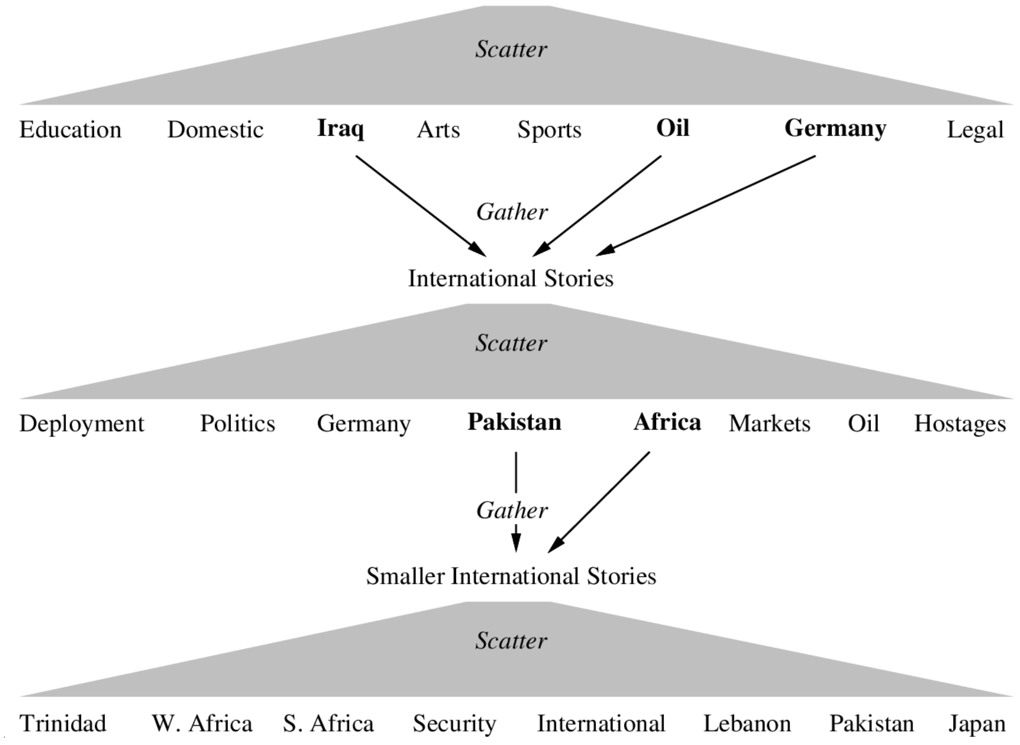


Figure 3: An example of a user session in Scatter-Gather (Christopher D. Manning, Prabhakar Raghavan & Hinrich Schütze, 2009).

In figure 3 above, a collection of New York Times stories is clustered ("scattered") into eight clusters (top row). Then, the user manually gathers three of these into a smaller collection "International stories" and performs another scattering operation. This process repeats until a small cluster with relevant documents is found.

Collection clustering is an alternative to the user-meditated iterative clustering in Scatter-Gather. In this application we can compute a static hierarchical clustering of collection that is not influenced by user interactions. Google News and Columbia NewsBlaster system follow this approach. In the case of news, we need to frequently re-compute the clustering to make sure that users can access the latest breaking stories. Clustering is well suited for access to a collection of news stories since news reading is not really search, but rather a process of selecting a subset of stories about recent events.

Language modeling application of clustering exploits the clustering hypothesis directly for improving search results, based on a clustering of the entire collection. We use a standard inverted index to identify an initial set of documents that match the query, but we then add other documents from the same clusters even if they have low similarity to the query. For example, if the query is a car and several car documents are taken from a cluster of automobile documents, then we can add documents from this cluster that use terms other than car (automobile, vehicle etc). This can increase recall since a group of documents with high mutual similarity is often relevant as a whole.

The fifth application, cluster-based retrieval, comes to action when we may not be able to use the traditional methods of finding the nearest neighbors to the query, for instance latent semantic indexing, efficiently. In such cases, we could compute the similarity of the query to every document, but this is slow. The cluster hypothesis offers an alternative: Find the clusters that are closest to the query and only consider documents from these clusters. Within this much smaller set, we can compute similarities exhaustively and rank documents in the usual way. Since there are many fewer clusters than documents, finding the closest cluster is fast; and since the documents match a query are all similar to each other, they tend to be in the same clusters. While this algorithm is inexact, the expected decrease in search quality is small (Christopher D. Manning, Prabhakar Raghavan & Hinrich Schütze, 2009).

2.1.4 CLUTO, clustering toolkit

For named entity clustering task we decide to use a preexisting clustering approach implemented by Karypis for two reasons. Fist, we can take advantage of using a freely available software package he developed, called CLUTO, and save the time we were supposed to spend implementing a clustering algorithm. Second, the method has been successfully used to cluster datasets arising in different application areas including information retrieval, commercial datasets, scientific datasets, and biological applications (Ying Zhao and George Karypis, 2003). CLUTO is preferred for clustering low- and high-dimensional datasets and for analyzing the characteristics of various clusters. It allows applying three different clustering algorithms; partitional, agglomerative, and graph-partitioning. The key feature of CLUTO's clustering algorithms is that they treat the clustering problem as an optimization process which seeks to maximize or minimize a particular clustering criterion function defined either globally or locally over the entire clustering solution space. It provides tools for analyzing the discovered clusters to understand the relations between the objects assigned to each cluster and the relations between the different clusters and tools for visualizing the discovered clustering solutions. It can also identify the features that best describe and discriminate each cluster which can be used to gain a better understanding of the set of objects assigned to each cluster and provide concise summaries about the cluster's content.

CLUTO has two clustering programs, vcluster ("v" stands for vector) and scluster ("s" stands for similarity), which are available for windows and Unix/Linux platforms. These programs are used to cluster a collection of objects into a predetermined number of clusters k. The key difference between these programs is that vcluster implements feature-based clustering algorithms whereas scluster implements similarity-based clustering algorithms. Therefore, vcluster takes the actual multi-dimensional feature vector representation of the objects that need to be clustered and finds the clustering solution by iteratively optimizing the similarity between each object and its cluster centroid. Scluster on the other hand takes the pairwise similarities between all the objects and uses these similarities to drive the overall clustering solution. The vcluster program treats each object as a vector in a high-dimensional space, and it computes the clustering solution using one of five different approaches. On the other hand, the scluster program operates on the similarity space between the objects and can compute the overall clustering solution.

The syntax for invoking these two programs on the command line is:

```
vcluster [optional parameters] MatrixFile NClusters

scluster [optional parameters] GraphFile Nclusters
```

Figure 4: Syntax for invoking CLUTO programs

Where, *MatrixFile* is the name of file that stores the n objects to be clustered, *GraphFile* is the name of the file that stores the adjacency matrix of the similarity graph between the n objects to be clustered and *NClusters* is the number of clusters that is desired (Karypis, 2003).

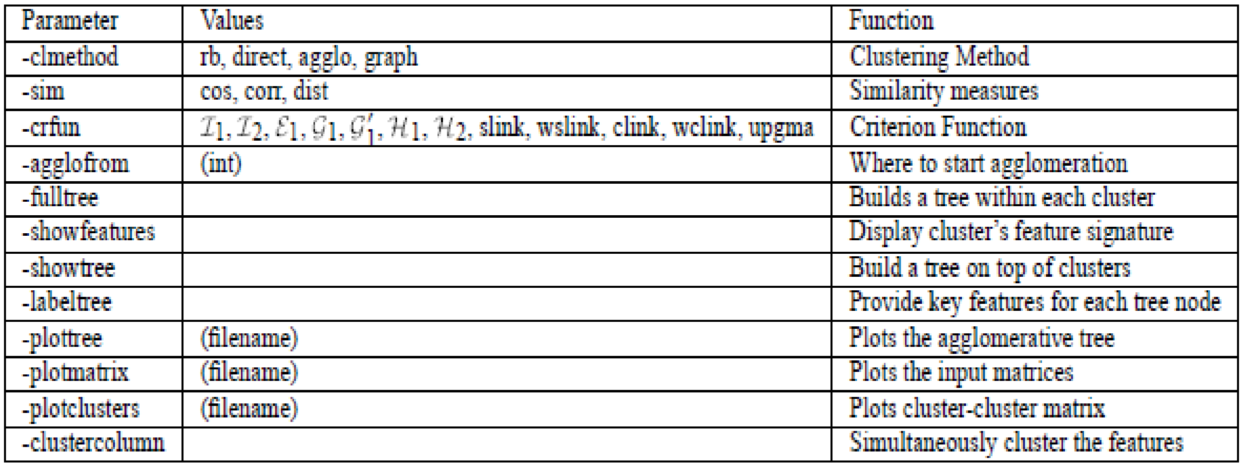
2.1.4.1 CLUTO's input file format

The vcluster program takes a matrix storing the objects to be clustered as primary input file. Each row of this matrix represents a single object, and its various columns correspond to the features of the objects. CLUTO understands two different input matrix formats; a format suitable for sparse matrices and a format suitable for dense matrices. A sparse matrix format input file for the vcluster program contains *n* rows and *m* columns stored in a plain text file that contains *n + 1* lines. The first line of the matrix file contains exactly three integer numbers. The first integer is the number of rows in the matrix (n), the second is the number of columns in the matrix (m) and the third is the total number of non-zeros entries in the $n \times m$ matrix. The remaining n lines store information about the actual non-zero structure of the matrix. The dense matrix format is suitable for storing dense matrices. If a given matrix has *n* rows and *m* columns, it is stored in a plain text file that contains n + 1 lines. The first line stores two integers, the number of rows (n) and columns (m) in the matrix, while the remaining n lines contain information for each row of the matrix.

The scluster program takes an adjacency matrix of the graph that specifies the similarity between the objects to be clustered as primary input file. As the vcluster program, it understands two different input graph formats. The sparse graph format stores the adjacency matrix of a sparse graph with *n* vertices in a plain text file. The first line contains two integer numbers. The first integer is the number of vertices in the graph (n) and the second is the number of edges in the graph. The remaining *n* lines store information about the actual non-zero structure of the adjacency matrix. The dense graph format is stored in a plain text file that contains n + 1 lines. The first line stores a single integer, which is the number of vertices (n) of the graph, and the remaining *n* lines contain information for each row of the adjacency matrix (Karypis, 2003).

2.1.4.2 CLUTO's algorithm parameters

The behavior of vcluster and scluster can be controlled by providing a number of optional parameters as shown in figure 4 above. These parameters are specified using the standard -*paramname* or -*paramname = value* format. Table1 below shows some of the most important parameters[1].

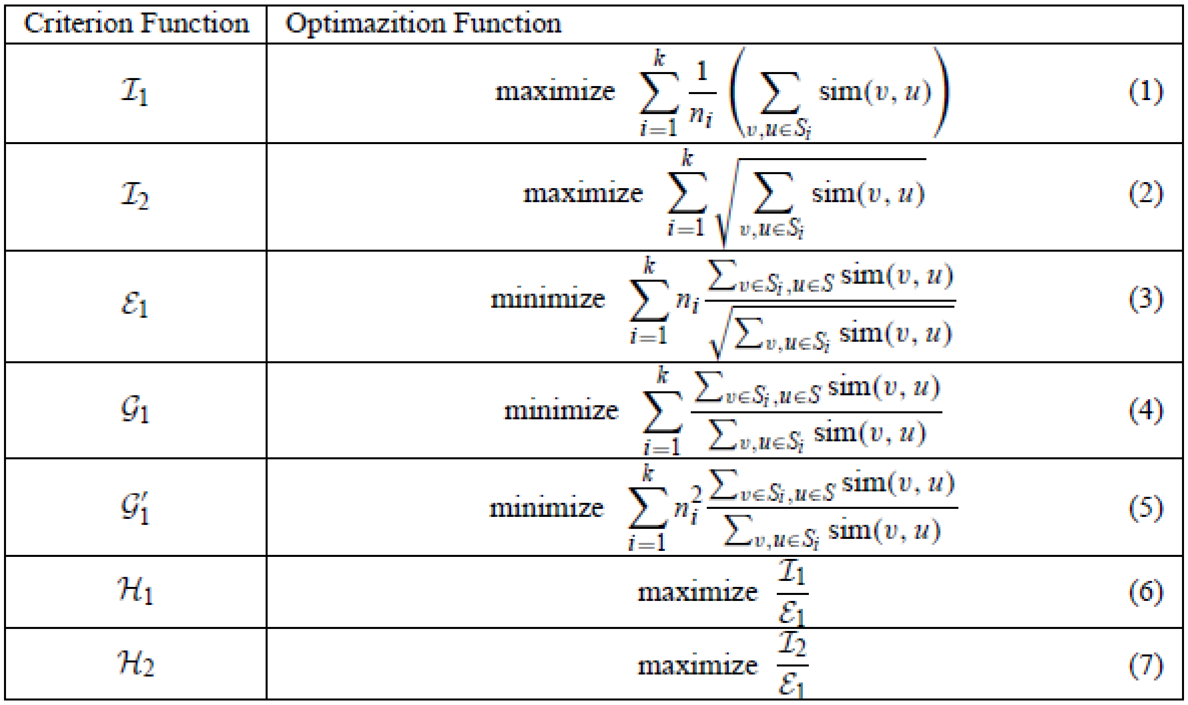


Parameter	Values	Function
-clmethod	rb, direct, agglo, graph	Clustering Method
-sim	cos, corr, dist	Similarity measures
-crfun	$\mathcal{I}_1, \mathcal{I}_2, \mathcal{E}_1, \mathcal{G}_1, \mathcal{G}'_1, \mathcal{H}_1, \mathcal{H}_2$, slink, wslink, clink, wclink, upgma	Criterion Function
-agglofrom	(int)	Where to start agglomeration
-fulltree		Builds a tree within each cluster
-showfeatures		Display cluster's feature signature
-showtree		Build a tree on top of clusters
-labeltree		Provide key features for each tree node
-plottree	(filename)	Plots the agglomerative tree
-plotmatrix	(filename)	Plots the input matrices
-plotclusters	(filename)	Plots cluster-cluster matrix
-clustercolumn		Simultaneously cluster the features

Table 1: Important parameters of CLUTO's algorithm (Ying Zhao and George Karypis, 2003)

[1] Full list of parameters with their values and explanation is available on the CLUTO manual, (Karypis, 2003)

The -*clmethod* parameter controls the type of algorithms to be used for clustering. The first two values, i.e. "rb" and "direct", follow the partitional clustering algorithm paradigm. The agglomerative algorithm paradigm can be applied using the "agglo" value and the "graph" value uses a graph-partitioning based clustering algorithm. The -*sim* parameter is used to define the similarity between object and it has three values; ("cos") supports the cosine, ("corr") supports correlation coefficient and ("dist") supports Euclidean distance derived similarity. The clustering criterion function that is used by the partitional and agglomerative algorithms is controlled by the -*crfun* parameter. Table 2 below shows their mathematical definition; where *k* is the total number of clusters, *S* is the total objects to be clustered, S_i is the set of objects assigned to the i_{th} cluster, n_i is the number of objects in the i_{th} cluster, *v* and *u* represent two objects, and *sim(v,u)* is the similarity between two objects. The first seven criterion functions are used by both types of algorithms whereas the last five are only applicable to agglomerative clustering. CLUTO's programs can also produce a number of visualizations that can help to see the relationships between the clusters using *plottree, plotmatrix* and *plotclusters* parameters (Ying Zhao and George Karypis, 2003).



Criterion Function	Optimazition Function	
$\mathcal{I}_1$	maximize $\sum_{i=1}^{k} \frac{1}{n_i}\left(\sum_{v,u\in S_i} \text{sim}(v,u)\right)$	(1)
$\mathcal{I}_2$	maximize $\sum_{i=1}^{k} \sqrt{\sum_{v,u\in S_i} \text{sim}(v,u)}$	(2)
$\mathcal{E}_1$	minimize $\sum_{i=1}^{k} n_i \frac{\sum_{v\in S_i, u\in S} \text{sim}(v,u)}{\sqrt{\sum_{v,u\in S_i} \text{sim}(v,u)}}$	(3)
$\mathcal{G}_1$	minimize $\sum_{i=1}^{k} \frac{\sum_{v\in S_i, u\in S} \text{sim}(v,u)}{\sum_{v,u\in S_i} \text{sim}(v,u)}$	(4)
$\mathcal{G}'_1$	minimize $\sum_{i=1}^{k} n_i^2 \frac{\sum_{v\in S_i, u\in S} \text{sim}(v,u)}{\sum_{v,u\in S_i} \text{sim}(v,u)}$	(5)
$\mathcal{H}_1$	maximize $\frac{\mathcal{I}_1}{\mathcal{E}_1}$	(6)
$\mathcal{H}_2$	maximize $\frac{\mathcal{I}_2}{\mathcal{E}_1}$	(7)

Figure 5: Clustering functions and their mathematical definition

The various criterion functions can sometimes lead to significantly different clustering solutions. In general, the I_2 and H_2 criterion functions lead to very good clustering solutions, whereas the $\mathcal{E}_1$ and $\mathcal{G}_1$ functions leads to solutions that contain clusters that are of comparable size. The computational complexity of the agglomerative clustering algorithms also depends on the criterion function that is selected; if n is the number of objects, the complexity for H_1 and H_2

criterion function is O(n^3), whereas the complexity of the remaining criterion functions is O(n^{2}log n) (Karypis, 2003).

2.1.5 Clustering Named Entities

2.1.5.1 Named entities taken from dependency based thesaurus

We use collections of named entities taken from an automatically constructed thesaurus together with their similarity matrix[1]. The thesaurus is constructed by Dekang Lin using a word similarity measure based on the distributional pattern of words. Lin defines a similarity between two objects as the amount of information contained in the commonality between objects divided by the amount of information in the descriptions of the object (Lin D. , 1997). To find the similarity between words, he first uses a broad-coverage parser and extracts dependency triples consisting of two words and the grammatical relationship between them from a text corpus. For example, the dependency triples for a sentence "*I have a brown dog*" are:

(have subj I), (I subj-of have), (dog obj-of have),
(dog, adj-mod brown), (brown adj-mod-of dog),
(dog det a), (a det-of dog)

Figure 6: Dependency triples for the sentence "I have a brown dog" (Lin D. , 1998)

He uses a notation ||w, r, w'|| to denote the frequency of count of the dependency count of the dependency triple (w, r, w'). If the wild card (*) replaces any of w, r or w', the frequency counts of all the dependency triples that matches the rest of the pattern are summed up. Then he assumes the frequency counts of the dependency triples as independent of each other, and calculates the information contained in the description of a word as the sum of the information contained in each individual frequency count. Considering *I (w, r, w')* as the amount of information contained in ||w, r, w'|| = c, which is also equal to the mutual information between w and w', its value is computed as:

$$I(w,r,w') = log\frac{||w,r,w|| \times ||*,r,*||}{||w,r,*|| \times ||*,r,w'||}$$

Equation 1: The amount of information contained in ||w, r, w'||

Finally, he defines the similarity, sim(w_1,w_2), between two words w_1 and w_2 as shown in the equation 2 below, where T(w) is the set of pairs (r, w') and I(w, r, w') is positive.

[1] The thesaurus is available at http://webdocs.cs.ualberta.ca/~lindek/downloads.htm

$$sim(w_1, w_2) = \frac{\sum_{(r,w)\in T(w1)\cap T(w2)} (\mathrm{I}(\mathrm{w}_1, \mathrm{r}, \mathrm{w}) + \mathrm{I}(\mathrm{w}_2, \mathrm{r}, \mathrm{w}))}{\sum_{(r,w)\in T(w_1)} \mathrm{I}(\mathrm{w}_{1,} \mathrm{r}, \mathrm{w}) + \sum_{(r,w)\in T(w_2)} \mathrm{I}(\mathrm{w}_{2,} \mathrm{r}, \mathrm{w})}$$

Equation 2: Similarity between two words w_1 and w_2

To construct the thesaurus, he extracted 56.5 million dependency triples containing 5,469 nouns, 2,173 verbs and 2,632 adjectives/adverbs which occur at least 100 times in parsed text corpus[1]. Then he computed the pairwise similarity between all the nouns, all the verbs and all the adjectives/adverbs using the above similarity measure and take the top-N similar words for each word (Lin D. , 1998).

For our research we take a more developed version of this thesaurus which contains 44,450 nouns and all nouns similar with them, with the pairwise similarity value greater than 0.04. We then put the nouns on a separate plane text file and construct a 44,450 × 44,450 sparse similarity matrix, which can be fed to CLUTO's scluster program, using the pairwise similarity value.

2.1.5.2 Clustering using CLUTO

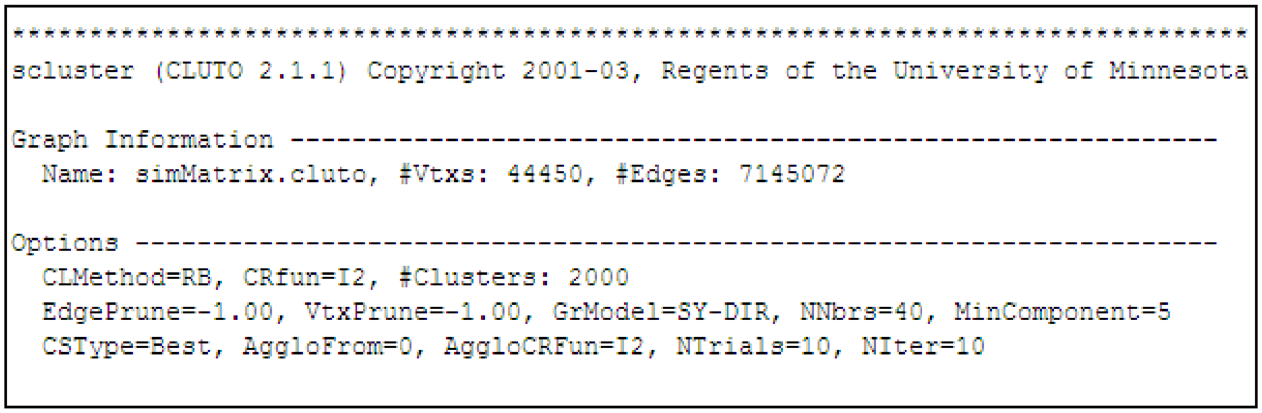
As we have discussed above, the decision to use either the scluster or vcluster program depends on the type of data to be clustered. Since we have sparse similarity matrix data, we use the scluster program and provide an input appropriate for sparse graphs with two integer numbers at the first line in a plain text file. The first integer is 44,450, which is the number of vertices in the graph, and the second integer is 7,145,072, which is the total number of non-zero entries in our similarity matrix. We then set the number of clusters and the optional parameters and invoke the scluster program on the linux shell as shown in figure 7 below.

```
/cluto-2.1.1/Linux/scluster simMatrix.cluto 2000
```

Figure 7: Invoking scluster program on Linux shell

Here, the first argument *(/cluto-2.1.1/Linux/scluster)* specifies that we are using CLUTO for Linux platform and it is version 2.1.1. The second argument (*simMatrix.cluto)* provides the plain text file which contains the sparse similarity matrix. The last argument (2000) tells the scluster program to cluster the named entities into 2,000 similar categories. Figure 8 below shows the initial output of scluster for the specified input.

[1] He used three text corpuses; Wall Street Journal (24 million words), San Jose Mercury (21 million words) and AP Newswire (19 million words).



```
********************************************************************************
scluster (CLUTO 2.1.1) Copyright 2001-03, Regents of the University of Minnesota

Graph Information -----------------------------------------------------------
  Name: simMatrix.cluto, #Vtxs: 44450, #Edges: 7145072

Options ---------------------------------------------------------------------
  CLMethod=RB, CRfun=I2, #Clusters: 2000
  EdgePrune=-1.00, VtxPrune=-1.00, GrModel=SY-DIR, NNbrs=40, MinComponent=5
  CSType=Best, AggloFrom=0, AggloCRFun=I2, NTrials=10, NIter=10
```

Figure 8: Initial output of scluster program

On figure 8 above, the first line and the *Graph Information* part of the output display the version of the program and information about the input we provided. The *Options* section specifies the values of the various parameters[1] that it uses to compute the clustering and the number of desired cluster (#Clusters). These parameters and their values are explained below:

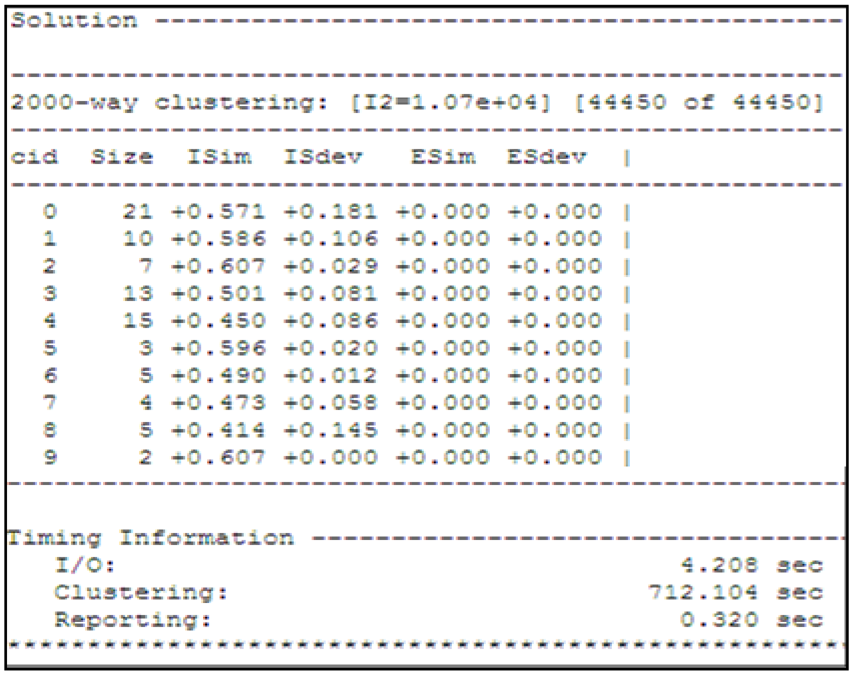
- The *CLMethod* parameter specifies the method used for clustering the names; in this case it is RB (repeated bisection), which computes the desired k-way clustering solution by performing a sequence of k – 1 repeated bisections.
- The *CRfun* parameter specifies the type of clustering criterion functions shown in figure 5 above; in this case it is I2.
- The *EdgePrune* parameter uses to eliminate certain edges from the nearest-neighbor graph that will tend to connect vertices belonging to different clusters; in this it has the default value (-1), indicating no edge pruning.
- The *Vtxprune* parameter uses to eliminate certain vertices from the nearest-neighbor graph that tend to be outliers; in our case it has the default value (-1), indicating no vertex pruning.
- The *GrModel* parameter controls the type of nearest-neighbor graph that will be constructed on the fly and supplied to the graph-partitioning based clustering algorithm; in our case it has *SY-DIR(Symmetric-Direct)* value which allows a graph to be constructed so that there will be edge between two objects if and only if both of them are in the nearest-neighbor list.
- The *MinComponent* parameter is used to eliminate small connected components from the nearest-neighbor graph prior to clustering; in our case it has the default value which is five.
- The *CSType* parameter selects the method that is used to select the cluster to be bisected next; in our case it has a value of *Best* which selects the cluster whose bisection will optimize the value of the overall clustering criterion function the most.

[1] Full list of parameters with their values and explanation is available on the CLUTO manual (Karypis, 2003)

- The *AggloFrom* parameter tells the clustering program to compute a clustering by combining both the partitional and agglomerative methods. The value to this parameter specifies the number of clusters to be created first and agglomerative method merge some of them to obtain the desired number of clusters. In our case, since the value for this parameter is zero, clustering is made only using partitional method.
- The *AggloCRFun* parameter controls the criterion function that is used during the agglomeration. Since we don't specify the *AggloCRFun,* it has the same function as the partitional clustering method.
- The *NTrials* parameter selects the number of different clustering solutions to be computed by the various partitional algorithms; in our case it has a value of ten which is default.
- The *Niter* parameter selects the maximum number of refinement iterations to be performed within each clustering step; in our case it has a value of ten which is the default.

After displaying this information scluster takes some time to compute the clustering solutions. It executes for about 12 minutes and displays information regarding the quality of the overall clustering solution, the quality of each cluster and the time taken by different phases of the program, as shown in figure 9 below.

The figure shows the information for only the ten top clusters. The first line reports the overall value of the criterion function for the computed clustering solution, i.e. 1.07e+04 is the value of I_2 criterion function of the resulting clusters. The next numbers state how many of the named entities scluster program is able to cluster, i.e. all 44,450 named entities are clustered. The table with six columns then displays various statistics about resulting clusters ordered in an increasing *ISim* and *ESim* value. The *cid* (*cluster id*) column corresponds to the cluster number which runs from 0 to 1999; The *Size* column reports the number of named entities that belong to each cluster; The *ISim* (*Internal Similarity*) column reports the average similarity between the objects of the cluster; The *ISdev* (*displays the Internal Standard Deviations*) column reports the standard deviation of the average internal similarities. The *ESim (External Similarity)* column reports the average similarity of the objects of each cluster and the rest of the object and the *ESdev (external standard deviations*) column reports the standard deviation of the external similarities. Finally scluster displays Time taken for input/output operation, finding the clustering solution and reporting the result and terminates (Karypis, 2003).



```
Solution ---------------------------------------------

------------------------------------------------------
2000-way clustering: [I2=1.07e+04] [44450 of 44450]
------------------------------------------------------
cid  Size  ISim   ISdev   ESim   ESdev  |
------------------------------------------------------
  0    21 +0.571 +0.181 +0.000 +0.000 |
  1    10 +0.586 +0.106 +0.000 +0.000 |
  2     7 +0.607 +0.029 +0.000 +0.000 |
  3    13 +0.501 +0.081 +0.000 +0.000 |
  4    15 +0.450 +0.086 +0.000 +0.000 |
  5     3 +0.596 +0.020 +0.000 +0.000 |
  6     5 +0.490 +0.012 +0.000 +0.000 |
  7     4 +0.473 +0.058 +0.000 +0.000 |
  8     5 +0.414 +0.145 +0.000 +0.000 |
  9     2 +0.607 +0.000 +0.000 +0.000 |
------------------------------------------------------

Timing Information -----------------------------------
   I/O:                                   4.208 sec
   Clustering:                          712.104 sec
   Reporting:                             0.320 sec
******************************************************
```

Figure 9: scluster's output for 2000-way clustering of named entities

At this stage we have a clustering solution file discovered using partitional, nonhierarchical and similarity based clustering algorithms and the named entity list on a separate file. The clustering solution file contains a matrix with 44,450 rows and a single column. The i_{th} row of this file contains the cluster number of the named entity on the corresponding row of the named entity file. Appendix-A shows randomly selected named entity clusters.

2.2 Assigning Labels to the extracted clusters

2.2.1 Introduction to cluster labeling

In many applications of flat clustering and hierarchical clustering, particularly in analysis tasks and in user interfaces, human users interact with clusters. In such settings, we must label clusters so that users can see what a cluster is about. There are two basic categories of cluster labeling; *differential cluster labeling* and *cluster-internal labeling*. Differential cluster labeling selects cluster labels by comparing the distribution of terms in one cluster with that of other clusters. In particular, mutual information (MI) or, equivalently, information gain and the X^2-test will identify cluster labels that characterize on a cluster in contrast to other clusters. Cluster-internal labeling computes a label that solely depends on the cluster itself, not on other clusters. This method is efficient, but it fails to distinguish terms that are frequent in the collection as a whole from those that are frequent only in the cluster (Christopher D. Manning, Prabhakar Raghavan & Hinrich Schütze, 2009). In our cluster labeling experiment discussed below, we have used an approach presented by Van Durme and Pasca (2008) that incorporates a label selecting criterions of both differential and cluster-internal labeling methods.

2.2.2 Selected cluster labeling approach

Once we get the clusters of named entities the next step is to find a label for the clusters from a collection of parsed Wikipedia text and for this task we have selected the approach presented by Durme and Pasca, which tries to extract large numbers of semantic classes along with their corresponding instances (Benjamin Van Durme and M.Paşça, 2008). They introduce a TF×IDF (Term Frequency and Inverse Document Frequency) like method for deriving labeled classes of instances from unstructured, open domain text. For comparison, first we discuss TF-IDF measure.

TF-IDF is a statistical measure or weight often used in information retrieval and text mining to evaluate how important a word is to a document in a collection or corpus. Term frequency ($tf_{t,d}$) is the number of occurrence of a term in a document, where the subscript t refers term and d documents. The information that is captured by term frequency ($tf_{t,d}$) tells how salient a word is within a given document. The higher the term frequency the more likely it is that the word is a good description of the content of the document. Term frequency suffers from a problem of discriminating power; when considering all terms equally important to assess the relevancy of query, certain terms have little or no discriminating power. Document frequency (df_t) on the other hand can be interpreted as an indicator of informativeness. It is the number of documents in the collection that contain the term. A semantically focused word will often occur several times in a document if it occurs at all whereas semantically unfocussed words are spread out homogeneously all over documents. Inverse document frequency (idf_t) is used to scale down the term frequency of terms with high total number of occurrences in the collection. It is defined as shown in equation 3 below, where N is the total number of documents.

$$idf_t = log \frac{N}{df_t}$$

Equation 3: Inverse document frequency

One way to combine a word's term frequency ($tf_{t,d}$) and inverse document frequency (idf_t) into single weight is a TF-IDF measure shown in equation 4 below:

$$TF - IDF(t, d) = tf_{t,d} \times idf_t$$

Equation 4 TF-IDF measure (Christopher D. Manning, Prabhakar Raghavan & Hinrich Schütze, 2009)

Similarly, Durme and Pasca's labeling algorithm considers the clusters of named entities as documents and the class labels as document terms. Following the same analogy, the normalized term frequency is the number of instances in a cluster initially assigned a given label divided by the total number of instances in that cluster. In the algorithm shown on figure 10 below the parameter J acts as a TF constraint. Labels with wide distribution across clusters are less significant than those occurring more narrowly. The IDF counterpart of the algorithm, parameter K, is used to regulate this distribution directly in terms of the number of clusters covered (Benjamin Van Durme and M.Pasça, 2008).

In the algorithm shown below, P represents a set of *is-a* phrase pairs which suggest a mapping from instance to class labels, section 2.2.1 explains how these pairs are extracted from dependency based parsed Wikipedia files. The variable C represents partitioning of instances extracted by distributional similarity. These are the 2000 clusters of semantically related phrases we have extracted from Lin's thesaurus.

```
Given: I: set of instance phrases
       L: set of label phrases
       C: partitioning of I by distributional similarity
       $P \subseteq I \times L$: set of is-a phrase pairs
Returns: $P_{JK} \subseteq P$: set of filtered phrase pairs
Parameters: $J \in [0, 1]$: label freq. constraint (intra-cluster)
            $K \in N$: label freq. constraint (inter-cluster)
Algorithm:
Let $P_{JK} = \{\}$
For each semantic cluster $S \in C$:
    For each class label L, where $\exists I \in S_{s.t.} <I, L> \in P$:
        Let $S_L = \{I | I \in S, <I, L> \in P\}$
        Let $C_L = \{S' | S' \in C, \exists I \in S': <I, L> \in P\}$
        If $|S_L| > J \times |S|$:
            If $|C_L| < K$:
                Set $P_{JK} = P_{JK} \cup \{<I, L> | I \in S, <I, L> \in P\}$
```

Figure 10: Algorithm for extracting {instance, class label} pairs (Benjamin Van Durme and M.Pasça, 2008)

The algorithm begins by initializing the return value P_{JK} to an empty set. Then it will loop through each semantic cluster S contained in C. For each label L which labels at least one instance in S, the algorithm verifies whether the number of such instances paired with L is at least J of the size of S; this is intra-cluster constraint. If the label is viable based on the intra-cluster constraint, the next step will be verification of whether it is acceptable according to the inter-cluster constraint K. This constraint checks if the set of all clusters where at least one member of each cluster is paired with the label L is less than K. If the number of such clusters

$|C_L|$ is less than K, the label L is considered to be a good label for the supporting instances in S. Finally each instance in S that is paired with L is added to filtered phrase pairs collection P_{JK}, representing an assignment of those instances to the class specified by L (Benjamin Van Durme and M.Pasça, 2008).

Van Durme and Pasca have used this algorithm to extract a large collection of filtered {instance, label} pairs form unstructured, open domain text. We also follow the same approach, with minor modification on the algorithm as discussed in section 2.2.4 below, to assign labels to semantically similar clusters we extracted from Lin's thesaurus.

2.2.3 Extracting {Label, Instance} pairs (P)

To implement the above algorithm we need a large collection of {Label, Instance} pairs, together with the collection of semantically similar clusters of named entities. Such collections of pairs can be extracted using the approach presented by Hearst (Hearst, 1992). According to Hearst, when interpreting domain- independent text it is difficult to determine in advance what kind of information will be encountered and how it will be expressed. Instead of interpreting everything in the text in great detail, it is possible to search for specific lexical relations that are expressed in well-known ways. Based on this principle, she provides a method for the automatic acquisition of the hyponym lexical relation from unrestricted text.

There are many ways that the structure of a language can indicate the meanings of lexical items, but the difficulty lies in finding constructions that frequently and reliably indicate the relation of interest. However, Hearst identified a set of lexico-syntactic patterns that indicate the hyponymy relation and satisfy three conditions: they occur frequently and in many text genres; they always indicate the relation of interest and they can be recognized with little or no pre-encoded knowledge. Since the instances of these relations are appear in different form and those with the same pattern are only a small subset, she used many patterns that indicate the hyponym relations as shown in figure 11 below.

On the patterns shown, when a relation hyponym (NP_0, NP_1) is discovered the noun phrase is left as an atomic unit. This kind of discovery procedure can be a partial solution for a problem like noun phrase interpretation because at least part of the meaning of the phrase is indicated by the hyponymy relation (Hearst, 1992).

1. such NP as {NP ,}* {(or | and)} NP

   E.g. … works by such authors as Herrick, Goldsmith, and Shakespeare.

   - Hyponym ("author", "Herrick"),
   - Hyponym ("author", "Goldsmith"),
   - Hyponym ("author", "Shakespeare")
2. NP {, NP}* {,} or other NP

   E.g. Bruises, wounds, broken bones or other injuries…

   - Hyponym("bruise", "injury"),
   - Hyponym("wound", "injury"),
   - Hyponym("broken bone", "injury")

3. NP {, NP}* {,} and other NP

   E.g. … temples, treasuries, and other important civic buildings.

   - Hyponym ("temple", "civic building"),
   - Hyponym ("treasury", "civic building")

4. NP {,} including {NP ,}* {or | and } NP

   E.g. All common-law countries, including Canada and England …

   - Hyponym ("Canada", "common-law country"),
   - Hyponym ("England", "common-law country")
5. NP {,} especially {NP ,}* {or | and} NP

   E.g. most European countries, especially France, England, and Spain

   - Hyponym ("France", "European country"),
   - Hyponym ("Spain", "European country")

Figure 11: list of lexico-syntactic patterns that indicate the hyponymy relations (Hearst, 1992)

We have also followed a similar approach to extract label-instance pairs from parsed Wikipedia files. We used an unstructured text collection from English Wikipedia, August 2007 dump, which contains approximately 470M words. The collection is downloaded and parsed with the Stanford dependency parser[1] for part-whole, meronymy, relation learning from unstructured text by Ittoo and Bouma (2010). They use this corpus to obtain archetypal patterns, seed-sets, which characterize all different types of part-whole relations and extract similar relations by initializing a minimally-supervised information extraction algorithm with the seed-set.

The Stanford dependencies provide a represenation of grammatical relations between words in a sentence. They have been designed to be easily understood and effectively used by people who want to extract textual relations. Stanford dependecies are triplets: name of the relation, governor and dependent[2]. The parser output for the sentence, which is taken from Wikipedia, "*Sawyer hatches a plan to escape from his cage*" is shown in figure 12 below. The first part displays the dependency tree representation of the sentence including part of speech for the individual words. Then dependency triplets are shown, the relation followed by governor and dependent inside a bracket. For instance the relation, syntactic property, between "*hatches*" and "*Sawyer*", is "*Subject*" where "hatches" is governor and "Sawyer" is dependent.

```
(ROOT
  (S
    (NP (NNP Sawyer))
    (VP (VBZ hatches)
      (NP (DT a) (NN plan)
        (S
          (VP (TO to)
            (VP (VB escape)
              (PP (IN from)
                (NP (PRP$ his) (NN cage)))))))
    (. .)))

nsubj(hatches-2, Sawyer-1)
det(plan-4, a-3)
dobj(hatches-2, plan-4)
aux(escape-6, to-5)
infmod(plan-4, escape-6)
prep(escape-6, from-7)
poss(cage-9, his-8)
pobj(from-7, cage-9)
```

Figure 12: Stanford dependency for the sentence "*Sawyer hatches a plan to escape from his cage*"

[1] http://nlp.stanford.edu/software/lex-parser.shtml
[2] http://nlp.stanford.edu/software/stanford-dependencies.shtml

To extract the {Instance, Label} pairs from these parsed unstructured Wikipedia text collection, we first we use a Linux "grep" command to select out lines from the files that matches a pattern shown below:

```
grep "NN . *NNP"
```

Figure 13: grep Linux command used to extract lines with instance label pair

This command matches lines in a parsed Wikipedia file which contain a NN, singular or mass Noun, followed by any word sequence and zero or more NNP, proper noun. For instance, if we have a file containing the example phrases shown in figure 11 above as an argument for this expression, considering singular and mass nouns are tagged as NN and proper nouns are tagged as NNP, some of them would match the pattern. Some sample noun phrases taken from the parsed Wikipedia text collection and matched by the pattern are shown in figure 14 below:

```
1.  (NP (DT the) (NN rock) (NN band) (NNP Lynyrd) (NNP Skynyrd)))))
2.  (NP (JJ Visual) (NN artist) (NNP Sean) (NNP D'Anconia))
3.  (NP (NN football) (NN superstar) (NNP Diego) (NNP Maradona))
4.  (NP (JJ national) (NN hero) (NNP Jos) (IN de) (NNP San) (NNP Mart)))))
5.  (NP (DT The) (JJ American) (NN film) (NN director) (NNP John) (NNP Ford))
6.  (NP (VBG leading) (NN economist) (NNP Wynne) (NNP Godley))))
7.  (NP (NN writer) (NNP Stan) (NNP Lee))
8.  (NP (JJ Comics-artist) (NN legend) (NNP Wally) (NNP Wood))
9.  (NP (DT the) (NNP Fox) (NN television) (NN program) (NNP Prison) (NNP Break)))
10. (NP (DT the) (JJ British) (NN athlete) (NNP Paula) (NNP Radcliffe)))
```

Figure 14: sample noun phrases matched by extraction pattern

We extract 310 files containing noun phrases similar to those shown in the above figure and matched by the pattern shown in figure 13 from the parsed unstructured Wikipedia files. We then write a python regular expression script to filter out {Label, Instance} pairs from these files as the shown in figure 15 below.

```
Label = re.compile(r'\D*\(NN\s(\w+)\)\s\(NN\s(\w+)\)\D* |'
                   r'\D*\(VBG\s(\w+)\)\s\(NN\s(\w+)\)\D* |'
                   r'\D*\(JJ\s(\w+)\)\s\(NN\s(\w+)\)\s\(NN\s(\w+)\)\D* |'
                   r'\D*\(JJ\s(\w+)\)\s\(NN\s(\w+)\)\D* |'
                   r'\D*\(NN\s(\w+)\)\D*')
Instance = re.compile(r'\D*\(NNP\s(\w+)\)\s\(NNP\s(\w+)\D*')
```

Figure 15: python regular expression code fragment to filter {Label, Instance} pairs

The above python regular expression extracts sequence of singular or mass nouns (NN, NN), gerund followed by singular or mass noun (VBG, NN), adjective followed by two consecutive singular or mass noun (JJ, NN, NN), adjective followed by singular or mass noun (JJ, NN) and only singular or mass noun (NN) as Labels. It then extracts two consecutive proper nouns (NNP, NNP) as Instances. There are also a few instance names in the collection constructed from one or three consecutive proper nouns (NNP). However, since their frequency of occurrence in the collection is low, the pattern they appear in a sentence is complicated and the error rate for extracting them is high we only extract instances that are constructed from two consecutive proper nouns.

Running the script, providing with the 310 Wikipedia files as an argument, returns ~305,500 label-Instance pairs. For instance, the script extracted the label-instance pairs listed in table 2 below from the lines of noun phrases extracted using *grep* command as shown in figure 13.

No	Instance	Label
1.	Lynyrd Skynyrd	rock band
2.	Sean D	Visual artist
3.	Diego Maradona	football superstar
4.	San Mart	national hero
5.	John Ford	film director
6.	Wynne Godley	leading economist
7.	Stan Lee	Writer
8.	Wally Wood	comics legend
9.	Prison Break	television program
10.	Paula Radcliffe	British athlete

Table 2: List of sample {Instance, Label} pairs

2.2.4 Implementing the labeling algorithm

Durme and Pasca used the algorithm presented in figure 10 for extracting a large number of semantic classes along with their corresponding instances using a collection of approximately

100 million Web documents (Benjamin Van Durme and M.Pasça, 2008). On the other hand, our aim is to utilize the same algorithm to assign a label for semantically similar clusters of named entities that we have already extracted using labels from a much smaller Wikipedia document collection. With this intension, we modify the algorithm to make it suitable for our requirement as shown in figure 16 below and implement it using python programming language.

```
Given:  C: set of semantically similar clusters extracted from Lin's thesaurus
        $P \subseteq I \times L$: set of is-a phrase pairs extracted from parsed Wikipedia collection
Returns: $C_L \subseteq C$: set of Labeled clusters
Parameters: $J = 0.1$: label freq. constraint (intra-cluster)
            $K = 20$: label freq. constraint (inter-cluster)
Algorithm:
Let $C_L = \{\}$
Let Labeled_NEs = {}
For each semantic cluster $S \in C$:
    For each name $N \in S$:
        For each $I \times L \in P$:
            If $N = I$:
                Add $I \times L$ to Labeled_NEs
    If |labeled_NEs| > 0:
        For each $I \times L \in$ Labeled_NEs:
            $M_1$ = most frequent (L);
            $F_1 = |M_1|$
            $M_2$ = second most frequent (L):
            $F_2 = |M_2|$
        For all $I \times L \in P$
            If L.count($M_1$) < K and $F_1 > J \times |S|$:
                Add $M_1 \times C$ to $C_L$
            If L.count($M_1$) > K and $F_1 < J \times |S|$:
                If L.count($M_2$) < K and $F_2 > J \times |S|$:
                    Add $M_2 \times C$ to $C_L$
            Otherwise add UKNOWN $\times$ C to $C_L$
    Otherwise add UNKNOWN $\times$ C to $C_L$
```

Figure 16: Modified algorithm for labeling clusters of semantically similar named entities

Our modified algorithm starts by initializing the labeled clusters, $C_{L,}$ and labels for names in each cluster, *Labeled_NEs,* to empty sets. It then loops through the names in each cluster and checks if it is available in the label-instance pairs extracted from Wikipedia text. If it matches with an instance (*I*) from the pairs, then the matched L × I pair is added to *Labeled_NEs*. After all the names in the cluster have been checked and the labels set is not empty, it takes the first most frequent label assigned for the names in the clusters and checks whether it satisfies the intra-cluster and inter-cluster constraints or not. If it satisfies, the label and cluster ($M_1 \times C$) is added to labeled cluster set (C_L). If the first most frequent label doesn't satisfy the constraints it checks for the second most frequent label. If the second most frequent label satisfies the constraints, the label and cluster ($M_2 \times C$) is added to labeled cluster set (C_L). If both of the labels don't satisfy the constraints it assigns UKNOWN label for the cluster and adds it to labeled clusters set (C_L). It also assigns an UNKNOWN label if none of the names in the cluster has a match in label-instance pairs, i.e. |labels| = 0.

2.2.5 Results obtained

We run the python implementation of the above algorithm providing ~305,500 instance-label pairs (P) that are taken from parsed Wikipedia files and 2,000 clusters containing a total of 44,450 named entities as input with two different settings. First we take only the first most frequent label from the labels assigned for the names in the cluster and get 850 labeled clusters out of 2,000. We then take the second most frequent label as an alternative when the first most frequent cluster fails the two constraints and get 924 labeled clusters. Table 3 below shows an instance cluster and how the first and second most frequent instances are selected.

Named entities in the cluster (N)	**Label (L) where I × L ∈ P and I = N**
Alessandro Del Piero	Not found
Christian Vieri	[footballer, *player*, **striker**, Italian star, international striker, half time]
Filippo Inzaghi	[Italian striker]
Gabriel Batistuta	[Argentinian legend]
George Weah	[**striker**, football hero, soccer star]
Gianfranco Zola	[**striker**]
Gianluca Vialli	[manager, **striker**, new manager, Chelsea boss]
Pierluigi Casiraghi	[*player*]
Roberto Baggio	[fan favorite]
Zinedine Zidane	[playmaker, teammate, French captain, French footballer, French midfielder, French star, former footballer, former teammate, French footballer, football player, football star, sign playmaker, soccer player, team captain]

Table 3: Instance cluster with labels for the name entities (N) obtained from instance label pairs (P)

On the above table the number of named entities in the cluster, |S|, is 10. According to the modified algorithm, the first most frequent label, M_1, is "*striker*" with a frequency of, F_1, 4 and the total number of times it is used as a label in the overall cluster collection, *L.count(M1)*, is also 4. The second most frequent label, M_2, is "*player*" with a frequency of, F2, 2 and the total number of times it is used as a label in the overall cluster collection, *L.count(M_2)*, is also 2. According to intra-cluster constraint, i.e. $(|F_1| = 4) > (|S|=10) \times (J = 0.1)$, M_1 is viable label for the cluster. According to inter-cluster constraint also, i.e. *(L.count (M_1) =4) < (K =30),* M_1 is a viable collection; therefore "striker" × M_1 will be added to labeled clusters collection (C_L).

Chapter 3

3 Evaluation, conclusion and future work

In this chapter we discuss how we evaluate the result obtained, draw conclusions and describe what can be done for the future based on the result obtained. The first part discusses the approaches followed to evaluate clustering tasks in different studies, the types of clustering evaluation metrics and report the result of our evaluation on random samples that are subjectively judged. In the second part, we try to summarize all the tasks we done, point out which parts should be improved for a better result and draw conclusion. In the last part, we discuss applications that use the output of this research and what we will do for the future to utilize the labeled clusters we harvest in this study.

3.1 Evaluation

In our study, evaluation involves measuring the results obtained from both tasks; the clustering of named entities based on their semantic similarity and the class label assignment for the clusters from Wikipedia document collection. Rosell stated that it is very hard to perform a reliable evaluation of clustering results, partially since what is a good partition of a text set is very subjective. It depends on the text set, the purpose of the clustering and if it is constructed as an intermediate step in further processing or if it is to be used directly by a human (Rosell, 2009). Named entity clustering evaluation is also dependent on the above constraints and hard to make it perfectly reliable.

Most researchers evaluate clustering and semantic relations by conducting an additional task on the outputs of their experiment, i.e. task based evaluation, or manually selecting random instances and making subjective judgments, i.e. manual evaluation. In his study for finding categorized named entities from unstructured text, Pasca did an incipient evaluation by computing the precision over the top 50 instance names of 20 randomly selected compound name categories (Pasça, 2004). Durme and Pasca determine the quality of results in the class labeling task by manually evaluating one hundred randomly selected instances and authors' common agreement on the usefulness of each term. They have also done a separate set of experiments that uses the extracted classes as input data for the task of attributes in their task based evaluation (Benjamin Van Durme and M.Pasça, 2008).

In this study, we collect subjective judgments for semantic relatedness of named entities in a cluster and the accuracy of the labels assigned for the clusters by taking a small portion of randomly selected clusters. For semantic relatedness of named entities in a cluster, we have set a standard on the portion of similar named entities in a given cluster and a category will be assigned based on this standard. We have also used the external cluster quality measure called purity to evaluate quality of the clusters. For the accuracy of a label assigned to a cluster there is no guidance and the decision is made based on the evaluator's subjective judgment.

3.1.1 Types of Clustering evaluation

There are two types of text clustering quality measures, *internal* and *external*. Internal measures evaluate the result by looking at how cohesive and how well separated the clusters are. They are suitable for comparisons of different clusters of the same text set, if they are produced using the same representation (Rosell, 2009). Good scores on an internal criterion do not necessarily translate into good effectiveness in an application. An alternative to internal criteria is direct evaluation in the application of interest. For search result clustering, we may want to measure the time it takes users to find an answer with different clustering algorithms. This is the most direct evaluation, but it is expensive, especially if large user studies are necessary (Christopher D. Manning, Prabhakar Raghavan & Hinrich Schütze, 2009).

External quality measures take advantage of some external context. Some of them assess the quality of clustering through other tasks in which it plays a part; like a dimensionality reduction experiment or as an aid for a search task. Other external measures compare the clustering to another partition, such as those created by manual categorization (Rosell, 2009). There are four external criteria of clustering quality; *purity*, *Normalized mutual information*, *Rand index* and *F measure*. These text clustering quality measures are also applicable for named entity clustering evaluation and we briefly discuss them below in the context of it.

Purity is a simple and transparent evaluation measure. To compute this external quality measure, each cluster is assigned to the class which is most frequent in the cluster, and then the accuracy of this assignment is measured by counting the number of correctly assigned named entities and dividing by *N*. It is expressed formally as:

$$purity\ (\Omega, \mathbb{C}) = \frac{1}{N}\sum_{k} \max_{j} |w_k \cap c_j|$$

Equation 5: Purity, external cluster quality measure (Christopher D. Manning, Prabhakar Raghavan & Hinrich Schütze, 2009)

In the above equation, $\Omega = \{w_1, w_2, \ldots, w_k\}$ is the set of clusters and $\mathbb{C} = \{c_1, c_2, \ldots, c_k\}$ is the set of classes, w_k is interpreted as the set of named entities in w_k and c_j as the set of named entities in c_j. High purity is easy to achieve when the number of clusters is large, in particular, purity is 1 if each named entity gets its own cluster. Thus, we cannot use purity to trade off the quality of the clustering against the number of clusters.

Normalized mutual information, NMI, is a measure that allows trading off the quality of the clustering against the number of clusters. It is expressed formally as:

$$\mathrm{NMI}(\Omega, \mathbb{C}) = \frac{I(\Omega; \mathbb{C})}{[\mathrm{H}(\Omega) + \mathrm{H}(\mathbb{C})]/2}$$

Equation 6: Normalized mutual information, external cluster quality measure (Christopher D. Manning, Prabhakar Raghavan & Hinrich Schütze, 2009)

In equation 6 above I (Ω; ℂ) measures the amount of information by which our knowledge about the classes increases when we are told what the clusters are. Mutual information, I, and entropy, H, are mathematically expressed as shown in equation 7 & 8 below.

$$I\ (\Omega; \mathbb{C}) = \sum_k \sum_j P\left(w_k\ \cap\ c_j\right) \log \frac{P(w_k \cap c_j)}{P(w_k)\, P(c_j)} = \sum_k \sum_j \frac{|w_k \cap c_j|}{N} \log \frac{N|w_k \cap c_j|}{|w_k||c_j|}$$

Equation 7: Mutual Information of named entities in cluster, Ω, and class, ℂ (Christopher D. Manning, Prabhakar Raghavan & Hinrich Schütze, 2009)

$$H(\Omega) = -\sum_k P(w_k) \log P(w_k) = -\sum_k \frac{|w_k|}{N} \log \frac{|w_k|}{N}$$

Equation 8: Entropy of named entities in a cluster (Christopher D. Manning, Prabhakar Raghavan & Hinrich Schütze, 2009)

Rand index, RI, is an alternative to normalized mutual information. It views cluster measuring as a series of decisions, one for each of the *N (N - 1) / 2* pairs of named entities in the collection. We want to assign two named entities to the same cluster if and only if they are similar. A true positive (TP) decision assigns two similar named entities to the same cluster; a true negative (TN) decision assigns two dissimilar named entities to different clusters. There are two types of errors we can commit. A false positive (FP) decision assigns two dissimilar named entities to the same cluster. A false negative (FN) decision assigns two similar named entities to different clusters. The RI measures the percentage of decisions that are correct and expressed as:

$$RI = \frac{TP + TN}{TP + FP + FN + TN}$$

Equation 9: Rand Index external cluster quality measure (Christopher D. Manning, Prabhakar Raghavan & Hinrich Schütze, 2009)

The RI gives equal weight to false positives and false negatives. Separating similar named entities is sometimes worse than putting pairs of dissimilar named entities in the same cluster. We can use the F measure to penalize false negatives more strongly than false positives by selecting a value β>1, thus giving more weight to recall.

$$P = \frac{TP}{TP + FP} \quad R = \frac{TP}{TP + FN} \quad F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

Equation 10: Precision, Recall and F-measure; external cluster quality measure (Christopher D. Manning, Prabhakar Raghavan & Hinrich Schütze, 2009)

3.1.2 Evaluating the results obtained

As we have discussed above, to apply internal and external cluster quality measuring criterions it needs a deep analysis on the output of the cluster. To make a direct evaluation as an alternative to internal criteria, we need to design an application which can use the labeled clusters and make

large users studies. To apply the external measuring criterions, we need evaluators who can critically observe the clusters and decide how many of the named entities in a given cluster are semantically similar and how many of them are dissimilar. However, most of the named entities in the cluster are not commonly known and the evaluators cannot easily make the decision.
Considering the allocated time for this study and the difficulty of the situation, we decide to make the evaluation on 100 randomly selected clusters. We select these 100 random representative samples using python's random module as shown in figure 17 below.

```
for index in range (0,100):
     index=random.randint(0,923);
    random_file.write("("+cluster[index]);
```

Figure 17: segment of python code for selecting random clusters

The above segment of python code generates 100 random numbers in the range 0-923, i.e. the number of labeled clusters. It then selects 100 clusters from the array of all the clusters using the generated number as index and write it on a file.
We provide these randomly selected clusters for two evaluators to decide how many of the named entities in a give cluster are semantically similar. We advise them to refer to Wikipedia for the unknown names. To simplify the task we make five based scales and tell the evaluators to select one of them for each cluster. The scales are; "*Perfect cluster*" if 100% of the names in the cluster are similar, "*Very good cluster*" if 99-85% of the names are similar, "*Good*" if 84-70% of the names are similar, "*Fair cluster*" if 69-50% of the names are similar and "Noise cluster" if <50% of the names are similar. We design an interactive evaluation website[1] which will calculate the percentage of similar cluster when the evaluator checks out names that are not similar with the majority of the cluster. We also keep track of the number of named entities the evaluators believed to fall in the same semantic category to calculate the purity of the clusters. As we have discussed in the previous section, purity doesn't allow trading off the quality of the clustering against the number of clusters and is not appropriate measure for large number of clusters. But in our case, since we have few clusters, specifically 100, this trade of is not a question.
On the other hand, there is no standard quality measure for the labels assigned for the clusters and the only option we have is evaluators' subjective judgment. As was the case for the clusters, we make five based scales (*Perfect*, *Very good*, *good*, *medium* and *wrong*) to guide the evaluators. These scales are to be assigned based on the quality of the labels in representing the majority of semantically similar named entities in the clusters.

[1] The evaluation website is available at http://siegfried.let.rug.nl/s1951866/evaluation.php

Within the allocated days the evaluators evaluated the first 86 clusters out of the randomly selected 100 clusters. The result for both, the clusters and the labels quality, are reported in table 4 and 5 below and sample evaluations are also shown in Appendix B, at the end of the document.

Scale	Scale criterion	Number of clusters	Percentage
Perfect cluster	100% similarity	16	18.6%
Very good cluster	99-85% similarity	33	38.4%
Good cluster	84-70% similarity	23	26.7%
Fair cluster	69-50% similarity	5	5.8%
Noise cluster	<50% similarity	9	10.5%
Total		**86**	**100%**

Table 4: Cluster quality evaluation result

Scale	Number of labels	Percentage
Perfect label	16	18.6%
Very good label	14	16.3%
Good label	10	11.6%
Medium label	7	8.1%
Wrong label	39	45.4%
Total	**86**	**100%**

Table 5: Label quality evaluation result

To check the inter-evaluators agreement, we took 20 of the randomly selected clusters and let the two evaluators evaluate them. The result shows 14 of the cluster quality and 13 of label quality evaluation results of the two evaluators are overlapping. For most of non-overlapping evaluations also there is only one scale difference; for instance, if evaluator one assigns “Very good cluster” scale evaluator two assigns “Good cluster” scale and so on. We also calculate the Kappa statistic to measure the agreement between the two evaluators. Kappa statistic is a common measure for agreement between judges that is designed for categorical judgments and corrects a simple agreement rate for the rate of chance agreement (Christopher D. Manning, Prabhakar Raghavan & Hinrich Schütze, 2009). The mathematical expression is shown in equation 11 below.

$$kappa = \frac{P(A) - P(E)}{1 - P(E)}$$

Equation 11: kappa statistic

In the above equation, P(A) is the times the judges agreed, and P(E) is the proportion of the times they would be expected to agree by chance. For the two evaluators, the clustering evaluation and label evaluation agreement table is shown in tables 6 & 7 below. To compute the kappa statistic, we first need to calculate the observed agreement and the expected rates by which the two evaluators to agree by chance, i.e. the probability of agreement considering them as totally independent. For cluster evaluation these values are:

Observed Agreement P(A) = (2 + 7 + 2 + 1 + 2)/20 = 0.7

Chance agreement P(E) = (0.1*0.15) + (0.35*0.5) + (0.3*0.1) + (0.1*0.15) + (0.15*0.1) = 0.25

Then the value of Kappa is defined as:

$$kappa = \frac{0.7 - 0.25}{1 - 0.25} = 0.6$$

Evaluator one								
		Perfect	Very Good	Good	Fair	Noise	Total	Proportion
Evaluator Two	Perfect	**2**	0	1	0	0	3	0.15
	Very Good	0	**7**	2	1	0	10	0.5
	Good	0	0	**2**	0	0	2	0.1
	Fair	0	0	1	**1**	1	3	0.15
	Noise	0	0	0	0	**2**	2	0.1
	Total	2	7	6	2	3	20	
	Proportion	0.1	0.35	0.3	0.1	0.15	0.2	

Table 6: Cluster evaluation agreement

Similarly, for the labels evaluation:

Observed Agreement P(A) = (1 + 2 + 1 + 0 + 9)/20 = 0.65

Chance agreement P(E) = (0.1*0.2) + (0.15*0.25) + (0.15*0.05) + (0.15*0.0) + (0.45*0.5) = 0.29

$$kappa = \frac{0.65 - 0.29}{1 - 0.29} = 0.5$$

Different people have different interpretation of Kappa value. Landis and Koch have proposed the following as standards for strength of agreement: ≤ 0 *poor*, 0.1- 0.2 = *slight*, 0.21 – 0.4 = *fair*, 0.41 – 0.6 = *moderate*, 0.61 – 0.80 = *substantial* and 0.81 – 1 = *almost perfect* (J. Richard Landis and Gary G. Koch, Mar., 1977). Base on this standard our evaluators' agreement for both tasks is moderate.

		Evaluator one						
		Perfect	Very Good	Good	Medium	Wrong	Total	Proportion
Evaluator Two	Perfect	**1**	1	1	1	0	4	0.2
	Very Good	1	**2**	1	1	0	5	0.25
	Good	0	0	**1**	0	0	1	0.05
	Medium	0	0	0	**0**	0	0	0.0
	Wrong	0	0	0	1	**9**	10	0.5
	Total	2	3	3	3	9	20	
	Proportion	0.1	0.15	0.15	0.15	0.45	0.2	

Table 7: Label evaluation agreement

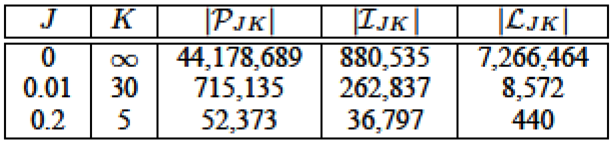
Finally, we estimate the purity of the clusters based on the randomly selected samples. As we have discussed in section 3.1.1 purity is measured by counting the number of correctly assigned documents to the class which is most frequent in the cluster and dividing by the total number of documents, N. In our case the total number of named entities in the random samples, N, is 1,687 and the classes which are most frequent in the clusters are the classes in which majority of semantically similar named entities fall in each cluster. The total number of correctly assigned named entities to these majority classes, i.e.$\sum_k \max_j |w_k \cap c_j|$, is 1,287. Then the purity of the clusters will be:

$$purity\ (\Omega, \mathbb{C}) = \frac{1}{N}\sum_k \max_j |w_k \cap c_j| = \frac{1}{1{,}687} \times 1287 = 0.76$$

The subjective evaluation as well as the purity measure results based on 86 clusters, 9.3% of the total clusters, shows the quality of the clusters is much better than the average. The subjective evaluation for the labels quality on the other hand shows 45.4% of the randomly selected clusters assigned wrong labels. Generally, these results indicate a good quality of clusters' semantic relatedness and poor quality of the labeling task. In the remaining sections we draw a conclusion based on the results we obtain and the evaluation outcome and indicate what can be done for the future.

3.2 Comparison with a related work

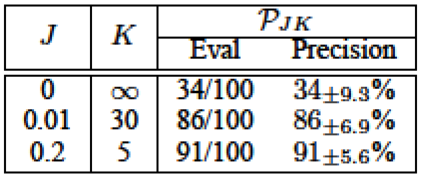
Van Durme and Pasca followed the same approach and extracted large number of labeled classes of instances using a range of parameter settings (Benjamin Van Durme and M.Pasça, 2008). They used wide (J = 0.01, K = 30) and narrow (J = 0.2, K = 5) parameter settings. Table 8 below shows that as the value of J is lowered and K is increased the resultant instance-label pairs $|P_{JK}|$, instances $|I_{JK}|$ and class labels (L_{JK}) increase. We can also see that for the same value of a parameter J = 0.01, they have extracted and used more than twice as much as the {label, instance} pairs we have used (i.e. 305,500) to get the class labels $|L_{JK}|$.



J	K	$\lvert\mathcal{P}_{JK}\rvert$	$\lvert\mathcal{I}_{JK}\rvert$	$\lvert\mathcal{L}_{JK}\rvert$
0	∞	44,178,689	880,535	7,266,464
0.01	30	715,135	262,837	8,572
0.2	5	52,373	36,797	440

Table 8: The number of {instance, label} pairs $|P_{JK}|$, instances $|I_{JK}|$ and class labels $|L_{JK}|$ extracted with different parameter settings (Benjamin Van Durme and M.Pasça, 2008).

They have also manually evaluated the extracted pairs of {label, instance} by randomly selecting one hundred sets. The result in table 9 below shows their approach is successful in separating high quality {class, instance} pairs from the lower average quality input data. It has also shown that the quality of the extracted pairs significantly increases as the value of J gets bigger; however the number of extracted classes will get lower as shown on table 8. The best option to retain the high quality and to obtain large number of pairs is at the J = 0.01.



J	K	$\mathcal{P}_{JK}$	
		Eval	Precision
0	∞	34/100	$34_{\pm 9.3}\%$
0.01	30	86/100	$86_{\pm 6.9}\%$
0.2	5	91/100	$91_{\pm 5.6}\%$

Table 9: Quality of pairs with different parameter settings (Benjamin Van Durme and M.Pasça, 2008)

Following similar approach and parameter setting (i.e. J = 0.01 and K = 20) for the cluster labeling task, we are able to obtain 55% acceptable labels with different degree of accuracy.

3.3 Conclusion

In this study we have tried automatically construct clusters of semantically similar ~44,000 named entities taken from freely available thesaurus. By making a rough estimation that an individual cluster will have 20 named entities, we decide the number of cluster to be 2,000. We then assign a label for 924 of the clusters that can express what the cluster is about using a simple algorithm that uses label-instance pairs taken from unstructured Wikipedia texts. Finally we consider two options to evaluate the result; either to develop an application which can use the labeled clusters as input and measure the quality of the application or conduct a subjective evaluation on randomly selected representative clusters. Taking the allocated time for the study into account, we select the second option. The evaluation result shows that the randomly selected 86 clusters have a purity value of 0.76 where as 45.4% of the assigned labels are inappropriate.

Following the same approach and similar parameter settings, Durme and Pasça are able to derive labeled classes of instances from unstructured open domain text with the precision of 86% (Benjamin Van Durme and M.Pasça, 2008). For the low quality of the assigned labels we predict two major reasons. First we have used much smaller Wikipedia document collection to extract the label instance pairs that will be used by the labeling algorithm. We have used English Wikipedia text which contains only ~470M words. If we compare the document collection with that of Durme and Pasças' experiment, that uses ~100 million web documents for similar experiment, we can easily tell the reason for the poor quality. Second there is a domain mismatch between the named entity collections taken from the thesaurus and the label-instance pairs we used to assign the labels. The thesaurus is automatically constructed by Dekang Lin in 1997 using a word similarity measure based on the distributional pattern of words. On the other hand, the unstructured text collection we used to extract the label-instance paris is from English Wikipedia, August 2007 dump. Durme and Pasças used the same collection of web documents to construct the clusters of related terms and initial label-instance pairs for their task.

We hope that it is possible to get much better result if it is possible to get more parsed web document collections and there is no domain mismatch between the named entities in the cluster and the source of label-instance pairs. We also believe that the output obtained from this study can be used for information retrieval application as we discuss in the further work below.

3.4 Future work

The main aim of this research is to automatically harvest clusters of labeled lexicon resources that can be used for the improvement of information retrieval application. Even if they are not accurate and large enough for the intended task, we managed to obtain these resources. We also believe that it is possible to refine the result and get more clusters with better quality. For the future, we are planning two studies based on the output we have. First we will try to use a large collection of web documents, not only Wikipedia, to extract a label-instance pairs and a more recent and large thesaurus to improve the quality and the quantity of the lexicon resource. Second we will use the clusters of named entities for automatic clustering and categorization of web pages with the aim of developing a web recommender application in which users can navigate by means of category labels as well as by looking for similar pages.

A good example of recommender systems is a system which provides advice to users about items they might wish to purchase or examine. Recommendations made by such systems can help users navigate through large information spaces of product descriptions, news articles or other items. There are three approaches to develop recommender systems; *collaborative- or social-filtering approach*, *content-based approach* and *knowledge-based approach*. Collaborativefiltering approach aggregate data about customers' purchasing habits or preferences, and make recommendations to other users based on similarity in overall purchasing patterns. Systems that are developed using content-based approach are classifier systems derived from machine learning research. These systems use supervised machine learning to induce a classifier that can discriminate between items likely to be of interest to the user and those likely to be uninteresting. Knowledge-based approaches use knowledge about users and products to generating a recommendation, reasoning about what products meet the user's requirements (Burke, 2001).

We are planning to use our labeled clusters of named entities for developing a content based recommendation system as a start up feature for the machine learning and automatic labeling and clustering of similar web pages.

4 Appendix

4.1 Appendix A (Randomly selected Named Entity clusters)

Cluster Id	List of Named Entities	Labels assigned Automatically
14	Blackburn Rovers Bradford City Coventry City Derby County Leeds United Leicester City Newcastle United Tottenham Hotspur Watford West Ham United	Football Clubs
33	Andhra Pradesh Assam Bihar Gujarat Haryana Himachal Pradesh Karnataka Kerala Madhya Pradesh Maharashtra Orissa Pradesh Rajasthan Tamil Nadu Uttar Pradesh West Bengal	Indian state
43	Al Ahram Al Hayat Alam Al Yom Bangkok Post Business Day Daily Nation Daily Star Dominion East African East African Standard Egyptian Gazette Ethiopian Herald Financial Mail Gulf News Iran News Jakarta Post Jordan Times Kenya Times Khaleej Times Malaya Moscow Times New Light New Nigerian	Newspaper

	New Vision Noticias Post Express Sowetan Tehran Times The Monitor The New Vision Vietnam News Wenhui Daily Zambia Daily Mail al Hayat newsmagazine	
60	Adam Vinatieri Gary Anderson John Hall Morten Andersen Sebastian Janikowski Vinatieri	kicker
88	Angelo Peruzzi Brad Friedel Briana Scurry David Seaman Drazen Ladic Fabien Barthez Gianluca Pagliuca Jorge Campos Jose Luis Chilavert Kasey Keller Kevin Hartman Mark Bosnich Peter Schmeichel Schmeichel Taffarel Tom Presthus Tony Meola Walter Zenga Zach Thornton	goalkeeper
163	ABN AMRO Banc BancBoston Robertson Stephens Bear Stearns CS First Boston Credit Suisse First Boston DRI Dean Witter Donaldson Lufkin AMP Jenrette Donaldson Edward Jones Goldman Sachs AMP Co Goldman Sachs Goldman Sachs AMP Co Hambrecht AMP Quist ING Barings Investors Service JP Morgan Jenrette	Bank

	Lehman Brothers Merrill Lynch Merrill Lynch AMP Co Morgan Stanley Dean Witter Morgan Stanley Dean Witter AMP Co NationsBanc Montgomery Securities Paine Webber PaineWebber Prudential Securities Putnam Robertson Stephens Royal Bank of Scotland Salomon Smith Barney Smith Barney Standard AMP Poor Warburg Dillon Read	
229	Bernard Barmasai Colin Jackson Dan O Brien Daniel Komen De Bruin Emma George Florence Griffith Joyner Greg Foster Haile Gebrselassie Heike Drechsler Hicham El Guerrouj Hiroyasu Shimizu Irina Privalova Jackie Joyner Kersee Javier Sotomayor Jeff Hartwig Jeremy Wotherspoon Jonathan Edwards Khalid Khannouchi Kim Batten Maria Mutola Noureddine Morceli Randy Barnes Sergei Bubka Stacy Dragila Steve Backley	runner
237	Barbara Boxer Bob Graham Bob Kerrey Charles E Schumer Charles Schumer Christopher Dodd Dale Bumpers Daniel Patrick Moynihan Dianne Feinstein Edward Kennedy Edward M Kennedy Evan Bayh Frank Lautenberg Gary Hart	Senator

	John Breaux John Edwards John Glenn John Kerry Joseph I Lieberman Joseph Lieberman Patrick Leahy Paul Tsongas Paul Wellstone	

4.2 Appendix B (Sample evaluation results)

Cluster	Label	Cluster Evaluation result	Label Evaluation result
Caritas Communist Youth League of China Oxfam Project Hope World Vision	international charity	Good Cluster	Perfect label
Charles Stenholm Henry Bonilla Jim Turner Lloyd Doggett Martin Frost Ron Paul Sheila Jackson Lee	Republican congressman	Perfect Cluster	Perfect Label
A J Foyt Arie Luyendyk Billy Boat Buddy Lazier Greg Ray Jeff Ward Kenny Brack Luyendyk Sam Schmidt Scott Goodyear Scott Sharp	car driver	Good Cluster	Good Label
Abdulsalam Abubakar Abdulsalami Abubakar Charles de Gaulle Francisco Franco Manuel Noriega Omar Torrijos Pervaiz Musharraf Pervez Musharraf Robert Guei Sani Abacha	Dictator	Very Good Cluster	Medium Label

mane			
Russian bank building society clearing house clothing chain corporates credit union finance company insurance firm investment house merchant bank savings bank trust company	grunge band	Perfect Cluster	Wrong Label

5 Bibliography

Ashwin Ittoo, Gosse Bouma. (2010). On learning parts and wholes: Do not Mix your Seeds. *ACL2010* .

Baker, L. (n.d.). *Vivisimo Search Engine Clusty Offers More Targeted Search*. Retrieved from search engine journal: http://www.searchenginejournal.com/vivisimo-search-engine-clusty-offers-more-targeted-search/917/

Benjamin Van Durme and M.Pasça. (2008). Finding Cars, Goddesses and Enzymes: Parametrizable Acquisition of Labeled Instances for Open-Domain Information Extraction. *Twenty-Third AAAI conference on Artificial intelligence,* (pp. 1243-1248).

Burke, R. (2001). Knowledge-based recommender systems. In *Encyclopedia of Library and Information Science, volume 69* (pp. 180-200).

Christopher D. Manning and Hinrich Schutze. (1999). In *Foundation of statistical Natural Languge processing* (pp. 495-528). Cambridge, Massachusetts: The MIT press.

Christopher D. Manning, Prabhakar Raghavan & Hinrich Schütze. (2009). *An Introduction to Information Retrieval.* Cambridge, England: Cambridge University Press.

Dongen, Stijn Marinus van. (2000). *Graph clustering by flow simulation.* Universiteit Utrecht.

Dragomir R. Radev, Weiguo Fan and Zhu Zhang. (2001). WebInEssence: A Personalized Web-Based Multi-Document Summarization and Recommendation System. *In NAACL 2001 Workshop on Automatic Summarization,* (pp. 79-88).

Hearst, M. A. (1992). Automatic acquisition of hyponyms from large text-corpora, Volume 2. *International Conference On Computational Linguistics* (pp. 539 - 545). Morristown, NJ, USA: Association for Computational Linguistics.

J. Richard Landis and Gary G. Koch. (Mar., 1977). The Measurement of Observer Agreement for Categorical Data. *Biometrics, Vol. 33, No. 1* , 159-174.

J.Jansen, B. (2000, October). The effect of query complexity on Web searching results. *Information Research, Vol.6 No.1* .

J.O.Ramsey and David Wiley. (1978). Applied Psychological Measurement . West Publishing Co.

Karypis, G. (2003). *CLUTO a clustering toolkit.* Minneapolis.

Lin, D. (1998). Automatic retrieval and clustering of similar words. *Proceedings of the 17th International Conference on Computational Linguistics and the 36th Annual Meeting of the Association for Computational Linguistics COLING-ACL '98*, (pp. 768–774). Montreal, Canada.

Lin, D. (1997). Using syntactic dependency as local. *ACL/EACL-97*, (pp. 64–71). Madrid, Spain.

Lin,D. and Pantel,P. (2002). Concept discovery from text . *19th International Conference on Computational Linguistics (COLING-02)*, (pp. 1-7).

M.Pasça and Benjamin Van Durme. (2008). Weakly-Supervised Acquisition of Open-Domain Classes and Class Attributes from Web Documents and Query Logs. *ACL* , 19–27.

Michael Strube and Simone Paolo Ponzetto. (2006). WikiRelate! Computing semantic relatedness using Wikipedia. *AAAI, volume21* (pp. 1419-1424). Menlo Park, CA: MIT Press.

Pantel, P. and D.Ravichandran. (2004). Automatically labeling semantic classes. *HLT/NAACL, volume4*, (pp. 321-328).

Pasça, M. (2004). Acquisition of categorized named entities for web search. *CIKM*, (pp. 137-145). Washington, DC.

Rosell, M. (2009). *Swedish Text Representation and Clustering Results Unraveled, Doctoral Thesis.* Stockholm, Sweden.

Wang, R.C. and W.W. Cohen. (2007). Language-independent set expansion of named entities using the web. *IEEE International Conference on Data Mining (ICDM), volume 6.*

Ying Zhao and George Karypis. (2003). *Clustering in Life Sciences.* Minneapolis: Humana Press.