NTNU
Innovation and Creativity

Optimal Information Retrieval Model for Molecular Biology Information

Jon Rune Paulsen

Master of Science in Informatics
Submission date: May 2007
Supervisor: Herindrasana Ramampiaro, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Table Of Contents

Abstract

Search engines for biological information are not a new technology. Since the 1960s computers have emerged as an important tool for biologists. Online Mendelian Inheritance in Man (OMIM) is a comprehensive catalogue containing approximately 14 000 records with information about human genes and genetic disorders.

An approach called Latent Semantic Indexing (LSI) was introduced in 1990 that is based on Singular Value Decomposition (SVD). This approach improved the information retrieval and reduced the storage requirements. This thesis applies LSI on the collection of OMIM records. To further improve the retrieval effectiveness and efficiency, the author propose a clustering method based on the standard k-means algorithm, called Two step k-means.

Both the standard k-means and the Two step k-means algorithms are tested and compared with each other.

Problem description

Optimal IR model for molecular biological information

In this task the student shall explain and develop a suitable information retrieval model for storing molecular biological information. This also include to find an optimal way to index the information and ease the retrieval later.

Assignement given	16. November 2005
Supervisor:	Herindrasana Ramampiaro

Acknowledgments

Hereby, I would like to thank my teaching supervisor Herindrasana Ramampiaro, associate professor by Norwegian University of Science and Technology, for finding this thesis topic and supervising it.

I would also like to thank my family and friends for inspire me to working steady on this project.

1 Introduction

"*Biology has traditionally been an observational rather than a deductive science. Although recent developments have not altered this basic orientation, the nature of the data has radically changed*" [29]. The field of biology produces a significant amount of information and the need to store and retrieve it in a reasonable matter is of great interest. The Online Mendelian Inheritance in Man (OMIM) [40] is an online catalogue of human genes and genetic disorders that is covering the heritable variations that follows the Mendelian Inheritance patterns.

Genetics is a scientific discipline that focuses on genes, heredity, and the variation of organisms. The scientific field of genetics seeks to understand the mechanisms of inheritance by implicitly utilize the inheritance phenomenon in breeding of organisms and selection for desired traits.

The chemical structure of DNA molecules contains the genetic information of organisms. Regions in the DNA sequence are called genes and corresponds to individually inherited traits. When synthesizing proteins the genes encode the necessary information. The DNA sequence is then transcribed into an intermediate molecule called mRNA. The mRNA sequence is then translated by ribosomes to form a chain of amino acids to form a protein. This process is known as the central dogma of molecular biology [7] [39].

Genetics is not the only aspect in the determination of the appearance and behavior of organisms. The ultimate outcome is determined by the interaction of the genetics with the environment. The outcome follows some inheritance patterns and makes it possible to forecast. Examples of heritable variations in humanse are eye and hair color, genetic disorders, and etc.. The inheritance pattern is called Mendelian Inheritance and was first introduced by Gregor Mendel. Mendelian Inheritance is a study of heritable variations that can be observed among individuals in a population. The study is based on a set of primary principles relating to the transmission of hereditary characteristics from the parent organisms to their children.

This thesis will concentrate on the development of a suitable information retrieval model for the OMIM collection. The selected model will facilitate the indexing and retrieval process of the information in the collection.

1.1 Motivation

Search engines are mostly developed and associated with the retrieval of web pages on the internet, but areas such as biology has seen the usefulness of search engines and has adapted them to their use. The biology field develops a lot of information and the amount is still growing, so the need for tools to require and process the information is of great interest. Since the 1960s computers have been an important tool in biology for storing and retrieval of biological information [20]. Just like standard search engines the amount of data gives rise to problems for the developers and the users such as:

- How to store the data?
- How to reach the required information?
- How to retrieve information in a quick and accurate way?

- Are the retrieved data relevant to the user's requirements?
- Are the users satisfied with the search result?

NCBI's OMIM seach engine let the users search the collection with a basic Boolean retrieval model. This thesis will look at how Latent Semantic Indexing (LSI) and two clustering methods affects the retrieval results when applied on the OMIM collection. LSI and have been applied to regular text documents earlier with good retrieval results [10].

1.2 Problem statement

In the introduction part the importance of storage and retrieval of information are stated. Retrieving information from huge collections is not always an easy task, so the main idea of this project is to explain and develop a suitable information retrieval model for storage of molecular biological information, specifically the OMIM collection. This also includes to find an optimal way to index the information and ease the retrieval later.

The main problems are:

- What is relevant to search for? Whether only biological information or some common data is retrievable. For instance, words such as "father", "stretchable", etc. are common, but are they relevant to search for? The choice here influences the retrieval and the precision of a search.
- What can be indexed? Whether to omit data types such as numbers, dates, etc..
- How to index? What underlying structure to use, Boolean retrieval model, vector space model, etc..
- How to search? Which similarity measure to use, cosine coefficient, Okapi BM25, etc..

In this project the author propose to investigate the retrieval effectiveness and precision of a self-made clustering method, called Two step k-means, compared with the standard k-means clustering method. The underlying structure is based on the LSI method. The goal of the proposed clustering method is to be not so greedy as the standard version and have a more rational document distribution.

1.3 Related works

Gleich and Zhukov [17] evaluate the application of the singular value decomposition to a search term suggestion system. They investigated the effect of SVD subspace projections for term suggestion ranking and clustering. They demonstrates the clustering behavior that occurs with LSI, but not with the exact cosine similarity. The steep of the clusters decline of the LSI curve corresponds to the end of related terms. The behavior does not occur for all terms in the dataset, where the terms do not display any clear clustering behavior.

A spherical k-means method was introduced for clustering documents, and the cluster centroids are identified as concept vectors and are compared to LSI index vectors. The subspace spanned by the concept vectors are close to the LSI subspace. This method has been further developed into concept indexing [12].

Jing et al. presents a text clustering system developed on a k-means type subspace clustering algorithm to cluster, high dimensional and sparse text data. They add a new step in the k-means clustering process to automatically calculate the feature weights for each cluster so that the important features to form a cluster subspace can be identified by the weight values This extension enables the k-means clustering algorithm to cluster high dimensional text data in subspaces of the keyword features, so that sparsity of text data can be effectively handled. The additional step does not increase the number of iterationes, and the performance of the k-means clustering process is preserved [25].

1.4 Structure of the thesis

The content of this thesis contain elements from both computer science and biology, so in order to understand and benefit from reading this thesis, basics from both subjects is provided. Some parts of this thesis may seem obvious and uninteresting for some readers, but this is done in order to give the readers the same foundation.

- Chapter 1 - Introduction of the master thesis and its topic. Motivation and problem statement are explained to give the reader some insight in the subject of this thesis, before related works, which describe some projects similar to the subject of this thesis, are listed.

- Chapter 2 - Necessary theoretical background and techniques used in this project are described in this chapter. First there is a short introduction of molecular biology before the basics of Mendelian Inheritance are described. Bioinformatics will also be described before some of the basic concepts of information retrieval, and finally Latent Semantic Indexing are described.

- Chapter 3 - Will give an overview of the Online Mendelian Inheritance in Man database, and specify research approach and its limitations for this project.

- Chapter 4 - Will apply the knowledge represented in Chapter 2 to describe how the solution got worked out. First a more indepth problem analysis is performed, listing of alternative solutions (LSI/PLSI), and finally the chosen solution.

- Chapter 5 - Technical details of the Lucene and the JAMA packages are described first, Then an overview of the architectural design is described before going into detail of each of the modules (Lucene modules - indexing process, clustering process, searching process, analyzer, application logic and functionalities, user interface). Three test cases is also provided in order to explain how it works.

- Chapter 6 - Will describe the validation of the system by performing different kind of tests: subsystem tests, system tests, usability tests and acceptance tests. Finally, the test results are evaluated.

- Chapter 7 - conclusion (future work). Analyze of test results represented in chapter 6 and evaluate the system. At the end of this chapter a summary of the results of the work performed in this project and suggested future research and improvement for the system.

1 Introduction

2 Background

Although it is not a requirement, the reader should have insight in both computer science and biology to best understand the content of this thesis. This chapter will give a brief description of each field to support readers with little or no insight in any of the areas. First, the biology field will be described, more precisely an introduction to molecular biology, and then go a little in depth on the subject Mendelian Inheritance. After the biology part has been described, some definitions of bioinformatics will be explained before basic information retrieval (IR) techniques and finally LSI is described.

2.1 Molecular biology

In molecular biology the organisms is studied at a molecular level, i.e. the smallest particle of a pure substance that still retains its composition and chemical properties. By studying the organisms at a molecular level, molecular biologists can study the interaction between the various systems of a cell. This includes the interrelationship of DNA, RNA and protein synthesis and how these interactions are regulated. Molecular biology overlaps with other areas of biology and chemistry, especially genetics and biochemistry [7].

Techniques, such as polymerase chain reaction, gel electrophoresis, southern blotting, and arrays, is used by researchers in molecular biology to characterize, isolate, and manipulate the molecular components of cells and organisms. The boundaries between the disciplines molecular biology, genetics and biochemistry is not as hard-lined as they once was, mainly because the disciplines interchange techniques more increasingly. Figure 2.1 illustrates one possible view of the relationship between the disciplines [7]:

- *Biochemistry* is the study of the chemical processes and transformations occuring in living organisms. It deals with the structure and function of cellular components, such as proteins, carbohydrates, lipids, nucleic acids, and other biomolecules.
- *Genetics* is the study of genes, heredity, and the effect of genetic differences on organisms. The scientific field of genetics seeks to understand the mechanisms of inheritance. Variation of organisms can be inferred by the absence of a normal component, e.g. a gene. In a natural population the normal phenotype for a character is called the wild type phenotype, and alternative traits to the wild type are called

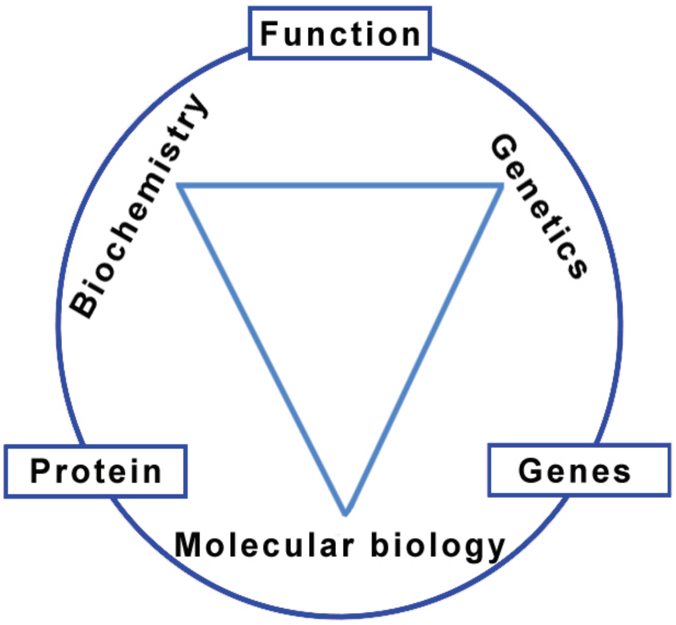


Figure 2.1: Relationship between the biology disciplines

mutant phenotypes. Mutant phenotypes have originated from changes or mutations in the wild type allele.

- *Molecular biology* is the study of molecular underpinnings of the process of replication, transcription and translation of the genetic material. The central dogma of molecular biology is a framework for understanding the transfer of sequence information between sequential information-carrying biopolymers[1] [39]. There are three major classes of such biopolymers: DNA and RNA (both nucleic acids), and protein. The general transfers describe the normal flow of biological information: DNA can be copied to DNA (DNA replication), DNA information can be copied to mRNA (transcription), and proteins can be synthesized using the information in messenger RNA (mRNA) as a template (translation) [7]. Although the figure 2.2 is an oversimplified picture of the central dogma of molecular biology, it still provides a good starting point for understanding the field.[2]

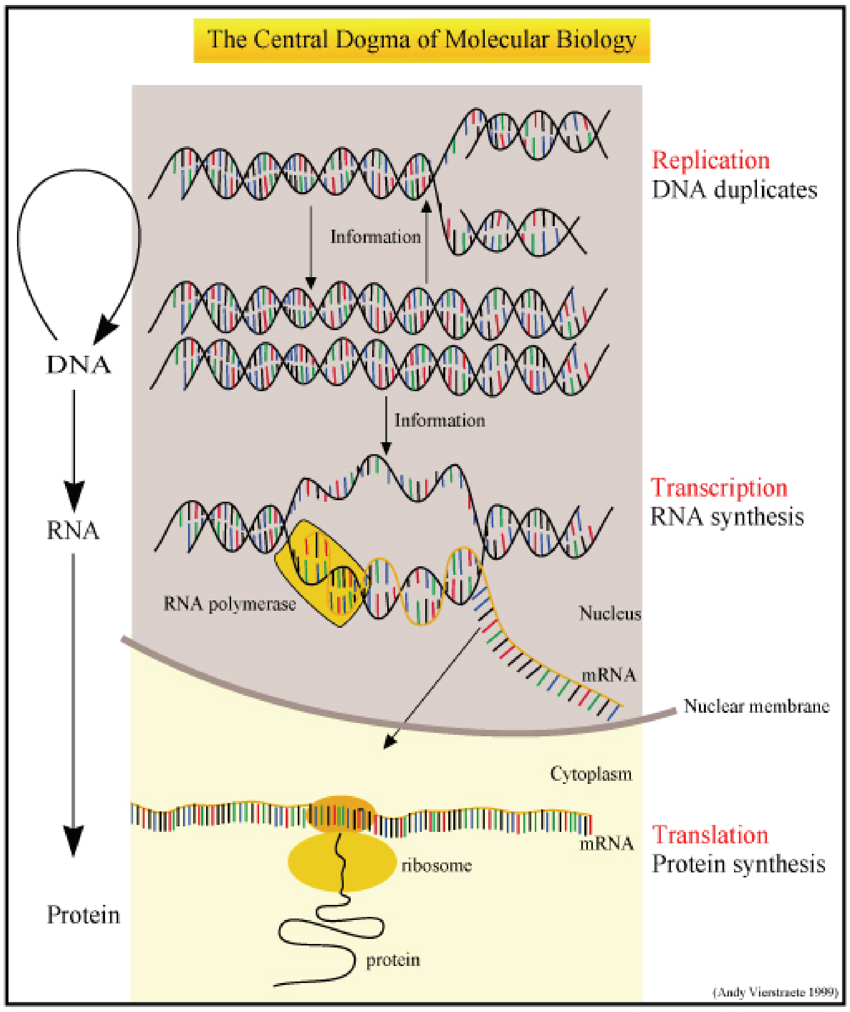


Figure 2.2: The Central Dogma of Molecular Biology

1. A special class of polymers produced by living organisms.
2. Picture is taken from http://users.ugent.be/~avierstr/principles/centraldogma.html

2.1.1 The cell and molecules of life

Every living organism is made of cells - a human being consists of approximately 10^{13} cells [1]. There are two types of cells; prokaryotic and eukaryotic cells. Only organisms of the domains bacteria and archaea consists of prokaryotic cells, while animals, plants, fungi, and protists consists of eukaryotic cells. Prokaryotic cells are smaller than eukaryotic cells, which can be so big one may spot it with the human eye, and have simpler structure. To some extent is a cell self-contained and self-maintaining, meaning it can take in nutrients, convert the nutrients into energy, carry out specialized functions, and reproduce as necessary. The instructions needed to carry out the mentioned vital functions are stored in each cell. In eukaryotic cells the genetic instructions are stored in the nucleus and carried out by the ribosomes, see figure 2.3 for an illustration of a generalized animal cell [7].

Common for cells are reproduction by cell division, use of enzymes and other proteins, metabolism, response to external and internal stimuli (e.g. changes in temperature, pH or nutrient levels), and the cell content is encapsulated with a cell surface membrane. Cell division, called cell cycle, is the cell's fashion to reproduce itself, this is done by binary fission, mitosis or meiosis. Multicellular organisms typically begin life as a single cell, usually a result of fusion of a male and a female sex cell. The sex cells are called gametes.

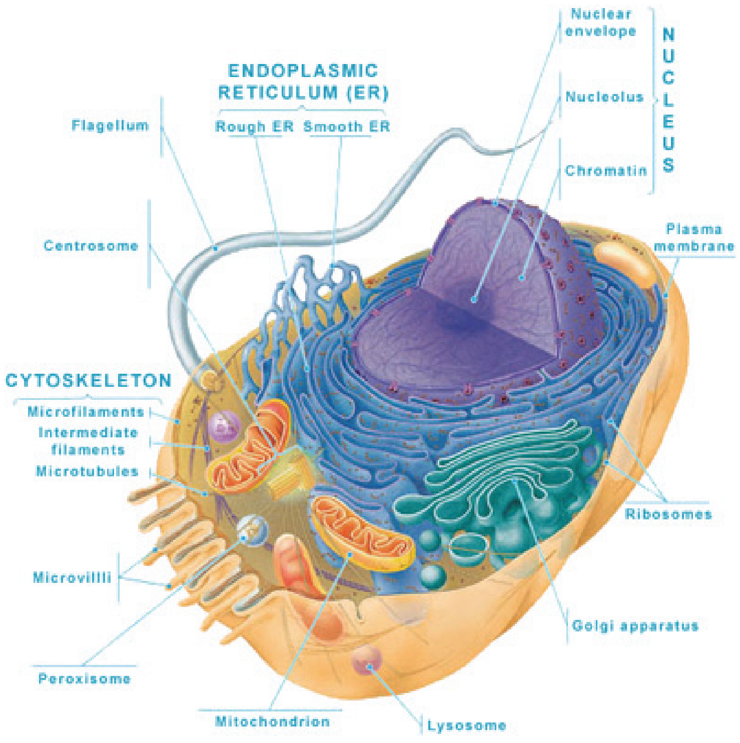


Figure 2.3: Model of a generalized animal cell (picture taken from [7].

Metabolism is a set of chemical reactions that occurs in living cells. These processes are the basis of life that takes in raw materials, builds cell components, converts energy, molecules and release by-products. Metabolic pathways derive energy from chemical energy stored in organic molecules.

To produce tissues and organs a single cell has to grow, divide and differentiate into different cell types. If this cell division and differentiation is not controlled then cancerous cells can grow to form tumors.

The difference between prokaryotic and eukaryotic cells is on the basis of nuclear organization, specifically the prokaryotic cells' lack of a nuclear membrane. Cells of this type also lack most of the intercellular organelles and structural characteristics of eukaryotic cells (except the ribosomes, which are present in both prokaryotic and eukaryotic cells). The prokaryotic plasma membrane takes over most of the functions of organelles, such as mitochondria, chloroplasts and the Golgi apparatus. Architecturally prokaryotic cells have three regions [7]:

- proteins attached to the cell surface (appendages called flagella and pili)
- a cell envelope consisting of a capsule, a cell wall and a plasma membrane;
- a cytoplasmic region containing the cell genome (DNA) and ribosomes and various sorts of inclusions

Some other differences [7]:

- the plasma membrane (a phospholipid bilayer) separates the interior of the cell from its environment and serves as a filter and communications beacon.
- most prokaryotes have a cell wall (some exceptions are Mycoplasma (a bacteria) and Thermoplasma (an archaeon)). It consists of peptidoglycan in bacteria and acts as an additional barrier against exterior forces. It also prevents the cell from "exploding" (cytolysis) from osmotic pressure against hypotonic environment. A cell wall is also present in some eukaryotes like plants (cellulose) and fungi, but has a different chemical composition.
- a prokaryotic chromosome is usually a circular molecule (an exception is that of the bacterium Borrelia burgdorferi, which causes Lyme disease). Even without a real nucleus, the DNA is condensed in a nucleoid. Prokaryotes can carry extrachromosomal DNA elements called plasmids, which are usually circular. Plasmids can carry additional functions, such as antibiotic resistance.

Eukaryotic cells are about 10 times the size of a typical prokaryote and can be as much as 1000 times greater in volume, and they contain membrane-bound compartments in which specific metabolic activities take place. The presence of a cell nucleus, a membrane-delineated compartment that houses the eukaryotic cell's DNA, is the most important among these compartments. This nucleus gives the eukaryote its name, which mean "true nucleus". Other differences are [7]:

- the plasma membrane resembles that of prokaryotes in function, with minor differences in the setup. Cell walls may or may not be present.
- the eukaryotic DNA is organized in one or more linear molecules, called chromosomes, which are separated from the cytoplasm by a membrane. Some eukaryotic organelles also contain some DNA.
- eukaryotes can move using cilia or flagella. The flagella are more complex than those of prokaryotes.

All cells have a membrane that envelopes the cell. This membrane separates its interior from its environment, regulates what moves in and out (selectively permeable), and maintains the electric potential of the cell. A salty cytoplasm inside the membrane takes up most of the cell volume. Cells possess the hereditary material of genes (DNA), and RNA which contains the information necessary to build various proteins such as enzymes, the cell's primary machinery. Also other kinds of biomolecules are present in cells.

Two different kinds of genetic material exist: deoxyribonucleic acid (DNA) and ribonucleic acid (RNA). Some viruses (e.g. retroviruses) have RNA as their genetic material, but most organisms use DNA for their long-term information storage. The biological information contained in an organism is encoded in its DNA or RNA sequence. RNA is also used for information transport (e.g. mRNA) and enzymatic functions (e.g. ribosomal RNA) in organisms that use DNA for the genetic code itself.

Prokaryotic genetic material is organized in a simple circular DNA molecule (the bacterial chromosome) in the nucleoid region of the cytoplasm. Eukaryotic genetic material is divided into different, linear molecules called chromosomes inside a discrete nucleus, usually with additional genetic material in some organelles like mitochondria and chloroplasts.

A human cell has genetic material in the nucleus (the nuclear genome) and in the mitochondria (the mitochondrial genome). In humans the nuclear genome is divided into 46 linear DNA molecules called chromosomes. The mitochondrial genome is a circular DNA molecule separate from the nuclear DNA. Although the mitochondrial genome is very small, it codes for some important proteins.

It is also possible to artificially introduce foreign genetic material (most commonly DNA) into the cell by a process called transfection. This can be transient, if the DNA is not inserted into the cell's genome, or stable, if it is.

The simplest collection of matter that can live is the cell. Complex organisms, for example plants and animals, are multicellular. Their bodies are cooperatives of many kinds of specialized cells that could not survive long on their own, because even they are arranged into higher levels of organization, such as tissues and organs. The organism's basic unit of structure and function is certainly the cell.

2.1.2 Genes

Within the cells you can see structures called chromosomes, partly made of a substance called deoxyribonucleic acid (DNA). The DNA is in turn the substance of genes, the units of inheritance that transmit information from parents to offspring [7].

With hundreds or thousands of genes arranged along its length, each chromosome has one very long DNA molecule. Replication of the chromosomes' DNA is accomplished when a cell prepares to divide. A complete set of genes is then inherited by each of the two cellular offspring [7].

DNA's molecular structure accounts for its information-rich nature. Two long chains arranged into what is called a double helix make up each DNA molecule, see figure 2.4[3]. Each of the links in a chain is one of four kinds of chemical building blocks called nucleotides [7].

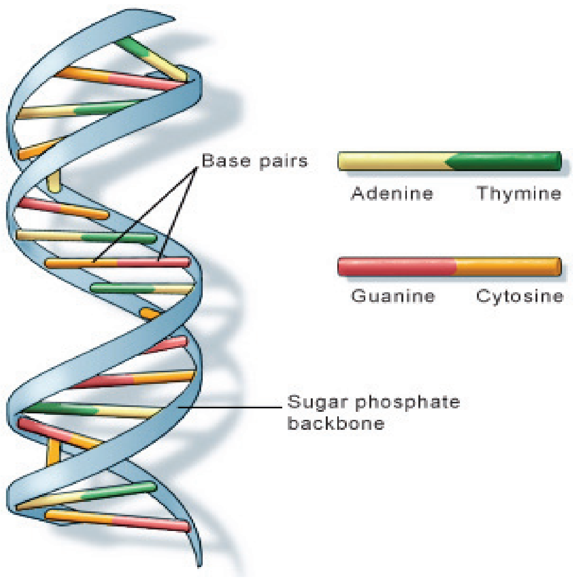


Figure 2.4: Double Helix

The work in molecular biology is often quantitative. A lot of work has been done at the interface of molecular biology and computer science in bioinformatics and computational biology recently. The study of gene structure and function, molecular genetics, has been amongst the most prominent sub-field of molecular biology since the early 2000s.

Many other fields of biology focus on molecules. The focus is either directly, i.e. studying their interactions in their own right such as in cell biology and development biology, or indirectly where the techniques of molecular biology are used to deduce historical attributes of populations or species, as in population genetics and phylogenetics (fields in evolutionary biology) [7].

2.2 Mendelian Inheritance

This thesis only covers a part of molecular biology known as Mendelian inheritance. In order to give the reader some background of Mendel and his work, this chapter is written with the non-biologist in mind.

Gregor Mendel was a monk in the 19th century and experimented with cross-pollinated two contrasting, true-breeding pea varieties in his monastery's experimental garden. Mendel's experiments brought forth two generalizations which later became known as *Mendel's Laws of Inheritance*. Mendel's Laws of Inheritance contains two laws; *the law of segregation* and *the law of independent assortment*. Mendel used, as mentioned, peas in his experiments and peas are available in many varieties. Available varities of peas are flower color, which is a heritable feature, shape, etc.. In this thesis flower color will be used to exemplify Mendel's discoverings unless other specified. One variety of peas is with purple flowers and another one with white flowers. In experiments Mendel crossed for example these two varieties to see the effect on the offspring. The crossing of two true-breeding varieties is called hybridization. The parents, in a true-breeding, are referred to as the P generation (parental generation), the offspring of the P generation are the F_1 generation (first filial[4] generation). The F_1 hybrids self-pollinates and

3. Picture taken from http://www.coe.drexel.edu/ret/personalsites/2005/dayal/curriculum1.html
4. Pertaining to the sequence of generations following the parental generation, each generation being designated by an F followed by a subscript number indicating its place in the sequence (ref.: dictionary.com "filial").

produces an F_2 generation (second filial generation) [7]. In the following subchapters the law of segregation, the law of independent assortment and the spectrum of dominance will be described.

2.2.1 The law of segregation

In Mendel's terminology there is a dominant trait and a recessive trait. In this thesis the purple flower color is the dominant and the white flower color is the recessive. Mendel's experiments produced a 3:1 (purple : white) inheritance pattern ratio for the F_2 generation, where the purple flower color is a dominant trait and the white flower color is a recessive trait. To explain the 3:1 ratio in his pea experiments Mendel developed four related concepts [7]:

- *Alternative versions of genes account for variations in inherited characters*. This is the concept of alleles. An allele is any of the possible forms in which a gene for a specific trait can occur. In other words alleles are different versions of genes that impart the same characteristic. For instance the gene for flower color exists in two versions; purple and white.

- *For each character, an organism inherits two alleles, one from each parent*. This means that when somatic cells are produced from two gametes, one allele comes from the mother, one from the father. These alleles may be the same (true-breeding organisms, e.g. ww and PP), or hybrids (e.g. Pw) in figure 2.5.

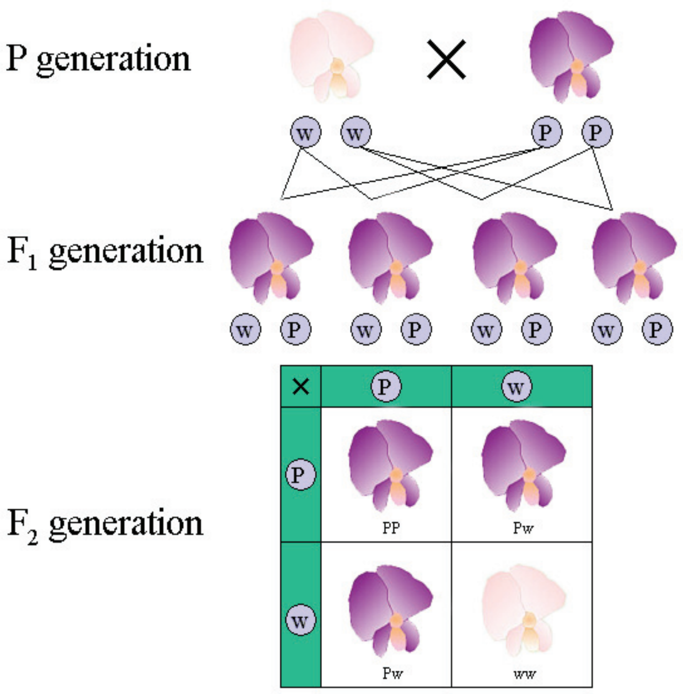


Figure 2.5: Dominant and recessive phenotypes

- *If the two alleles at a locus differ, then one, the dominant allele, determines the organism's appearance; the other, the recessive allele, has no noticeable effect on the organism's appearance*. In other words, the dominant allele is expressed in the phenotype of the organism; however this does not always hold true. There is incomplete dominance and codominance on a molecular level, e.g. people with sickle cell anemia, when normal and sickle-shaped red blood cells mix and prevent malaria.

- *The two alleles for a heritable character separate (segregate) during gamete formation and end up in different gametes*. This is the last part of Mendel's generalization and known as the law of segregation. The two alleles of the organism are separated into different gametes, ensuring variation.

Because of the different effects of dominant and recessive alleles, an organism's traits do not always reveal its genetic composition. Therefore, one distinguish between an organism's traits, called its phenotype, and its genetic makeup, its genotype (see figure 2.6)[5] [7] [11].

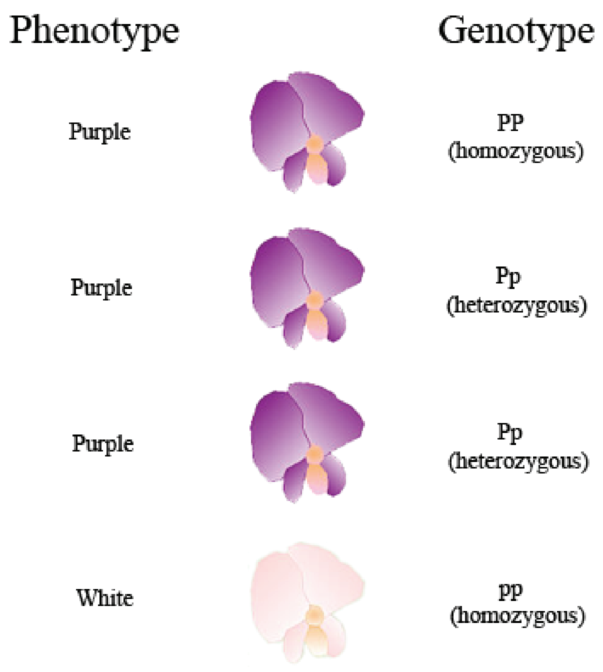


Figure 2.6: Phenotype and genotype

When the paired alleles are of same genotype, it is called to be homozygous for the gene controlling that character...

When the paired alleles (one on each of two paired chromosomes) are the same they are called *homozygous* for the gene controlling that character, if they are different they are called *heterozygous* (see figure 2.6). In heterozygous pairings, one allele is usually dominant, and the other recessive. The dominant allele decides the phenotype of the flower, in figure 2.6 the purple flower color is the dominant one. Simple traits such as eye color may be caused by just one pair of alleles, while complex traits such as height are usually caused by interactions of numerous pairs of alleles. Heterozygous genotypes are not true-breeding because they produce gametes with different alleles [7] [11].

2.2.2 The law of independent assortment

By following only a single character, such as flower color, in his breeding experiments Mendel derived the law of segregation. In Mendel's crosses of true-breeding parents all the F_1 progeny were monohybrids, i.e. heterozygous for one character. Expanding the previous experiment and crossing two true-breeding pea varieties, which differs in both color and shape, for instance a plant with yellow-round seeds and a plant with green-wrinkled seeds, the F_1 plants will be dihybrids, meaning that they are heterozygous for both characters.

Mendel's findings from the dihybrid experiments are the basis for what is now called the law of independent assortment. The law of independent assortment states that each pair of alleles segregates independently of other pairs of alleles during gamete formation. This law applies only to genes (allele pairs) located on different chromosomes, that is, on chromosomes that are not homologous. Genes located near each other on the same chromosome tend to be inherited together and have more complex inheritance patterns than predicted by the law of independent assortment.

5. P - dominant phenotype
   p - recessive phenotype

2.2.3 *The spectrum of dominance*

In relation to each other the alleles can show different degrees of dominance and recessiveness. This is refered to as the spectrum of dominance. The F_1 offspring of Mendel's classic pea crosses is one extreme on this spectrum. The complete dominance of one allele over another caused the F_1 plants to always look like one of the two parental varieties. In this situation, the phenotypes of the heterozygote and the dominant homozygote are indistinguishable.

In crosses where both of the alleles inherited from the parents can affect the phenotype in separate and distinguishable ways, is called codominance. For example, the human MN blood group is determined by codominant alleles for two specific molecules located on the surface of red blood cells, the M and N molecules. A single gene locus, at which two allelic variations are possible, determines the phenotype of this blood group. Individuals homozygous for the M allele (MM) have red blood cells with only M molecules; individuals homozygous for the N allele (NN) have red blood cells with only N molecules. But both M and N molecules are present on the red blood cells of individuals heterozygous for the M and N alleles (MN). Note that the MN phenotype is not intermediate between the M and N phenotypes. Rather, both the M and N phenotypes are exhibited by heterozygotes, since both molecules are present [7].

For some characters the alleles fall in the middle of the spectrum of dominance. The F_1 hybrids in this case, have a phenotype somewhere in between the phenotypes of the two parental varieties, and is called the incomplete dominance. This phenomena can be seen when crossing red snapdragons with white snapdragons: All of the F_1 hybrids have pink flowers (see figure 2.7) [7]. Unlike the situation in Mendel's pea plants experiments, where the Pp heterozygotes make enough pigment for the flower to be a purple color indistinguishable from thos of PP plants, these flowers have less red pigment than the red homozygotes.

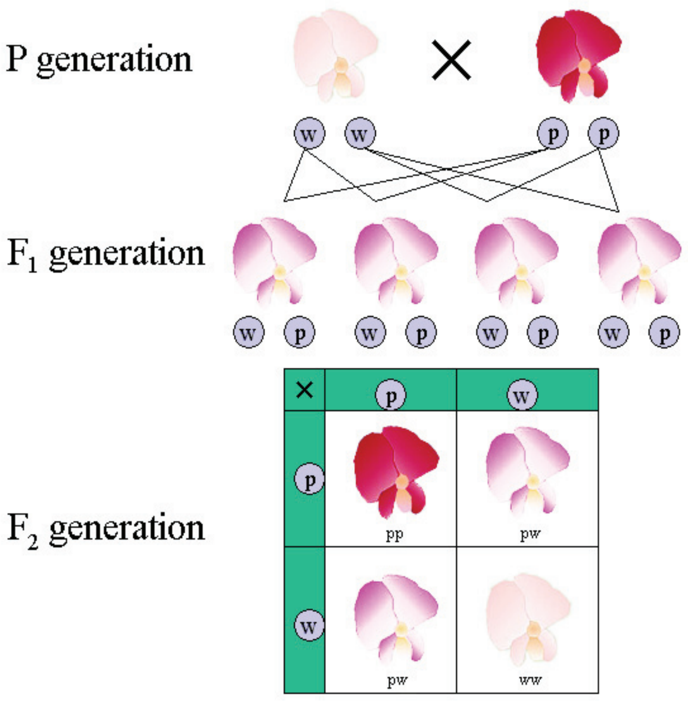


Figure 2.7: Dominant and recessive phenotypes (snapdragon)

2.2.4 *Mendelian patterns in human traits*

Unlike peas, humans are not convenient subjects for genetic research. Some reasons for this is that humans produce relatively few offspring and that one human generation span is about 20 years. Experiments like those Mendel performed on peas are unethical and not acceptable with humans. Since the geneticists can not manipulate the mating patterns of people, they must analyze the results of matings that have already occurred. This involves collecting information about a family's history for a particular trait and assembling this information into a family tree describing the interrelationships of parents and children across the generations. Traits such as attached or free earlobe, widow's peak or no widow's peak, eye color, and etc., can be that particular trait.

Pedigree analysis can be used to deduce the possible genotype of individuals and make predictions about future offspring. As one work oneself through the pedigree, one can apply the rules from the pea experiments to fill in the genotypes for most individuals. Although the pedigree analysis may predict the future, the predictions are not certainties but usually statistical probabilities. Causes of genetic disorders are:

- **Recessively inherited disorders** - Genetic disorders are caused by alleles that codes either for a malfunctional protein or for no protein at all. In the case of disorders classified as recessive, heterozygotes are normal in phenotype because one copy of the normal allele produces a sufficient amount of the specific protein. Although the individual's phenotype is normal with regard to the disorder, heterozygotes may transmit the recessive allele to their offspring. In these cases the individual is a carrier of the disorder. For a recessively inherited disorder to show the individual must be homozygous, whom has inherited one recessive allele from each parent. Examples of recessively inherited disorders are Taysachs disease, cystic fibrosis, sickle-cell disease, and many other.

- **Dominantly inherited disorders** - Lethal diseases are much less commonly caused by dominant alleles than recessive alleles. Changes to the DNA in a sperm or an egg arises such lethal alleles. A lethal dominant allele will not be passed on to future generations if the allele causes the death of the offspring before they mature and can reproduce. In contrast, heterozygous carriers of a lethal recessive allele can reproduce and pass on the allele from generation to generation. The lethal disease will only appear at homozygous recessive offsprings. If the lethal dominant allele causes death only at a relatively advanced age, it can escape elimination because the individual may have already transmitted the lethal allele to his og her children by the time the symptoms become evident. Examples of dominantly inherited disorders are achondroplasia and Huntington's disease.

- **Multifactorial disorders** - Both the recessively and dominantly inherited disorders are described as simple Mendelian disorders due to the result of abnormality of one or both alleles at a single genetic locus. Multifactorial disorders have both genetic and environmental components. Regular exercise, a healthful diet, and etc. reduce the risk of heart disease and cancer. Multifactorial disorders do not follow simple Mendelian inheritance patterns. Examples of multifactorial disorders are heart disease, diabetes, cancer, alcoholism, and certain mental illnesses.

Many families wants to determine what the odds are that their children will have any genetic disorders before conceiving their first child. By using family histories and genetic counselors a couple can determine the risk of having a child with a disorder. If a child is conceived amniocentesis and chorionic villus sampling can help determine whether a genetic disorder is present. After the child is born further genetic tests can be performed.

2.3 Bioinformatics

Already in the early 1960s computers emerged as an important tool in molecular biology, which laid important conceptual and technical foundations for what is known as bioinformatics today [20]. Although computers emerged as an important tool in molecular biology in the 1960s, it was not until the early 1990s that bioinformatics got acknowledged as a discipline, which included informatics, mathematics, medicine, physics and biology [2] [4]. Computational biology is occasionally used synonymously with bioinformatics, although of all computational and information science approaches computational biology tends to be more inclusive than bioinformatics. Bioinformatics is currently focused on computational molecular biology [2].

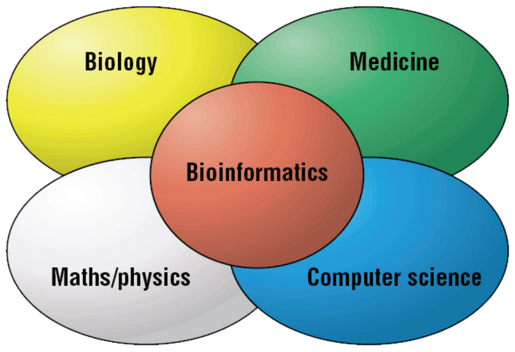


Figure 2.8: Overview of bioinformatics (picture taken from [4])

There are various definitions of what bioinformatics is and Luscombe [32] defines bioinformatics as "*the application of computational techniques to understand and organise the information associated with biological macromolecules*". The European Bioinformatics Institute [14] have a slightly different definition of bioinformatics: "*the application of computer technology to the management and analysis of biological data*". The term bioinformatics can be used in a broad or a narrow way. A broad usage of the term bioinformatics includes all applications of computers and information sciences to problems in biology, while a narrow usage is refered to the creation and management of biological databases of genomic sequences [2].

The goals of bioinformatics can be divided into three parts: (1) the simplest goal of bioinformatics is to organize the data in such manner that the researchers get easy access to existing information and can submit new entries gradually as they are produced, or edit old entries. Until the information in these databases is analyzed they are useless. (2) the next goal is to develop tools and resources to aid the analysis of the data stored in the databases. For instance when one have a sequence for a protein and wants to compare it with previously characterized sequences. Tools developed for this comparison makes it easier and saves times. (3) the final goal is to use the tools to analyze the data and interpret the results in a meaningful manner [32].

Unlike traditional analyses one can now conduct global analyses of all the available data. Where traditional studies examined individual systems in detail and then compared it with a few that are related, one can with a global analysis unveil common principles that can be applied across many systems and extract novel features. In bioinformatics most of the analysis are focused on three primary data sources: (1) DNA or protein sequences, (2) macromolecule structures, and (3) the results of functional genomics experiments [32].

The challenges in bioinformatics in latter time is to get an intelligent and efficient storage of biological data. It also requires easy and reliable access to the data for the users.
This project will focus on the development of information retrieval tools for biological information on genetic dissorders. The tool make use of Latent Semantic Indexing which will be described in chapter 4.

2.4 Basic concepts of information retrieval

Traditional information retrieval (IR) systems make use of index terms to index and retrieve documents. This project will use this method to retrieve information from the various dissorders. An index term is a regular word extracted from the documents. The document's main theme is aided by the word's semantics and characterizes the content of the document. In IR, there are three fundamental models; Boolean model, vector space model, and probabilistic model. The models describe the fundamentals of indexing, searching, weighting, ranking, providing queries and document evaluations. The models are important to fully understand the steps regarding the indexing and searching functions. In addition to the three fundamental IR models a cluster-based model is often used to increase retrieval effectiveness and efficiency, grouping similar documents into clusters, etc. [30]. All of the models will be described later, but the vector space model and the cluster model is utilized in this project and the focus will be on them.

Before getting into the basic IR models there are som steps that has to be described. In IR there are two classical parameters to determine the quality of the provided service; recall and precision. The recall is the number of retrieved documents divided by the number of relevant documents, while the precision is the number of retrieved relevant documents divided by the number of relevant documents. The trade-off of these values is that when the recall increases the precision decreases, and vice versa.

$$Recall = \frac{Retrieved\ documents}{Relevant\ documents}$$

$$Precision = \frac{Retrieved\ relevant\ documents}{Retrieved\ documents}$$

The recall and precision values are important to evaluate the quality of the search engine according to user preferences. For instance, if the precision value increases then the user gets more relevant documents.

Prior to the actual indexing it is often an advantage to do some preprocessing. Some of the words in the text are not suitable as index terms and should therefore be removed. A proposed view of the preprocessing is illustrated in figure 2.9.

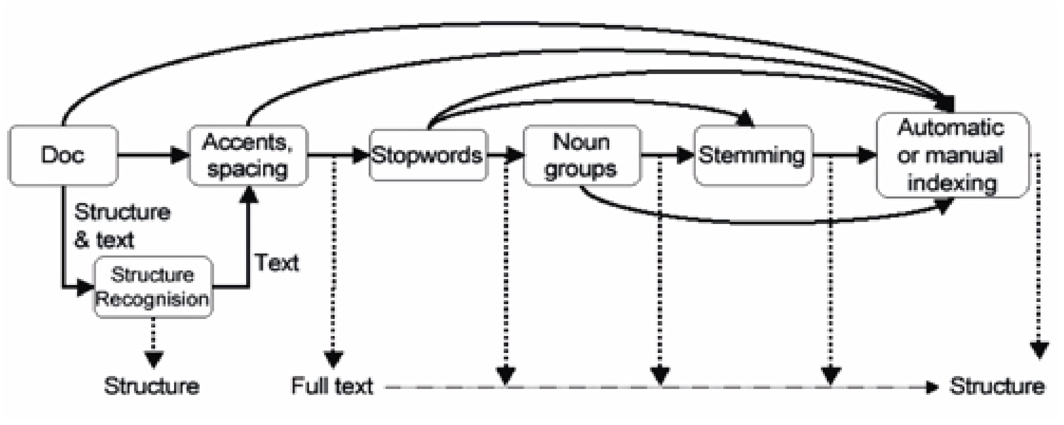


Figure 2.9: Preprocessing phases (picture taken from [3])

The text preprocessing can be divided into three operations:

- *Lexical analysis*: in lexical analysis the stream of characters is converted to a stream of words. Digits and dates, hyphens and punctuation marks are removed from the stream. In addition all of the characters can be set to lowercase or uppercase to ease the identification of index terms, this is done in standard Lucene implementations.
- *Elimination of stopwords*: stopwords are words that are filtered out as potential index terms. These words occurs too frequently to be good discriminators. Natural candidates for a list of stopwords are articles, prepositions, conjunctions, adverbs, and adjectives. In addition some application specific words with low discrimination value should be added to the list of stopwords.
- *Stemming*: stemming is a process where words in their syntactic variations (plurals, gerund forms, past tense suffix, etc.) gets reduced to their root form, e.g. the words "fishing", "fishes", "fish", and "fisher" gets stemmed to their root form "fish". By reducing words to their root form the size of the index structure is also reduced. One popular stemming algorithm, and used in this project, is the Porter stemmer [30] [41].

Because of the varying usefulness of the index terms to describe the document content, it might be rational to normalize the term weights. The normalization of the index term weights is done so large documents with many keywords will not overwhelm similar documents in the result set. One weighting scheme, the best known weighting scheme, is the tf-idf. This weighting scheme will be used in this project and will therefore be described in detail in the subsequent subchapter.

tf-idf

Index term weighting is often used in IR and text mining. Tf-idf is a weighting method built upon statistics to evaluate how important the words are for the documents in the collection. The number of times a term in a given document is called the term frequency. In this project the weighting is normalized to prevent a bias towards longer documents, to give a measure of the importance of the term t_i within the particular document.

In a collection of N number of total documents and the term t_i appears in n_i number of documents, the normalized frequency $f_{i,j}$ of term t_i with the raw term frequency $freq_{i,j}$ is given by the following formula:

$$f_{i,j} = \frac{freq_{i,j}}{maxfreq_{i,j}}$$

The maximum frequency is computed over all terms mentioned in the document d_j. The formula to compute the inverted document frequency idf_i is:

$$idf_i = \log\frac{N}{n_i}$$

The normalized index term weighting is then given by the formula:

$$w_{i,j} = f_{i,j} \times idf_i$$

Basic Boolean retrieval model

The Boolean IR model is a classical IR model and most of the commercial systems today can be classified as Boolean IR system or text-pattern search systems [30]. The model is based on set theory and Boolean operators, so the documents to be searched and the query are treated as a set of terms. During retrieval the documents containing the query terms are returned to the user - the retrieval strategy is binary, either the document contain the term or not (cf. relevant or not). There are in other words no grading of relevance. The query terms can also be joined by logical operators that supplies relationships between the terms in the query, three common used operators are; AND, OR, and NOT. However, the retrieval results are very sensitive to the query formulations.

Basic vector space retrieval model

The vector space model is an alternative retrieval model to the Boolean model, and it represents the documents and the query as vectors. Each dimension in the vector is the weight of the term t in the document i or term t in query j. For instance a document D_i and a query Q_j with N total number of terms are represented as (figure 2.10 illustrates a document vector, query vector and the angle between them):

$$D_i = [T_{i1}, T_{i2}, \ldots, T_{ik}, \ldots, T_{iN}]$$
$$Q_j = [Q_{j1}, Q_{j2}, \ldots, Q_{jk}, \ldots, Q_{jN}]$$

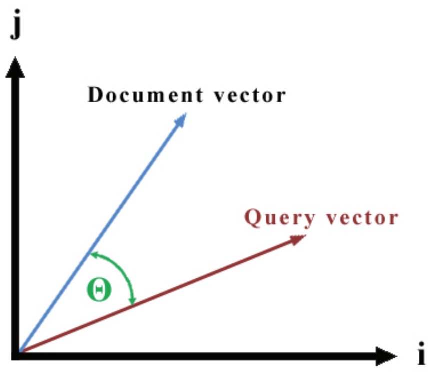


Figure 2.10: Vector space model

The weight of the terms can be binary, based on the tf-idf or weighted by some other index term weighting scheme.

During retrieval a similiarity score is calculated. There are several similarity measures but the cosine coefficient is the most popular. The cosine coefficient calculates the similarity between the document *i* and the query *j* by the following formula:

$$Sim(D_i, Q_j) = \frac{\vec{D_i}\ \vec{Q_j}}{|\vec{D_i}||\vec{Q_j}|} = \frac{\sum_{k=1}^{N} T_{ik} \cdot Q_{jk}}{\sqrt{\sum_{k=1}^{N} T_{ik}^2} \cdot \sqrt{\sum_{k=1}^{N} Q_{jk}^2}}$$

The similarity generated by the formula above is the cosine of the angle between the two vectors. Unlike the Boolean retrieval model the vector space retrieval model ranks documents according to similarity rather than attempting to predict relevance between the query and the documents. The documents retrieved with the vector space model may only be partial match with the query. The result set can also be sorted in descending order.

Although the vector space model have its advantages, compared to other IR models by improving the retrieval performances with term weighting, its main limitation is that it treats terms as unrelated and it only works well with short documents and queries [30]. The Latent Semantic Indexing model is based on the vector space model and will be described in chapter 2.5.

Probabilistic retrieval model
Within a probabilistic framework the probabilistic retrieval model considers term dependencies and relationships. The model is based on these parameters [30]:

P(rel): the probability of relevance of a document
P(nonrel): the probability of nonrelevance of a document
a_1: the cost associated with the retrieval of a nonrelevant document
a_2: the cost associated with the nonretrieval of a relevant document

P(rel) and P(nonrel) has to be estimated and this is the main issue of the probabilistic retrieval model. Normally, a certain term occurrence distribution in documents is assumed to estimate those parameters [30]. The probabilistic retrieval model has not improved retrieval effectiveness greatly, mainly because there are some difficulties of obtaining P(rel) and P(nonrel). Relevant and nonrelevant documents need to be initial separated. One of the most popular probabilistic retrieval models are the Okapi BM25 ranking function [43].

Cluster-based retrieval model
Information retrieval models discussed so far might not put similar documents in close proximity in the file system. It is difficult to implement browsing capability in such a file organization. Low retrieval effectiveness and efficiency occur because not all relevant items may be retrieved and whole document space has to be searched. Document clustering, the grouping of similar documents into clusters, was introduced to overcome these disadvantages [30].

Two approaches to cluster generation are present today; hierarchial clustering and flat clusters [45] [30]. The first one is based on all pairwise document similarities and assembles similar items into common clusters. The second uses heuristic methods that do not require pairwise document similarities to be computed.

In the pairwise similarities approach, each document is represented as a document vector in the vector space model. The similarity between each pair of documents is then calculated. Each document is initially placed into a class by itself during the clustering process. Then the two most similar documents based on the pairwise similarities are combined into a cluster. The similarities between the newly formed cluster and other documents are calculated, and the most similar documents (including the cluster) are combined into a new cluster. This combining process continues until all documents are grouped into a super-cluster. This process is called a hierarchical, agglomerative clustering process.

The hierarchical clustering methods, based on all pairwise similarities between documents, are relatively expensive to perform, but produce a unique set of well formed clusters for each set of documents. Hierarchical clustering methods are reported to be far too limited to capture the rich semantics of most document sets and do not permit cross classifications [10], are therefore not utilized in this project.

The other approach, the heuristic clustering methods, produces rough cluster arrangements rapidly at relatively little expense. The one-pass procedure, the simplest heuristic process, takes the documents to be clustered one at a time in arbitrary order. The first document is placed in a cluster of its own, and then each subsequent document is compared with all existing clusters, and is placed into an existing cluster if it is sufficiently similar to that cluster. If it is not similar enough the document is placed in a new cluster of its own. The process continues until all documents are clustered. The cluster structure generated depends on the order in which documents are processed and is uneven. To produce usable clusters some control mechanisms are required.

Document search and retrieval is effective and efficient after the clusters are formed. Each cluster has a representative vector, normally its centroid. The centroid is typically calculated as the average vector of all documents of the cluster (i.e. the weight of centroid term i is defined as the average of the weights of the ith terms of all documents).

During document retrieval the query vector is compared with the clusters' centroids. After identifying the cluster with the highest similarity to the query vector there are two alternatives. Alternative 1; all documents in the cluster are retrieved. Normally this is done when clusters are small. Alternative 2; compare the query vector with each document vector in the cluster and retrieve only the most similar documents [30].

2.5 Latent Semantic Indexing

Latent Semantic Indexing (LSI) is an approach to automatic indexing and retrieval that is designed to overcome fundamental problems that exists in some retrieval techniques. As foundation LSI make use of the vector space model.

Users want to retrieve documents based on the conceptual content, and individual words provide unreliable evidence about the conceptual topic or meaning of a document. LSI was designed to overcome such problems that exact matching (*boolean*) techniques have. A given concept can be expressed in many ways (*synonymy*), so in exact matching the literal terms in the user's query may not match those of a relevant document. Depending on the user's contexts, needs, knowledge, linguistic habits the same information can be described with different terms. On the other hand, words can have multiple meanings (*polysemy*). Polysemy causes low retrieval precision since the terms in the user's query will literally match terms in documents that are of no interest to the user [10].

Unlike exact matching techniques, LSI is based on concept matching rather than index term matching. Matching based on concept allows retrieval of documents even if they are not indexed by query index terms. For this to be possible an underlying latent semantic structur in the data must be present. By treating the unreliability of observed term-document association data as a statistical problem it tries to overcome the deficiencies of term-matching retrieval. To estimate the latent structure and get rid of the obscuring "noise" statistical techniques, for instance in this project the method known as Singular Value Decomposition (SVD), is used. From a large matrix of term-document association data a semantic space is constructed. The original matrix is replaced with a low rank approximation by using the SVD - the SVD creates three minor matrices, respectivelly a left and a right singular vector matrix, and a diagonal matrix of singular values [22]. In the semantic space the terms and documents that are closely associated are placed near each other [10] [30]. The semantic space contain a term-concept space and a document-concept space, respectivelly the left and right singular vector matrices.

The semantic space is arranged to reflect the major associative patterns in the data, and ignore the smaller and less important influences. This results in occurrences where terms that did not actually appear in the document may still end up close to the document. These problems can be traced to three factors; (1) index terms are incompletely identified, (2) no automatic method for dealing with polysemy, and (3) each word type is treated as independent of any other.

The first factor is that the terms used to describe or index a document contain usually only a fraction of the terms that the users will try to look it up under, which induce that the index terms are incompletely identified. Reasons for this is that the documents themselves do not contain all of the terms that the users most probably will apply in order to retrieve the desired documents. An another reason is that the term selection procedures omit many of the terms intentionally, e.g. stopwords removal.

A common approach for dealing with the second factor is to use controlled vocabularies and human intermediaries to act as translators. However, this solution is reported to be extremely expensive and not always necessarily effective [10].

The third factor treats each word type independent of any other, and result in redundancy causing distorted results to. This problem makes it difficult for a user to use compound-term queries effectively to expand or limit a search [10].

Singular Value Decomposition
In order to construct the semantic space mentioned earlier, the latent semantic structure analysis starst with a matrix of terms by documents. The term-document matrix is then analyzed by Singular Value Decomposition (SVD). This analysis derive the latent semantic structure model.

The analysis begins with an arbitrary rectangular matrix with different entities on the rows and columns - in this thesis the entities are terms and documents. Next step is to decompose the matrix into three minor matrices by the SVD process. The newly created matrices contain singular vectors and singular values. Many of the components in the matrices are very small and may be ignored, leading to an approximation model that reduce the number of dimensions. The matrix are now approximated by values on a smaller number of dimensions for all of the matrices.

In relation with information retrieval, SVD can be viewed as a technique for deriving a set of uncorrelated indexing variables or factors. The terms and the documents are represented by vectors of factor values. During the dimension reduction there are possibilities for that the documents with a little different profiles of term usage is mapped into the same vector of factor values. This is all that is needed in order to accomplish the improvement mentioned earlier. It is said that by replacing individual terms with derived orthogonal factor values may help to solve all three of the problems described [10].

Algebraic representation of SVD
SVD was first introduced in 1965 by Golub and Kahan [18] as a decomposition technique for calculating the singular values, pseudo-inverse and rank of a matrix. The conventional way of doing this was to convert a matrix to a row-echolon form. The rank of a matrix is then given by the number of nonzero rows or columns of the echolon form, whichever of these two numbers is smaller. The approach of SVD is entirely different, the SVD technique decomposes a matrix A into three new matrices by the equation:

$$A = USV^T$$

where [5]:

- U is a matrix whose columns are the eigenvectors of the AA^T matrix. These are termed the *left eigenvectors*.
- S is a matrix whose diagonal elements are the singular values of A. This is a diagonal matrix, so its nondiagonal elements are zero by definition.
- V is a matrix whose columns are the eigenvectors of the A^TA matrix. These are termed the *right eigenvectors*.

V^T is the transpose of V - the transpose is to interchange rows and columns of a matrix [11].

SVD components within LSI:

A_k	= Best rank-k approximation of A		
		m	= Number of terms
U	= Term vectors	n	= Number of documents
S	= Singular values	k	= Number of factors
V	= Document vectors	r	= Rank of A

To reduce the number of dimensions the first *k* singular values are kept. The singular values are ordered in decreasing order along the diagonal of *S* and this ordering is preserved when constructing *U* and V^T. Reducing the matrices are done by keeping the first *k* rows and columns of *S*, the first *k* rows of V^T, and the first *k* columns of *U*. The equation is now modified to the following:

$$A_k = U_k S_k V_k^T$$

The dimension reduction that yields the matrix A_k is referred to as the *rank-k approximation of A* or the reduced SVD of *A*. The top *k* singular values are selected as a mean for developing a "*latent semantics*" representation of *A* that is free from noisy dimensions. In low dimensional space the latent semantics representation is a specific data structure where documents, terms and queries are embedded and compared. The latent data structure is masked by the noisy dimensions and beomes evident after the decomposition.

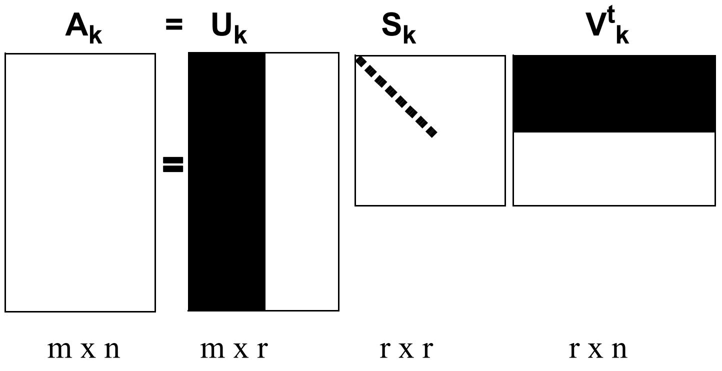


Figure 2.11: SVD matrices with rank k (picture taken from Berry [5]

SVD belongs to a class of dimensionality reduction techniques that deal with the uncovering of latent data structures, and the dimension reduction can be viewed as a noise reduction process.

In order to perform a search in the semantic space the query needs to be converted to a representative vector in the document-concept space, creating a pseudo-document vector:

$$V_q = A_q^T U_k S_k^{-1}$$

where A_q^T is the transpose of the original query vector before the coversion is performed. The search is performed in the document-concept space, which is significantly less than the term-document space represented by the A_k.

3 Research framework

In this project the main objective is to explain and develop a suitable model for storage and retrieval of molecular biological information. The proposed model in this thesis applies Latent Semantic Indexing, described briefly in the previous chapter, on the tf-idf index created during the indexing process of the collection. This chapter will present an overview of the database Online Mendelian Inheritance in Man, which is the searchable collection in this project, and its structure. Then the research framework's approach and limitations will be described.

3.1 Online Mendelian Inheritance in Man

Online Mendelian Inheritance in Man (OMIM)[6] is a free and comprehensive database, which is updated daily, of human genes and genetic disorders authored by Dr. Victor A. McKusick and his colleagues at John Hopkins University. The database is developed for the World Wide Web by the National Center for Biotechnology Information (NCBI), and contains information on disease phenotypes and genes, including extensive descriptions, gene names, inheritance patterns, map locations, gene polymorphisms and detailed bibliographies. OMIM is integrated with the Entrez[7] suite of databases and contains approximately 17 000 entries, including data on > 11 000 established gene loci and phenotypic descriptions. Such resources as locus-specific databases and GenTests are linked to from OMIM records [44]. Each entry in the OMIM database has a full-text summary of a genetically determined phenotype and/or gene and has numerous links to other genetic databases such as DNA and protein sequence, PubMed references, general and locus-specific mutation databases, HUGO nomenclature, MapViewer, GeneTests, patient support groups and many others. OMIM is an easy and straightforward portal to the burgeoining information in human genetics [21] [37] [44].

Some technical data on OMIM
Each entry in the OMIM collection is assigned a unique six-digit number whose first digit indicates the inheritance type. The available inheritance types are autosomal, X-linked, Y-linked or mitochondrial. The digits 1 and 2 are assigned to autosomal loci or phenotypes for entries that are created before May 15, 1994. The digit 3 is assigned to X-linked loci or phenotypes, and 4 is assigned to Y-linked loci or phenotypes. 5 is assigned to mitochondrial loci or phenotypes, and the digit 6 is assigned to autosomal loci or phenotypes for entries created after May 15, 1994 [21].

Whether the entries contain information on genes, phenotypes or both, the Mendelian Inheritance in Man (MIM) entries are categorized by using a symbol that precedes the MIM number. There are five basic symbols used to categorize the MIM entries, they are; an asterisk (*), a number symbol (#), a plus sign (+), a percentage sign (%), and a caret symbol (^). The asterisk is used before an entry number to indicate that the gene is of a known sequence. A number symbol is used before an entry number to indicate that it is a descriptive entry, usually of a phenotype, and does not represent a unique locus. In the first paragraph of the entry a reason for the use of the number symbol is given. Plus signs are used to indicate that the entries contain description of gene of known sequence and phenotype. To indicate that the entry describes a confirmed Mendelian

6. http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=OMIM
7. http://www.ncbi.nlm.nih.gov/Entrez/

phenotype or phenotypic locus for which the underlying molecular basis is not known, a percentage sign is used before the entry number. Entries with no symbol before the entry number generally indicates a description of a phenotype for which the Mendelian basis, although suspected, has not been clearly established ord that the distinctness of ths phenotype form that in another entry is unclear. Caret symbols are used before the entry number to indicate that the entry was removed from the database or moved to another entry as indicated [21].

An entry number consist of a six-digit number, allelic variants are given a 10-digit number, and are maintained within the relevant gene entry. The 10-digit number given to the allelic variants, the six first digits are the parent entry followed by a decimal point and a unique four-digit variant number (e.g. entry 253200 MUCOPOLYSACCHARIDOSIS TYPE VI with the allelic variants .0001 INTERMEDIATE, ARSB, GLY137VAL, and .0002 SEVERE, ARSB, CYS117ARG, etc.). Selected mutations are included as specified subentries, and criterias for inclusion are: first severeal mutations, unusual pathogenetic mechanism and distinctive inheritance (e.g. dominant with some mutations, recessive with other mutations in the same gene). Most of the allelic variants represents disease-producing mutations. A few polymorphisms are included, mainly those that show statistical association with particular common disorders [21].

Each entry in the OMIM collection may contain of 10 various fields, the fields are; NO (entry ID), TI (title), TX (text/body), AV (allelic variants), SA (see also), RF (references), CN (contributors), CD (creation date), ED (edit history), and CS (common symptoms) [40].

3.2 Approach and limitations

In order to achieve the goal given in this project one must analyze how to treat the data in the OMIM collection. The OMIM collection is one big text file that needs some preprocessing before it is indexed. First the collection get splitted into documents based on the *RECORD* fields, so each record is transformed to a corresponding text documents. This is done in order to have more control of the records and it is more manageable during testing. Each document in the collection is treated as a whole in order to get a superior foundation of the content. One limitation of treating a document as a whole is that the various field's importance will not arise, but rather suppressed.

The search engine implemented in this project is made from scratch, but utilizes some of the indexing features of the Apache Lucene package. Java is used to implement the search engine, this is done because Lucene and mainly JAMA is primarily implemented in Java. The search interface is implemented in HTML with snippets of JavaScript to handle the dynamic content. JavaScript may cause some problems, due to the user's preferences in the web browser. In newer web browsers the user can disable JavaScripts and this search engine requires that JavaScript is enabled.

4 Theoretical solution

In this chapter alternative indexing solutions and clustering algorithms will be discussed. The different solution alternatives will be compared. Finally, the chosen solution will be described and substantiated.

4.1 Alternative indexing solutions

In addition to the standard LSI approach new approaches has emerged the last decade. Two approaches that have emerged are the Probabilistic Latent Semantic Indexing and Variable Latent Semantic Indexing. In this section both of the different LSI solutions will be discussed according to their advantages and drawbacks.

4.1.1 Probabilistic Latent Semantic Indexing

Being based on a statistical latent class model for factor analysis of count data, the Probabilistic Latent Semantic Indexing (PLSI, introduced in 1999 by Thomas Hofmann [23]) is a novel approach to automated document indexing. PLSI concerns non-negative matrix factorization.

The utilized model, fitted from a training corpus of text documents by a generalization of the Expectation Maximization algorithm, is able to deal with domain-specific synonymy as well as with polysemous words. Having a solid statistical foundation, the probabilistic variant defines a proper generative data model, in contrast to standard Latent Semantic Indexing (LSI) by Singular Value Decomposition. Substantial performance gains over direct term matching methods, as well as over LSI, was indicated during retrieval experiments on a number of test collections. Especially the combination of models with different dimensionalities proved to be advantageous [23].

A statistical model called aspect model is the core of PLSI. The aspect model is a latent variable model for general co-occurrence data which associates an unobserved class variable with each observation [12] [23]. A latent class model (LCM) in statistics relates a set of observed discrete multivariate variables to a set of latent variables [26]. It is a type of latent variable model, being called a latent class model because the latent variable is discrete. A class is defined by a pattern of conditional probabilities that indicate the chance of variables taking on certain values.

The observed variables are statistically independent within each latent class. This is an important aspect. Observed variables are usually statistically dependent. Independence is restored by introducing the latent variable, thus, within classes variables are independent (local independence). The association between the observed variables can be said to be explained by the classes of the latent variables [35].

Standard LSI stems from linear algebra and downsizes the occurrence tables (usually via SVD), while PLSI is based on a mixture decomposition derived from a latent class model. This gives a more principled approach with a solid foundation in statistics.

Due to a very large number of mixing distributions, the aspect model used in the PLSI is reported to have severe overfitting problems [28]. The number of parameters grows linearly with the number of documents. A generative model is not allowed to have parameterizations. PLSI is not a generative model, hence, not providing a generative procedure for documents. A common way to turn it into a proper generative model is by marginalizing the model by summing over all strings in the training set [28].

Other deficiencies; (1) the aspect model in PLSI does not perform very well in supervised settings, (2) latent models are frequently not identifiable, i.e. their optimal parameters are not unique. Using the aspect model in experiments with supervised classification showed that its performance was rather poor [27]. PLSI does neither provide a way of assigning likelihoods to instances of the observed variables that do not occur in the training corpus [6].

4.1.2 Variable Latent Semantic Indexing

Variable Latent Semantic Indexing (VLSI) [9] is a new query-dependent low-rank approximation that minimizes approximation error for any specified query distribution. It is possible to tailor the LSI technique to particular settings with this tool, which in turn often results in vastly improved approximations at much lower dimensionality. A series of experiments on classical corpora validates this method. With an order of magnitude fewer dimensions, VLSI typically has similar performance to LSI. From the experiments a form of query-dependent or variable LSI is derived by characterizing the queries likely to arise through a probability distribution. The fact is, Dasgupta et al. [9] use a general model based on where there is a co-occurrence matrix on pairs of query terms. When the co-occurrence matrix is the scaled identity matrix, this approximation reduces to the standard LSI approximation for the special case.

Experiments show, that VLSI dramatically outperforms LSI on retrieval effectiveness for any given number of dimensions in the low-dimensional approximation. An alternative way of viewing these results: for any quantitative level of retrieval effectiveness, the number of dimensions in the low-rank approximation is dramatically lower for VLSI than for LSI. As an example, whereas LSI on text corpora appears to require hundreds of dimensions in the approximation, a few tens of dimensions often suffice for VLSI.

The lexicon used in this method is built as follows. To begin with, the text is extracted from the files in which the collection is delivered. Then, by splitting at whitespace boundaries the tokens are extracted. Tokens are then Porter-stemmed and case-folded, punctuation is removed, and a standard 416-word stopword list is employed to remove any stopwords. This results in a dictionary of terms.

Based on this dictionary, the corpus is scanned. This produces a term-document matrix of counts, for which is the number of occurrences of term *i* in document *j*. Two other matrices in which the experiments are performed are derived from this matrix. The first matrix is the Boolean matrix, which is produced by setting the cell to 1 of the corresponding cell in the matrix of counts is non-zero, and to zero otherwise.

The second matrix is the weighting matrix based on the Okapi weighting. This matrix is a weighted version of the term-document matrix. Then, a SVD is performed on the Boolean and Okapi matrices as a baseline. With a family of query distributions, VLSI is applied to each of these matrices [9].

4.2 Clustering methods

Because not all relevant items may be retrieved and the whole document space has to be search, the retrieval effectiveness and efficiency are low. By grouping similar documents into cluster these disadvantages can be overcomed [30]. In the following two cluster methods will be described. Both of the methods are based on all pairwise document similarities and assembles similar items into common clusters. In the approach based on pairwise similarities, each document is represented as a document vector as in the vector space model. Then the similarity between each pair of documents is calculated.

4.2.1 Standard k-means

The k-means clustering algorithm was developed by MacQueen [33], and is one of the simplest and the best known unsupervised learning algorithms that solve the well-known clustering problem. The procedure follows a simple and easy way to classify a given set of data through a certain number of clusters fixed a priori, where each cluster is defined by a centroid. In order to best make use of the k-means method is to place the centroids as far away from each other as possible [15].

Next step is to add points, in this case document vectors, to the clusters. For each document vector a similarity between the document and each of the clusters' centroids. The document is added to the cluster which the centroid is the most similar to. When all of the document vectors have been added to a cluster the first step is completed. The centroids are then recalculated, meaning a mean vector is created for each of the clusters, based on the document distribution from the previous step. With the new centroids, a new binding has to be done between them and the same data set, creating a loop. The loop is repeated until the centroids do not changes location any more, meaning the centroids no longer moves.

The k-means algorithm is composed of the following steps [34]:

1. *Place k points into the space represented by the objects that are being clustered. These points represent initial group centroids.*
2. *Assign each object to the group that has the closes centroid.*
3. *When all objects have been assigned, recalculate the positions of the k centroids.*
4. *Repeat steps 2 and 3 until the centroids no longer move. This produces a separation of the objects into groups from which the metric to be minimized can be calculated.*

The procedure will always, or mostly, terminate, but the algorithm does not necessarily find the most optimal configuration, for that the algorithm is too sensitive to the initial centroid selection. To reduce this effect the algorithm can be run multiple times.

K-means is a greedy algorithm that partitioning *n* samples into *k* clusters so as to minimize the sum of squared distances to the cluster centers. The algorithm does have some weaknesses [34]:

- The initialization of the means is not specified. One popular way to start is to randomly choose *k* of the samples.
- The initial values for the means affects the result and suboptimal partitions are frequently found. A standard approach is to try a number of different starting points.
- It can happen that the set of samples closest to a particular cluster is empty, so that the cluster cannot be updated.
- The results depend on the metric used to measure the similarity between the samples and the clusters. A popular solution is to normalize each variable by its standard deviation, though this is not always desirable.
- The results depend on the value of *k*.

The latter problem is especially troublesome, mainly because there are no easy way of knowing how many clusters exist, and whether a *k* value of 3 is a better choice than 2, and etc.. To determine the optimal number of clusters for any given data set is a difficult task. One approach is to compare results of multiple runs with different number of clusters and choose the one that give the best result according to a given criterion, although by increasing the *k* value may increase the risk of overfitting [34]. However, Fraley and Raftery [16] have proposed an analysis to determine the best choice of number of clusters and which clustering method.

There are some variants of the k-means algorithm, Jing et al. [25] added a new step in the clustering process to automatically calculate the weights of keywords in each cluster so that the important words of a cluster can be identified by the weight values. This extension enables the k-means clustering algorithm to cluster high dimensional text data in subspaces of the keyword features, so that sparsity of text data can be effectively handled. The performance of the k-means clustering process is preserved because it does not increase the number of iterations.

The k-means clustering algorithm is known to be efficient in clustering large data sets. The recent development of the new k-means type algorithms with variable weighting ability enables the efficient k-means clustering process to discover clusters from subspaces that are identified by the weights of variables.

4.2.2 Two step k-means

Two step k-means is a proposed modification of the standard k-means by the author. This version of the well-known clustering method has a goal of beeing less greedy than the standard version concerning the distribution of the documents. Unlike the standard version, the Two step k-means consists of two modes, a local and a global mode. In the local mode "local" centroids are calculated based on the distributed documents that fulfil a requirement that the similarity exceeds a marginal value, for instance the marginal value can be 0.9. An another requirement is that a document cannot be added to a cluster if it exceeds the marginal value compared with more than one cluster - if the document does not exceed the marginal value it will not be added to any of the clusters. This is done in order to force the centroids away from each other and by that give the clustering algorithm, hopefully, a better foundation for clustering the documents.

The two step k-means is composed of the following steps:

1. *Place k points into the space represented by the objects that are being clustered. These points represent initial group centroids.*
2. *If the similarity exceeds a given ratio and is only suitable for one group, assign the object to the cluster. Objects that are suitable for more than one group will not be assigned to a cluster.*
3. *When all objects have been tried assigned, recalculate the positions of the k "local" centroids.*
4. *Repeat steps 2 and 3 until the centroids no longer move. This will force the centroids away from each other and the clusters to be not so greedy.*
5. *Repeat steps 2 and 3 without the ratio and the constraint on number of suitable groups until the "global" centroids no longer move.*

The Two step k-means algorithm is not significantly more complex than the standard k-means, but the number of steps have increased by one and therefore causing the algorithm to increase the runtime. The runtime is approximately doubled compared with standard k-means.

Deficiencies with the method:

- Too low ratio may cause clusters without objects, this can be caused by initial centroids that are too close to each other. This is unfortunate because these clusters will not "get back in the game" again. The initial selection of the centroids is also here crucial.
- When the local centroids have been found, the algorithm start being greedy again.

4.3 Chosen solution

The chosen solution is the standard LSI method. Standard LSI have a longer time of service than for instance PLSI and VLSI. An another reason why standard LSI was chosen instead of PLSI, is that there are some contradictory reports of its performance. VLSI is a novel approach and there are not many reports about it, which also led to the choice of standard LSI. The author wanted to test the clustering performance on a known and well-reputated foundation, therefore the standard LSI method was selected.

Flat clustering is selected in this project since hierarchical clustering approaches are far too limited to capture the rich semantics of most document sets [10]. The k-means clustering method is a simple flat clustering approach and suits well in this project.

5 Implementation

In this chapter the implementation will be described. The necessary classes from the Apache Lucene package and the Java Matrix package will be explained in detail in chapter 5.1. The overall structure of the application is shown in figure 5.2 and described in chapter 5.2, prior to the indexing, clustering and searching process.

The following subchapters describe the technical information and usage of the various Lucene and JAMA classes in this project. In chapter 5.6 the application logic and functionalities is explained. The final subchapter describes and demonstrates how querying is executed, details of the user interface, and how it works in practice.

5.1 Technical information

The full-featured search engine library Apache Lucene [31] and the Java matrix package JAMA [24] has been imported and used in this project. This subchapter will give a brief description on how these two Application Programming Interfaces (APIs) work and how they are used in this project.

5.1.1 Lucene

The Apache Lucene is a high performance, scalable open-source search engine package written in Java. Lucene is not a ready-to-use appliction, but is a software library that provides core API for full-text indexing and searching functionalities to be used in the developer's applications. The Lucene API contains functions for indexing and searching, converters of various types of data, etc.. It is up to the developer to decide how to manage the indexing, searching, deal with user queries, and representing them and the results to the users of the application [19].

In every search engine the main concept is the concept of indexing, where the original data is processed into a searchable cross-reference lookup in order to achieve searching at great speed. The processing of the original data is done by analyzers. Analyzers converts documents into a list of words that are suitable as index terms. This is done by removing unusable words like stopwords and performing stemming on the remaining words. The analysis process will be described in more detail later. Document class instances defines the documents when they are added to the Lucene index, and Fields consists of name and value pairs.

The structure of the Lucene index is called an inverted index. This means the content of the documents that has been analyzed and the words suitable as index terms, are indexed as field name and value pairs. The Fields contains many terms that points to corresponding documents. A general structure of an inverted index is [19] [30] [31]:

Term *i*: Record no., Paragraph no., Sentence no., Word no.

When searching an inverted index the documents are retrieved by searching the fields and their values.

In order to perform indexing with the Lucene API, the basic Lucene classes needed to perform it are:

- IndexWriter
- Analyzer
- Directory
- Document
- Field

The main component of the indexing process is the IndexWriter. IndexWriter creates a new index to a desired path and adds documents to the index. The developer can decide which path to save the index, what sort of analyzer to be used, and whether to create a new index from scratch or append documents to an existing one.

The location of the index is represented by the Directory class. This class have originally two subclasses; FSDirectory and RAMDirectory. The FSDirectory stores the Lucene index on disk, while the RAMDirectory stores it in memory. Both of the subclasses have identical interface, but RAMDirectory is more useful for small indices and FSDirectory for large indicese due to their sizes and the the amount of storage space available.

The Analyzer class provides some preprocessing of the text before it is indexed. Words that are not fitted to be regarded as suitable index terms are eliminated from the collection, the rest is added to the index, e.g. stopwords removal. How the analyzer in this project is implemented will be described later.

In the index the fields are represented by the Document class. The Document class consists of one or more named fields that represent the document or meta-data (e.g. author, title, subject, etc.) associated with the document. Metadata is stored separately as fields of the current document as name and value pairs, e.g.:

> field name: "*author*"
> field value: "*William Shakespear*"

The named fields in the index is embodied in a class called Field. What type of field one should use depends on the usage of the field. When a user search for documents a request are queried against the fields in the index to retrieve documents.

The summary of the indexing steps in this project is shown in figure 5.1. The individual steps in this project will be described in more detail in the following subchapters.

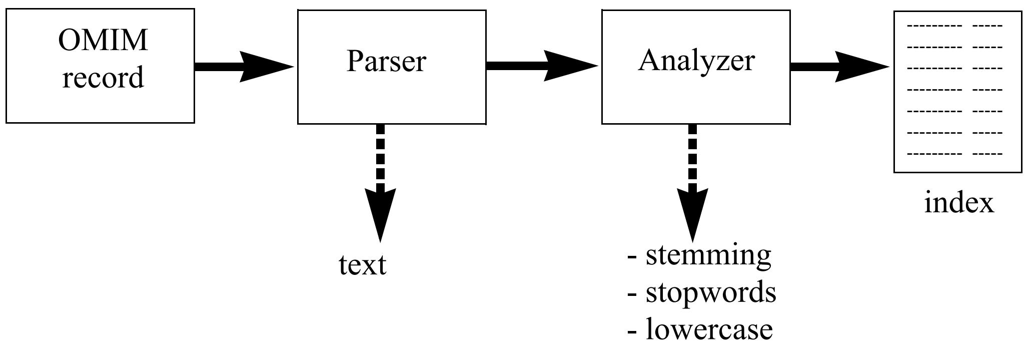


Figure 5.1: The indexing process

Lucene comes with classes for searching - this project make only use of Lucene in the indexing part, but applies Lucene's QueryParser class on the query terms given by the user. The QueryParser class requires an analyzer as parameter, the analyzer break pieces of the query into terms. The analyzer stems and remove possible stopwords from the query.

Information about how Lucene is implemented in this project will be described in chapter 5.3 about the indexing process.

5.1.2 JAMA

The JAMA package is an API for performing basic linear algebra tasks developed for Java. JAMA provides classes and functions for creating and manipulating real, dence matrices. The main class of JAMA is the Matrix class. On the Matrix class fundamental operations of numerical linear algebra, and various get and set operations are provided. In addition are basic arithmetic operations, such as addition and multiplication, provided.

The package provides the following five fundamental matrix decompositions:
- Cholesky Decomposition of symmetric, positive definite matrices
- LU Decomposition (Gaussian elimination) of rectangular matrices
- QR Decomposition of rectangular matrices
- Eigenvalue Decomposition of both symmetric and nonsymmetric square matrices
- Singular Value Decomposition of rectangular matrices

The five decompositions produce pairs or triples of matrices, permutation vectors, and the like. The decompositions are accessed by the Matrix class, and provides operations to compute solutions of simultaneous linear equations, determinants, inverses and other matrix operations[8].

In this project the only decomposition needed is the Singular Value Decomposition. The use of JAMA's Matrix and SVD classes will be described respectively in chapter 5.3 and 5.5.

8. For capabilities of JAMA, see appendix B

5.2 Architectural design

The overall system structure are shown in figure 5.2. The structure consists of two main sections: preprocessing of documents and the main application, which is further divided into two subsections; Lucene indexing and OMIM indexing and.

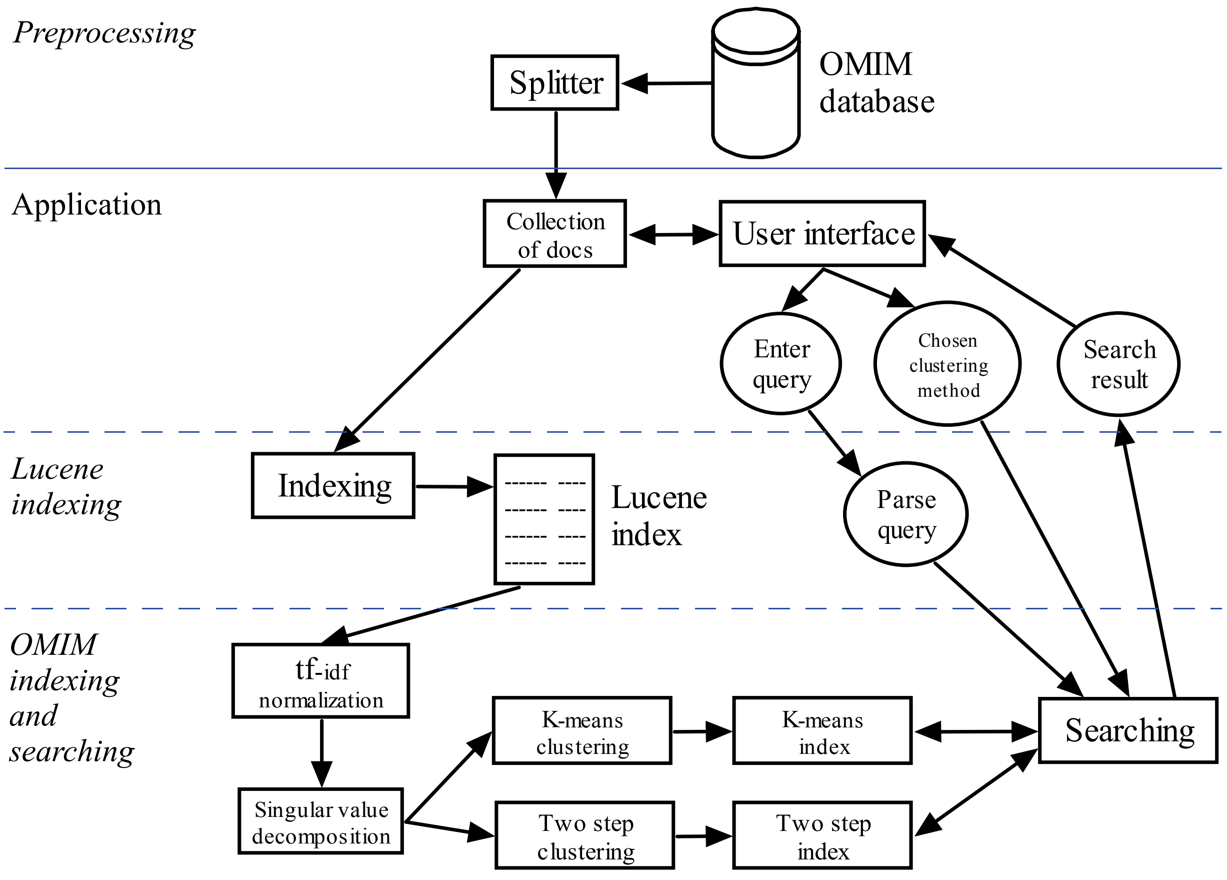


Figure 5.2: Architectural overview of the system

The first main section preprocesses the OMIM database. The OMIM database is a 100 MB text file containing approximately 14 000 records. By splitting the database into small text files, one file per record, where the record number becomes the document's name, it is more manageable and controllable by doing it this way. The data in each record are unaffected from the splitting process.

In the second main section, the first subsection make use of Lucene to index the collection of OMIM files. It lowercases the text, remove stopwords and stems the remaining words. This subsection may be viewed as an intermediate section, since it in one fashion does some preprocessing and the other a part of the main application.

The second subsection is the information retrieval application itself. All of the necessary information from the Lucene index is retrieved and reconstructet with the purpose to apply SVD. When the clustering processes are finished new indices are created - one for each clustering method. The searching process make use of the indices created during the clustering process and ranks the result, showing the most relevant documents.

A more detailed description of the indexing, clustering and searching processes follows in the next subchapters.

5.3 Indexing process

The indexing process may be divided into two parts: a Lucene part and a LSI with clustering part. The Lucene part is used as a preprocess to LSI. The reason why the Lucene and the LSI implementation is splittet from each other is because to implement LSI in Lucene some extensive transformations has to be made. To implement an LSI friendly structure from scratch rather than modify the structure of the Lucene index is easier and less time-consuming. The trade-offs here are speed and latency, storage requirements and extensibility. The trade-offs and their results will be discussed in chapter 6.4.

The indexing process begins by creating an FSDirectory to store the index created by Lucene, for each document an analyzer is applied to remove stopwords and stem the words, and an IndexWriter to write the index to file.

```
this._directory = FSDirectory.getDirectory(strIndexDirectory, true);
this._analyzer = new OMIMAnalyzer(StopWords.stopTable);
this._writer = new IndexWriter(this._directory, this._analyzer, true);
```

The source code snippet above instantiates the Lucene indexing process. The `FSDirectory.getDirectory` have two parameters; strIndexDirectory is the desired path to store the index, and a boolean to create or erase existing contents if true. OMIMAnalyzer is a custom-made analyzer that have a stopwords table as parameter. The IndexWriter takes three parameters; the desired path to store the index, analyzer to analyze the stream of text, and a boolean variable to determine whether to create a new index or modify an existing one.

For each document in the collection a Document instance is added to the index with the code:

```
this._writer.addDocument(FileDocument).Document(contents[i]));
```

The content in each document is treated as a whole, alternatively one may divide it in sections, e.g. one section for title, authors, etc.. Treating the content in a document as a whole is done in order to get a best possible overall similarity foundation of the documents in the collection. This will be discussed later.

As mentioned above an analyzer is applied to the stream of words from the documents. The custom-made analyzer used in this project, the OMIMAnalyzer, is built up in the following manner:

```
return new PorterStemFilter(new OMIMStopfilter(new
                              LowerCaseTokenizer(reader), stopWords));
```

First, the analyzer lowercases the words, this is done in order to make the system case "insensitive". Second, stopwords are removed - the OMIM collection provides a set of stopwords that have been used in this project, see appendix A for a list of the stopwords. Third, and finally, a Porter stemmer is applied to reduce the suitable index terms to their roots.

When there are no more documents in the collection to index the Lucene index is optimized, meaning merge all segments together into a single segment. After the optimization the term frequencies are transfered to a newly created Matrix object - the term frequencies are normalized before they are added to the matrix, the formula for normalization is described in chapter 2.4. The matrix consists of *m* number of index terms and *n* number of documents (an *m* x *n* matrix), and may become very large and therefore require a great deal of memory storage. During the transfer of the term frequencies a document and a term overview is written to file in order to keep controll of the term and document frequencies. When the transfer from the Lucene index to the matrix is done, JAMA's SVD operations are applied:

```
SingularValueDecomposition SVD = new SingularValueDecomposition(this._X);
this._U = SVD.getU();
this._S = SVD.getS();
this._V = SVD.getV();
```

The snippet above splits the matrix into three minor matrices, respectively a *left* (`this._U`) and *right* (`this_V`) *singular vector* matrix and a diagonal matrix of *singular values* (`this._S`) [10] - `this._X` is the original matrix created when the term frequencies are transferred from the Lucene index.

To reduce the matrices further *rank k* is applied to the SVD matrices, this result in a left singular vector matrix with the dimension *m* x *k*, a right singular vector matrix with *k* x *n*, and a diagonal matrix with *k* x *k* - the diagonal matrix is after the reduction inversed. The choice of *k* will be discussed later. The left and the inversed diagonal matrices are used during the search process, and after the reduction the SVD matrices are stored to file. Now that the index fulfil the LSI model described by Deerwester the clustering process will be described next.

5.4 Clustering process

As mentioned earlier this thesis applies two clustering methods to the LSI index created in the previous subchapter. The applied clustering methods are the standard k-means and a modified version of it called two step k-means.

The standard k-means algorithm is very basic and easy to implement, but it is a greedy algorithm. It is also very dependent of the location of the initial centroid vectors. Two step k-means is a proposed initiative to make the k-means algorithm less greedy and not so dependent of the initial centroid vectors, but keeping the simplicity of the algorithm intact. The following paragraphs will describe the implementation of the two clustering methods used in this project.

K-means implementation
First, the KMeans class reads the index statistics data, such as number of documents, number of index terms and the value of *k*, from file. Based on these data the reduced right singular value matrix is reconstructed, which is, according to Deerwester, the documet-concept space [10].

Second, to determine the initial centroid vectors each centroid receive the vector of a document. The allocation of the centroid vectors are made by guessing - the documents were evaluated by their size, on the basis of a large document contains more index terms than the small ones (cf. the documet's vector).

After the clusters have been created the population of the clusters begins. For each document vector a similarity between the document and the centroid vectors is calculated. The document is added to the cluster with most similarity. This is repeated until the centroid vectors no longer moves. Finally the clusters are written to files with a corresponding cluster list. The cluster list contains information about the clusters and their centroid vectors.

Two step k-means implementation
The TwoStepKMeans class extends the KMeans class, meaning it works in the same fashion. The method differs from the standard method by the introduction of a local and a global centroid vector. A restriction is also introduced in this method, the restriction applies on the allocation of documents to clusters. To be added to a cluster in "local mode" a document can not be suitable for more than one cluster and exceed a marginal value set by the developer.

In local mode the clustering method sorts out documents that have lower similarity than the given marginal value. Combined with the requirement of that a document can only be suitable for just one cluster, the intended outcome is to drive the clusters away from each other, cf. Faraoun and Boukelif [15].

When the centroids no longer move in local mode, it switch to global mode. Global mode works just like standard k-means but the initial centroid vectors are different because the centroids have been driven away from each other.

5.5 Searching process

The searching process make use of Lucene's Query and QueryParser classes. Each query written by the user have to utilize the same analyzer as the one in the indexing process, the OMIMAnalyzer. A user query is parsed with a QueryParser instance:

```
        this._analyzer = new OMIMAnalyzer(StopWords.stopTable);
        this._parser = new QueryParser("query", this._analyzer);
        this._query = this._parser.parse(userQuery);
```

The query parser remove stopwords and stem the words provided in the query to their roots, cf. the indexing process. Stemming is important here because the query must be in accordance with the index terms, meaning the query and the index terms must have the same basis when performing a match between them.

When the query is parsed, the query vector is created. The query vector is created by traversing the index term overview and the positions of matching terms are stored. Matching terms are found by matching the user query terms as a substring to the index terms, e.g. the user query “charac” matches all the words containing “charac”. Without the possibility to match a query as a substring of an index term, it requires that the user knows how the word are written or what sort of words the index contains (cf. exact match), or utilize an another matching algorithm.

To perform a search the query vector must be converted to the document-concept space described earlier. This is done by multiplying the query vector with the left singular vector matrix and the inversed diagonal matrix, which is reconstructed from file:

```
        conQuery = queryT.times(uk.times(skinv));
```

This formula converts the query vector to a representative vector in the document-concept space, and corresponds to Deerwester’s representation for pseudo-documents[9]:

$$D_q = X_q{}^t T S^{-1}$$

Where *T* respectively is the left singular vector matrix, S^{-1} the inversed diagional matrix, *D* is the right singular vector matrix, and $X_q{}^t$ the query vector. When the query have been converted the similarity between the query and the clusters are calculated - the similarity measure is the cosine coefficient described earlier. This determines which cluster is most similar to the query.

For each document in the cluster, similarity is calculated. Only documents that exceeds a marginal value of similarity are added to the result set, e.g. 0.80. The intended use of the marginal value is to omit documents that are not so relevant according to the user’s query (cf. Lu [30]).

Since there are two clustering methods available, the user have to check off which clustering method to be used during searching. Based on which clustering method is checked off the application retrieve the index for that method.

5.6 Application logic and functionalities

The main functionalities that the search engine provides are indexing biological data provided by OMIM and searching them. The original OMIM records are stored in a repository. The application logic, indexing and clustering the documents, and the searching process according to a given query, is explained in following paragraphs using sequence diagrams.

The first sequence diagram is for the indexer application and shown in figure 5.3. Firstly, the indexer class is instantiated. The indexer instance triggers the Lucene indexer implementation.

9. The formula provided by Deerwester, another representation of the same formula is provided in chapter 2.5.

The Lucene indexer has the main logical methods for retrieving data from the OMIM collection and indexing them.

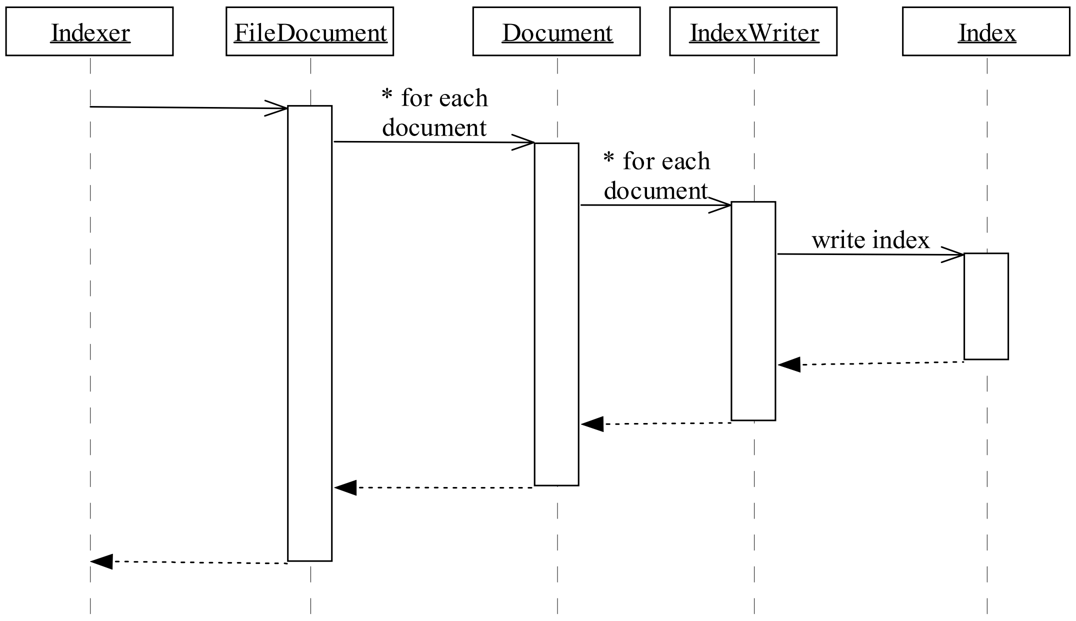


Figure 5.3: Sequence diagram of the indexing process

The content of each document is parsed and analyzed, the words suitable as index terms are extracted. The FileDocument class creates a Document instance, that contains the terms which are generated by the FileDocument, for each document in the collection. After the document have been parsed, all of the Document objects are sent to the IndexWriter object. The IndexWriter object performs the indexing operation.

The indexing operation is performed by writing all of the Document instances into the index. The index is created by the program in a file system. This operation is repeated until all of the Document instances are added to the physical index file on disk or memory. When the indexing is finished, the index is optimized, closed and returned to the Indexer application.

The second sequence diagram, see figure 5.4, illustrates the transformation from a Lucene index to a LSI index. A JAMA Matrix object is instantiated and the term frequencies located in the Lucene index are transferred to the matrix. Before the frequencies are added to the matrix physically, they are normalized as described earlier in chapter 2.4.

When all of the term frequencies have been transferred and normalized, a Singular Value Decomposition object is created. The Singular Value Decomposition object create the left and right singular value matrices, and the diagonal matrix. Each of the newly created matrices are reduced by performing *rank k*. The diagonal matrix is inversed before the matrices are written to file, one regular text file per matrix, and returned to the Indexer.

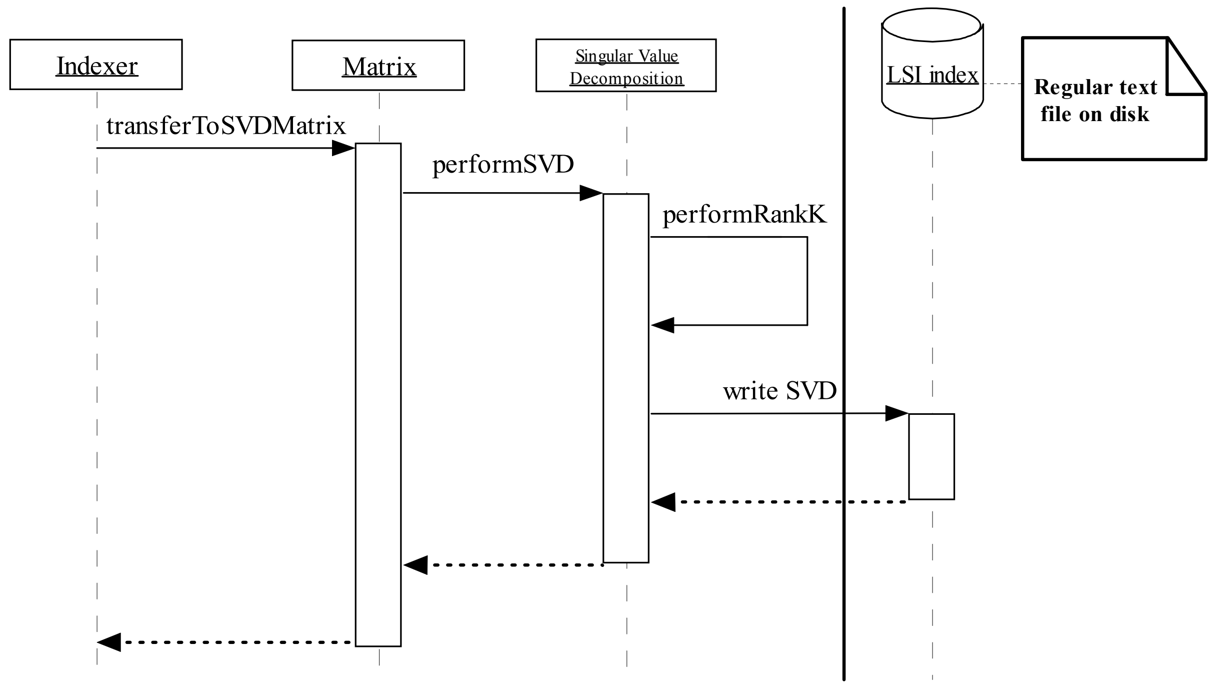


Figure 5.4: Sequence diagram of the LSI indexing process

The third sequence is a general sequence diagram of the clustering process, see figure 5.5. A clustering class requests SVD data from files, and creates a Matrix object to keep the SVD data - the only SVD data needed is the data of the right singular value matrix.

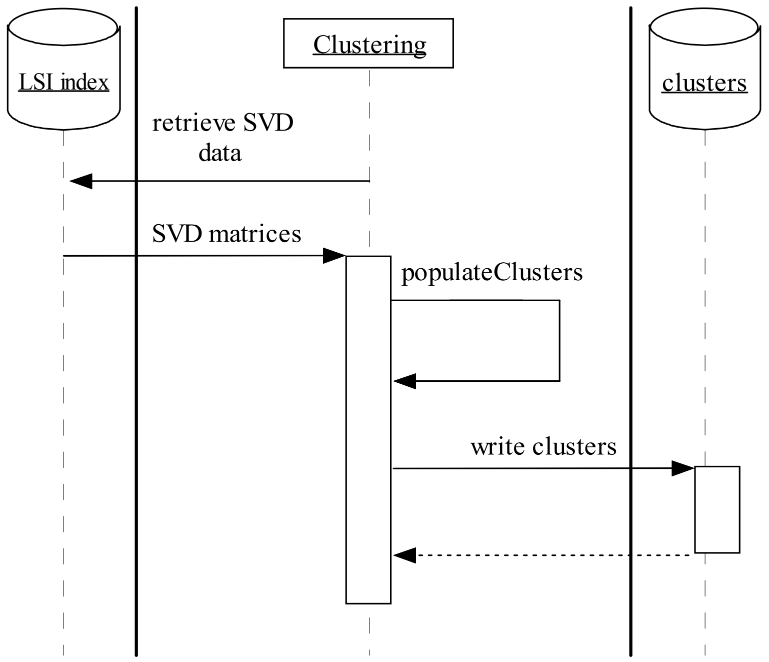


Figure 5.5: Sequence diagram of the clustering process

The clustering object populates the clusters until the centroid vectors no longer moves - in the Two Step K-Means clustering method the population is bisected, cf. local and global mode. The sequence is therefore repeated once for each mode.

When the centroids no longer moves each cluster and a superior cluster list is written to file, before it is returned to the clustering application.

The fourth, and final, diagram is the sequence diagram of the searching process, see figure 5.6. The searching process starts with a user that types in a query. This query is sent to a QueryParser object that applies the OMIMAnalyzer on the user query. An array of parsed query terms are returned to the Searcher.

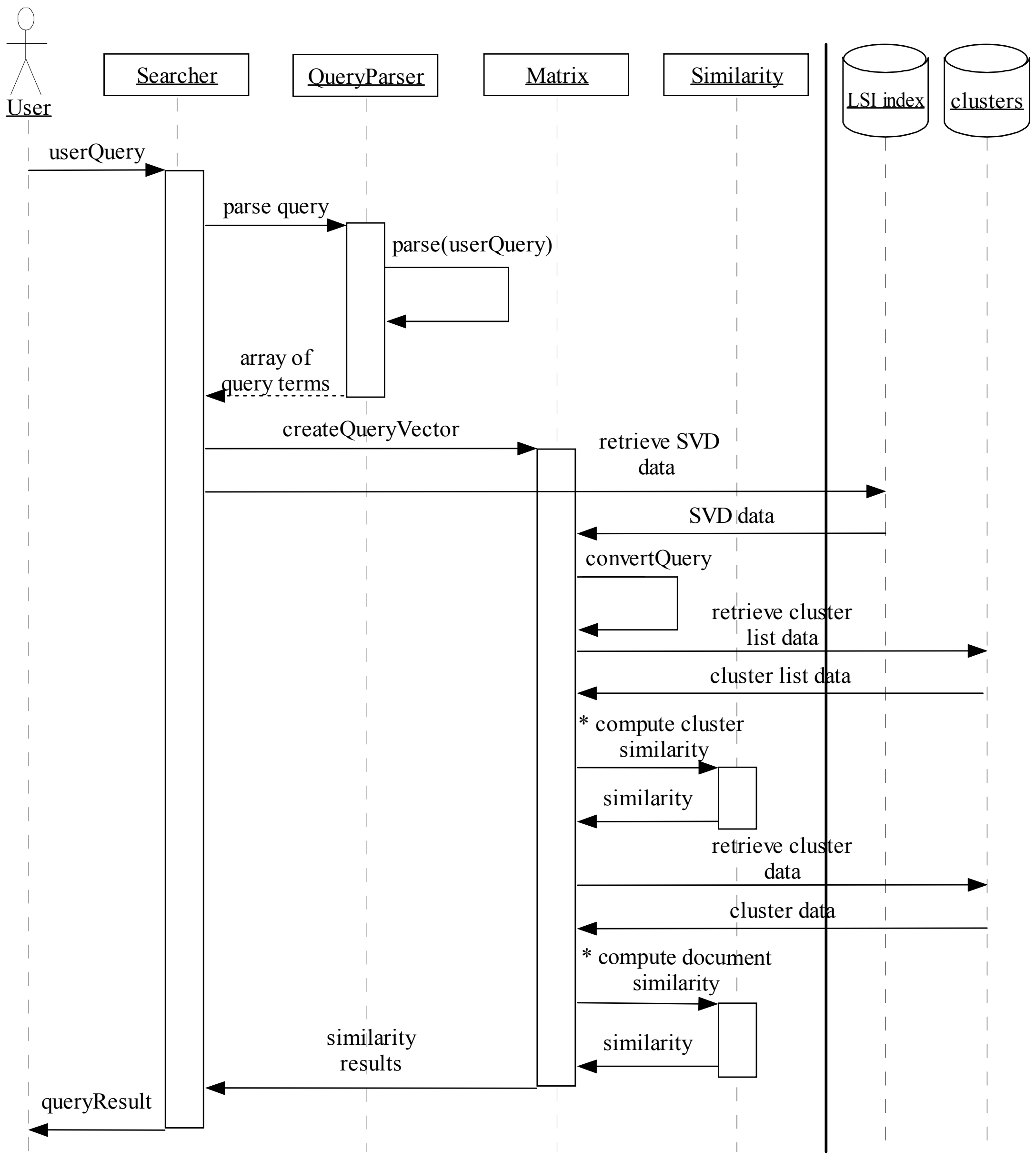


Figure 5.6: Sequence diagram of the searching process

Matrix objects are created to keep the query vector and the SVD data retrieved from file. To compute similarity between the user query and the centroid vectors and the document vectors, the query is converted to suit the document-concept - formula described in chapter 5.5. After the query has been converted cluster list data is retrieved from repository. For each centroid vector the similarity between it and the query vector is calculated. Based on the similarity, the cluster with most similarity is retrieved from the repository. For each document in the cluster similarity is calculated. The documents which are more similar to the query than a marginal value is returned to the Searcher application that presents the result for the user.

5.7 User interface

In this project the user interface is kept simple, see figure 5.7. The interface contain classical funtions like (1) a text box for entering user queries, (2) selection of clustering method, and (3) a search button - the clustering method selection is provided in this project to easily change clustering method.

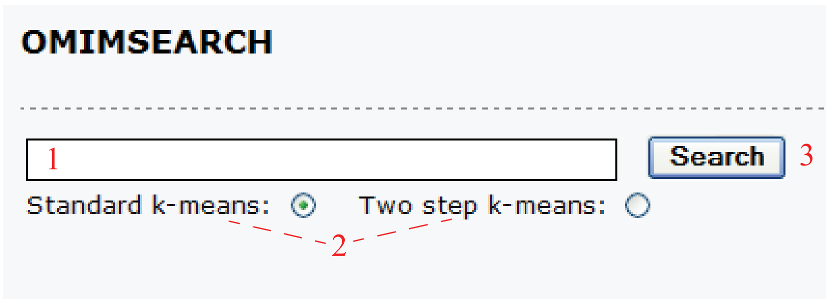
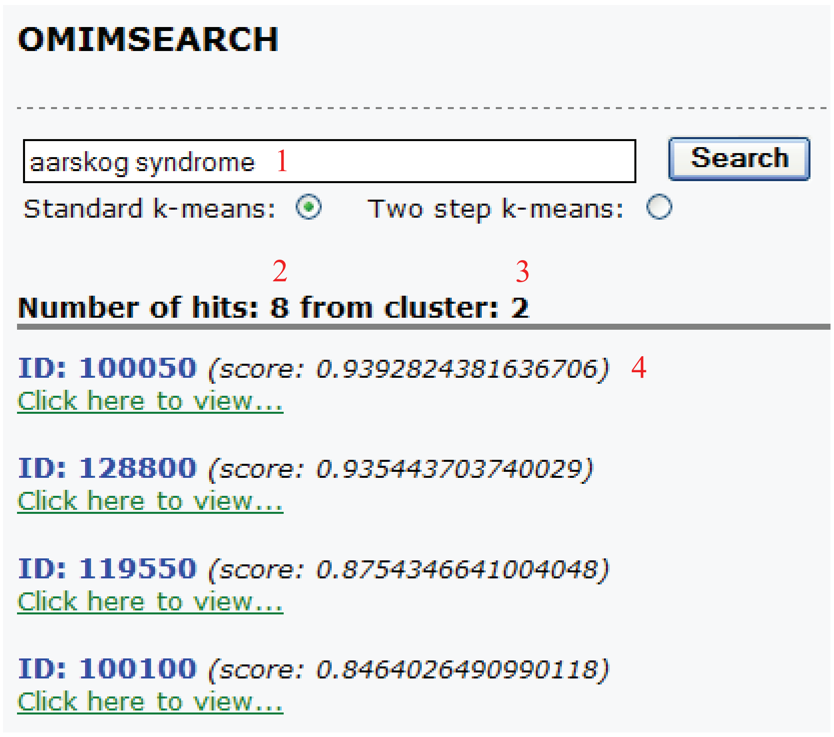


Figure 5.7: Search box

If the user wants to use the Two step k-means method the user must select it before executing the search. The standard k-means algorithm is selected by default when doing search.

Figure 5.8 illustrates an executed search using the clusters created by the standard k-means algorithm with the (1) user query "*aarskog syndrome*". For each search request number of hits (2) and which cluster the documents are retrieved from (3) is presented to the user.

The result list presents the retrieved documents by an identificator (*ID*), similarity (*score*) and an Uniform Resource Locator (URL) to the physical document on disk (4). The documents are sorted decreasingly based on the similarity score. Although the similarity score is redundant, since the result set only contain documents with similarity above a given marginal value, it is represented to give the user a hint of how relevant the document is to the user's query.


OMIMSEARCH

aarskog syndrome 1

Standard k-means: [✓] Two step k-means: ○

2 3

Number of hits: 8 from cluster: 2

ID: 100050 (score: 0.9392824381636706) 4
Click here to view...

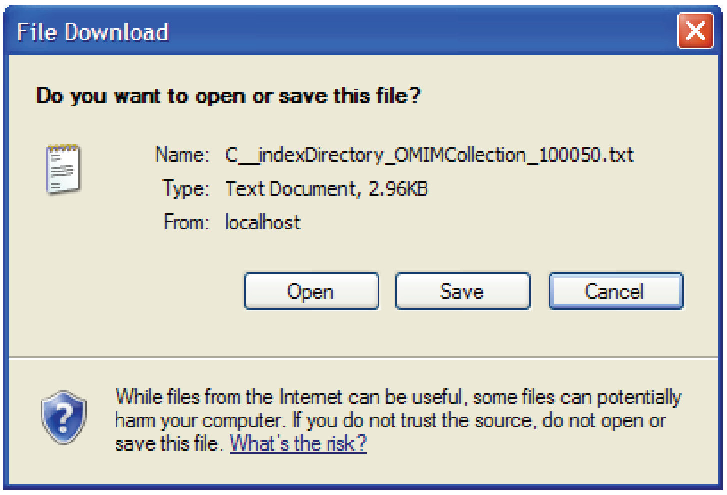
ID: 128800 (score: 0.935443703740029)
Click here to view...

ID: 119550 (score: 0.8754346641004048)
Click here to view...

ID: 100100 (score: 0.8464026490990118)
Click here to view...

Figure 5.8: Search result

To view one of the documents in the result set, the user must click on the "*Click here to view...*" URL. The user is then presented a download window, see figure 5.9 - the user may select "*Open*", "*Save*", or "*Cancel*".


File Download

Do you want to open or save this file?

Name: C__indexDirectory_OMIMCollection_100050.txt
Type: Text Document, 2.96KB
From: localhost

While files from the Internet can be useful, some files can potentially harm your computer. If you do not trust the source, do not open or save this file. What's the risk?

Figure 5.9: View document content

The user interface is kept on a simple level and it is very intuitive to use, cf. Jakob Nielsen's guidelines for web development [38].

6 Evaluation of results

The developed search engine application provides indexing and searching operations for biological information stored in the OMIM collection. The implemented application facilitates the retrieval of information about genetic disorders. This chapter deals with the evaluation of the system's capacity, limitations, etc., by performing various tests. The goal of this evaluation is to uncover defects and weaknesses of the implemented methods, and that the system behaves as intended by its designer.

6.1 System validation

A software testing stategy may be viewed as the spiral illustrated in figure 6.1 provided by Roger Pressman [42]. First, the role of software is defined by the system engineering, which leads to software requirements analysis. In the software requirements analysis the information domain, behavior, performance, constraints, and validation criteria for software are established. In order to develop computer software, one move inward along the spiral towards coding and decreases the level of abstraction on each turn.

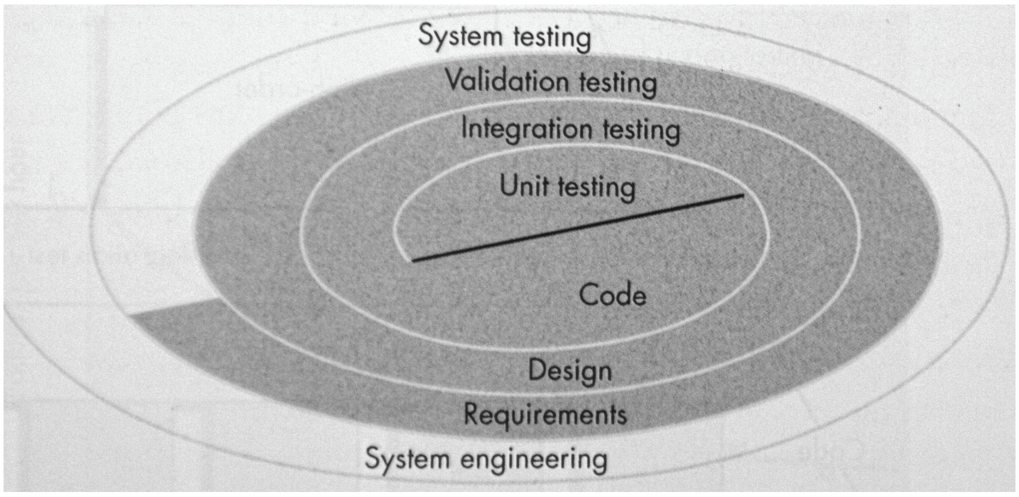


Figure 6.1: Software testing strategy (picture taken from [42])

Just like the strategy for software development a strategy for software testing may also be viewed as in the context of the spiral, however it begins at the center and move outward. *Unit testing* is the initial step and concentrates on each unit (e.g. component) of the software as implemented in source code.

As the testing progresses outward along the spiral, *integration testing* is the next testing phase. Integration testing focuses on desing and the construction of the software architecture. The third turn outward on the spiral lets one encounter the *validation testing*. In validation testing the software that has been constructed is validated against the requirements established as part of software requirements analysis.

The last turn on the spiral one arrive at *system testing*. System testing tests the software and other system elements as a whole. In contrary to the development of a system, the testing move along the streamline that broaden the scope of testing for each turn. Each turn in the testing phase will now be described a little more in detail.

Unit testing focuses on each component individually, hence the name, ensuring that it functions properly as a unit. It tries to verify the smallest units of the software design - a unit is a software component or a module. Within the boundary of the module, important control paths are tested to uncover errors on the component-level design description.

Integration testing is a turn consisting of two subgoals; constructing the program structure, and conduct tests to uncover errors associated with interfacing. The unit tested components is the building blocks of the program structure dictated by the design.

Validation testing begins when the software is completely assembled as a package, where interfacing errors have been uncovered and corrected. A simple definition of validation is that validation succeeds when software functions in a manner that can be reasonably expected by the customer [42]. Through a series of tests that demonstrates conformity with requirements, the software is validated.

The final step is *system testing*. Ultimately, software is incorporated with other system elements (e.g. hardware, people, information), and a series of system integration and validation tests are conducted [42]. These tests fall outside the scope of this thesis and are not described any further in detail.

This thesis will base its tests on a subset of possible test cases. As test data 537 OMIM records have been randomly but evenly selected - meaning that the selected records are distributed over all of the six different categories and is proportionately with the full OMIM collection. For an overview of the selected records, see Appendix C. If the full OMIM collection was to be indexed a matrix with the approximately 14 000 x 120 000 records has to be created. To keep a 14 000 x 120 000 matrix object requires a significant amount of memory, and because of this the test data contains only 537 records.

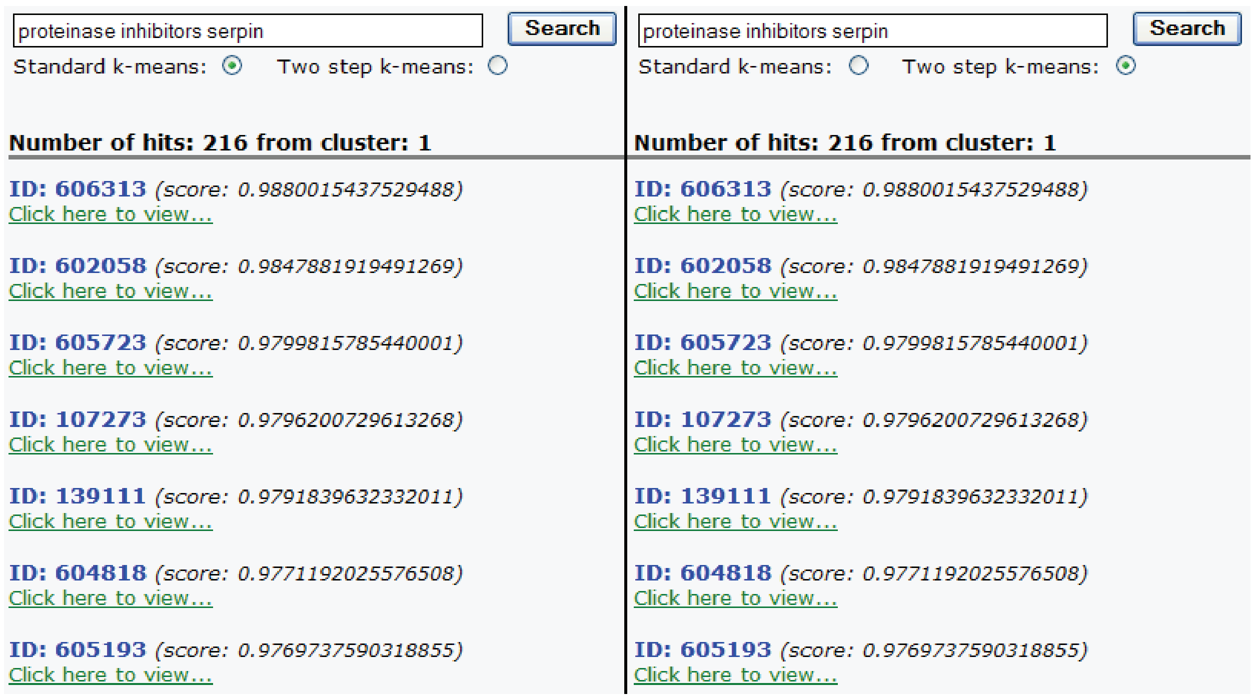
In this project *black-box testing* or *behavioral testing* will be applied, explained and executed in the following subchapter. The tests validates the system as a whole without considering the software's internal structure. The software's behavior is determined by studying its inputs and the related outputs [42].

6.2 Test cases

In this project there are implemented two clustering methods, both of them applied to the same LSI index producing two different cluster indices. Test cases are performed to understand the impacts by induce clusters in the index. In this chapter tests will be performed to validate the system as a whole - the units and the integration works properly and it is the totality of the system that are of interest. The results from each of these tests will be compared with results from NCBI's search engine in this chapter, and evaluated in chapter 6.4. All tests of the implemented application are executed with a marginal value of 0.80 for the similarity, this value must be exceeded in order for a document to be added to the result set. The following subchapters illustrates three cases. For an overview of all test queries see appendix E.

6.2.1 Case I

In this test case the user has typed in the query "*proteinase inhibitors serpin*", and executed using both of the clustering methods, see figure 6.2 for illustration - the user query is stemmed to "*proteinas inhibitor serpin*".



proteinase inhibitors serpin

[✓ Standard k-means:] Two step k-means:

Number of hits: 216 from cluster: 1

ID: 606313 *(score: 0.9880015437529488)*
Click here to view...

ID: 602058 *(score: 0.9847881919491269)*
Click here to view...

ID: 605723 *(score: 0.9799815785440001)*
Click here to view...

ID: 107273 *(score: 0.9796200729613268)*
Click here to view...

ID: 139111 *(score: 0.9791839632332011)*
Click here to view...

ID: 604818 *(score: 0.9771192025576508)*
Click here to view...

ID: 605193 *(score: 0.9769737590318855)*
Click here to view...

proteinase inhibitors serpin

Standard k-means: [✓ Two step k-means:]

Number of hits: 216 from cluster: 1

ID: 606313 *(score: 0.9880015437529488)*
Click here to view...

ID: 602058 *(score: 0.9847881919491269)*
Click here to view...

ID: 605723 *(score: 0.9799815785440001)*
Click here to view...

ID: 107273 *(score: 0.9796200729613268)*
Click here to view...

ID: 139111 *(score: 0.9791839632332011)*
Click here to view...

ID: 604818 *(score: 0.9771192025576508)*
Click here to view...

ID: 605193 *(score: 0.9769737590318855)*
Click here to view...

Figure 6.2: Standard k-means vs. Two step k-means (proteinase inhibitors serpin)

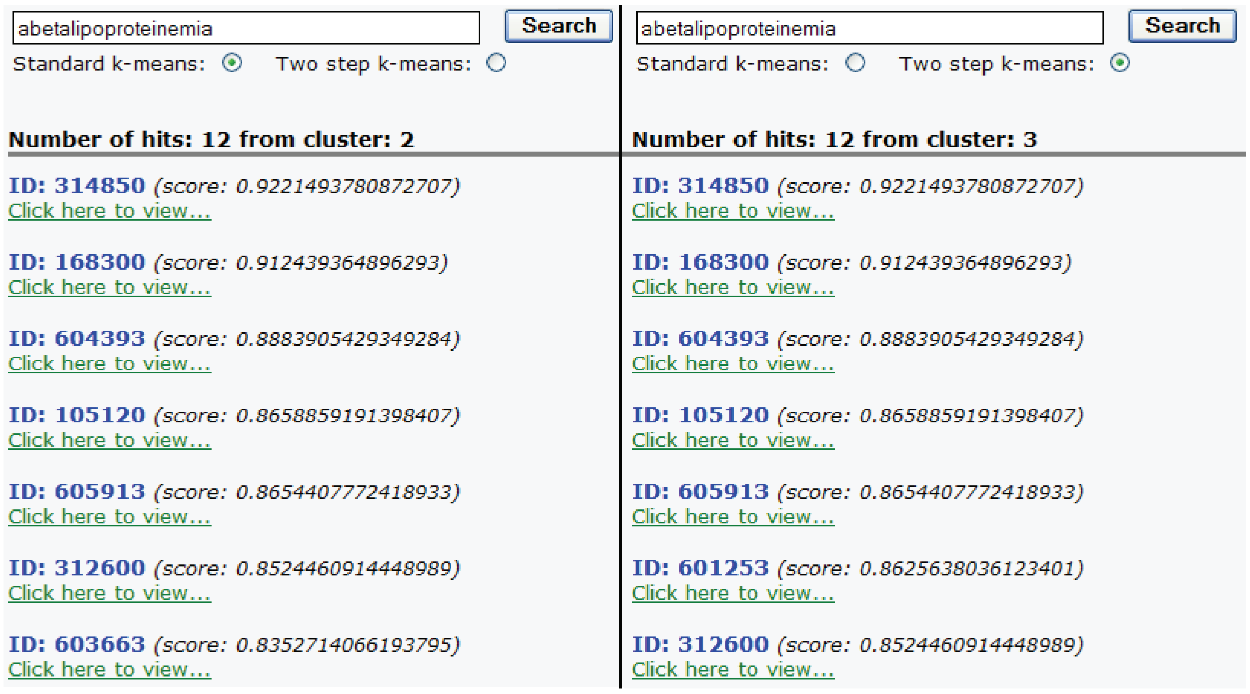
The application returns the same cluster number and the exact same result set using the two clustering methods, meaning that in this case the result sets coincides. One reason why the two clustering methods yields the exact same result set is that the centroid vectors of the two clusters may be very similar - the user query produce the same pseudo-document vector in both cases. Another reason might be that the distribution of the documents is approximately the same in both cases.

As mentioned in chapter 2 LSI matches concepts and not utilizing exact matching. Concept matching may result in occurrences of records that do not contain terms from the user query are conceptually more similar than records that actually contain terms from the user query. In this test the record ranked on top (606313) is, according to LSI, more conceptually similar than the rest of records in the result set. The record 606313 do not contain any of the terms from the user query in contrast to the record 602058 that contains all of the terms. The rest of the records that are shown in figure 6.2 contains variations of the words "*inhibitor*" and "*protein*". Proteinase and protein belong together conceptually since proteinase begins the hydrolytic breakdown of proteins usually by splitting them into polypeptide chains [7].

When executing a search with the same user query on NCBI's search engine, it produces a result where the record 602058 is the only record that exist in the test data for this project. NCBI's search engine utilize, as mentioned earlier, a boolean model or exact matching.

6.2.2 Case II

In test case I the two clustering methods yields the exact same result set and the same cluster number. In test case II the user has typed in the query "*abetalipoproteinemia*". The results from the two clustering methods differs in cluster number and some records have been replaced, see figure 6.3. The term is at its word root and is therefore not stemmed to a shorter word.


abetalipoproteinemia Search

Standard k-means: [✓] Two step k-means: [ ]

Number of hits: 12 from cluster: 2

ID: 314850 *(score: 0.9221493780872707)*
Click here to view...

ID: 168300 *(score: 0.912439364896293)*
Click here to view...

ID: 604393 *(score: 0.8883905429349284)*
Click here to view...

ID: 105120 *(score: 0.8658859191398407)*
Click here to view...

ID: 605913 *(score: 0.8654407772418933)*
Click here to view...

ID: 312600 *(score: 0.8524460914448989)*
Click here to view...

ID: 603663 *(score: 0.8352714066193795)*
Click here to view...

abetalipoproteinemia Search

Standard k-means: [ ] Two step k-means: [✓]

Number of hits: 12 from cluster: 3

ID: 314850 *(score: 0.9221493780872707)*
Click here to view...

ID: 168300 *(score: 0.912439364896293)*
Click here to view...

ID: 604393 *(score: 0.8883905429349284)*
Click here to view...

ID: 105120 *(score: 0.8658859191398407)*
Click here to view...

ID: 605913 *(score: 0.8654407772418933)*
Click here to view...

ID: 601253 *(score: 0.8625638036123401)*
Click here to view...

ID: 312600 *(score: 0.8524460914448989)*
Click here to view...

Figure 6.3: Standard k-means vs. Two step k-means (abetalipoproteinemia)

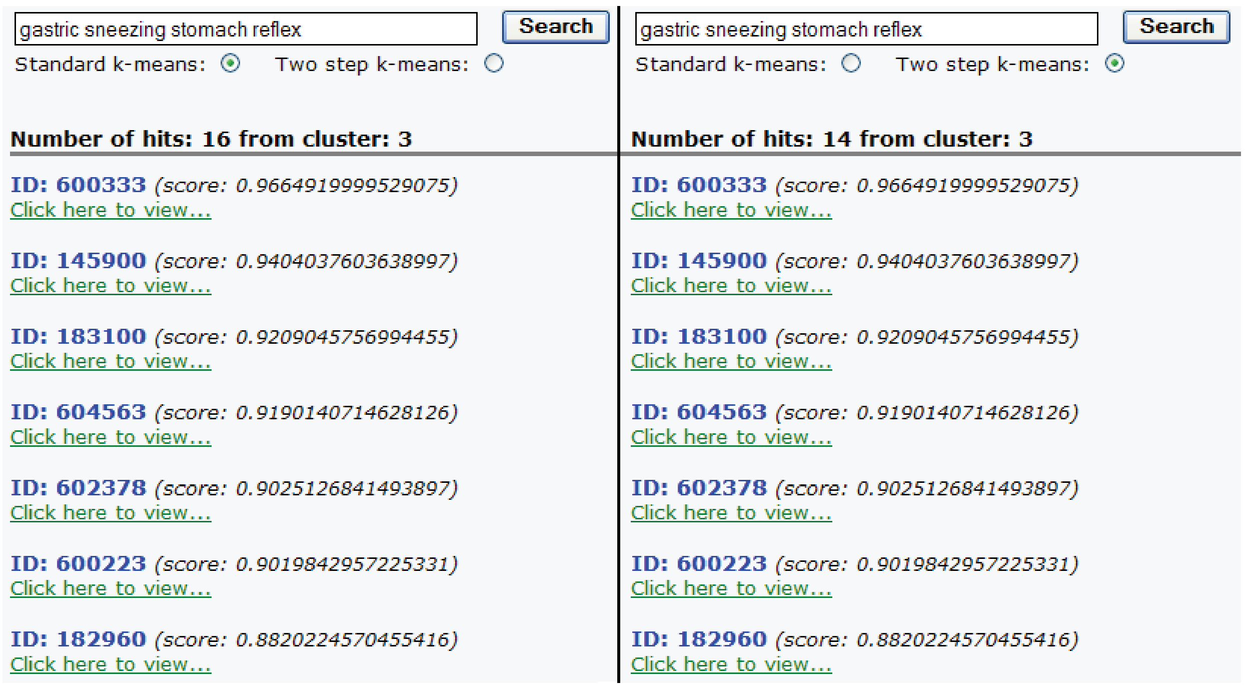
The biggest difference in this case is the change of cluster number. During the clustering process the Two step clustering method indicates some considerable changes in the document distribution in proportion to the standard k-means, mainly in cluster number 2 and 3, see appendix D - cluster number 1 is almost unaffected. The majority of documents in cluster 3 in the Two step method are documents from cluster 2 and some from cluster 1 in the standard k-means. The distribution is more uniform in the Two step method than in the standard k-means method.

Because of the difference in document distribution the Two step method retrieve a new record (601253) and omit another (300500, not shown in figure 6.3). The new record possess a higher similarity than the omitted one, meaning it is conceptually more similar to the user query.

When executing the same search in the NCBI's search engine the only record that contain the user query and is a part of the test data is the record 314850.

6.2.3 Case III

In the third, and final, test case the user has typed in the query "*gastric sneezing stomach reflex*", which is stemmed to "*gastric sneez stomach reflex*", see figure 6.4 for illustration of results.



gastric sneezing stomach reflex Search
[✓ Standard k-means:] Two step k-means:

Number of hits: 16 from cluster: 3

ID: 600333 *(score: 0.9664919999529075)*
Click here to view...

ID: 145900 *(score: 0.9404037603638997)*
Click here to view...

ID: 183100 *(score: 0.9209045756994455)*
Click here to view...

ID: 604563 *(score: 0.9190140714628126)*
Click here to view...

ID: 602378 *(score: 0.9025126841493897)*
Click here to view...

ID: 600223 *(score: 0.9019842957225331)*
Click here to view...

ID: 182960 *(score: 0.8820224570455416)*
Click here to view...

gastric sneezing stomach reflex Search
Standard k-means: [✓ Two step k-means:]

Number of hits: 14 from cluster: 3

ID: 600333 *(score: 0.9664919999529075)*
Click here to view...

ID: 145900 *(score: 0.9404037603638997)*
Click here to view...

ID: 183100 *(score: 0.9209045756994455)*
Click here to view...

ID: 604563 *(score: 0.9190140714628126)*
Click here to view...

ID: 602378 *(score: 0.9025126841493897)*
Click here to view...

ID: 600223 *(score: 0.9019842957225331)*
Click here to view...

ID: 182960 *(score: 0.8820224570455416)*
Click here to view...

Figure 6.4: Standard k-means vs. Two step k-means (gastric sneezing stomach reflex)

The standard k-means method retrieved two documents more than the Two step method, although both of the methods utilizes cluster 3. The two documents (192400 and 602195) have respectively been transferred to cluster 1 and 2 during the Two step clustering process. It is only the record 137130 that contain the terms in the user query, and terms from that record is used as foundation. The application utilizes the wrong cluster number, the record 137130 is located, in both of the clustering methods, in cluster 2, and can therefore not retrieve the correct document.

The documents with the highest conceptually similarity contains variations of the word reflex. One reason why the application retrieves the wrong cluster and therefore can not retrieve the desired record, may be the use of substring matching during search - e.g. the term *reflex* retrieve the index positions for index terms like; *areflexia*, *hyperreflexia*, *hyporeflexia*, and *reflex*. When performing the query conversion of the query vector to the document-concept space the three first index term positions may affect the retrieval by introducing noise to the search space - meaning retrieving index positions for index terms that the user in reality do not want to retrieve.

Another reason why the application utilizes the wrong cluster may be because the record 137130 is located in the border area of cluster 2 and 3 - meaning the document vector for the record 137130 is similar to the query vector, but the centroid vector for the cluster is less similar to the query vector than the centroid vector of cluster 3 and is therefore not selected by the application.

The NCBI's search engine retrieves the record 137130, which is the only record in the test data collection containing the query terms, when executing the same search.

6.4 Evaluation of test results

Each of the test cases produce different result, this chapter will evaluate these results and take a closer look at speed and latency, precision, capacity, and extensibility of the application.

6.4.1 Speed and latency

Indexing process

The Lucene indexing module utilize the compound file index structure. Multifile index creates multiple files to store separate segments of the index, in contrast to compound index that consists of three files; *deletetable*, *segments*, and a file containing the indexed documents and their field values. The multifile index structure is indexing the documents in less time than the compound. Reason for this is that the compound index structure merges all of the files into one single file.

To make use of the Lucene index in different stages the index is stored in a file system. The RAMDirectory provides faster indexing process than the FSDirectory, because RAMDirectory keep the index in the memory while the FSDirectory store it on disk. The reason why the FSDirectory is chosen instead of the RAMDirectory is to provide possibilities to do the indexing process in several independent steps. The trade-off is that it decreases the indexing speed significantly, and the Lucene indexing process takes about 0,5-1 hour[10] to index 537 documents.

Clustering process

The clustering processes differs a little in runtime. The standard k-means clustering method runs for about 0,5-1,5 minutes depending on the initial centroid vectors, while the Two step k-means clustering method runs for about 1-2,5 minutes. The reason why the Two step clustering method increases the clustering time is because the method runs in a local and a global mode.

Searching process

The speed of this application is not significantly high because the indices and the main searching operation are not optimized for speed, the indices consist of regular text to make debugging easier. The runtime of a search performed on the test data collection is approximately 5 seconds. This thesis does not emphasize the importance of speed during information retrieval, mainly because this is not the main scope of this project. An average retrieval runtime of 5 seconds may, however, be concluded as satisfiable in this projects.

6.4.2 Precision

Since the LSI method matches concepts instead of exact matching the precision of the application may be difficult to verify, because it requires good knowledge of the molecular biology field, especially when dealing with biology specific terms. This chapter will evaluate the precision of the application.

10.All of the runtimes are executed and tested on a 2 GHz computer with 512 MB RAM.

When performing a search the precision fluctuates, this may have its origin in the distribution of the documents and by the following factors; (1) treating a document as a whole during the indexing and clustering processes without any consideration of the various fields (TI, TX, CD, etc.), (2) the rank k value is too high or low and therefore fails to remove the neccessary noise, (3) the clustering methods creates constant boundaries between the clusters and therefore split up, for instance two documents that are conceptually similar into two different clusters, or (4) the use of substring matches during search generates a query vector with noise.

An another way of treating the documents, than by treating the document as a whole without any consideration of the various fields (1), is to make use of the more important fields, such as NO, TI, TX, AV and CS, and omit less important fields, such as ED, RF, SA, CN and CD. Fields such as ED exists in each record and consist generally of the same contributors and is a major associative pattern in the data, and may therefore be considered redundant. This may give a poor retrieval precision due to the less important fields, but the meaning is to give them a more superior foundation when comparing the documents with each other and with the queries.

The value of the rank k (2) influences the latent structure and the removal of noise significantly. The original term frequency matrix is 537 (number of documents) x 23 897 (number of index terms), and the rank k is set to 10. The value of k is estimated based on the the number of documents and index terms. For hundred thousands of documents a rank k value of 100 is said to give the best performance [5] [10] [13], so in this project a rank k value of 10 appears to be a rational value. However the value of k could be further experimented.

A third possible factor for the flactuating precision may be that the clusters are constant (3) and, e.g. a document that is similar to two different cluster is added to the second cluster, then the document will not be retrieved if the application utilizes the first cluster, c.f. test case III. This is a drawback of the cluster methods and should optimally be fixed so the documents are retrievable independent of the cluster distribution.

With substring matches (4) the query may obtain noise when creating the query vector. The application creates a query vector with maybe unwanted index term positions, c.f. test case III, and therefore the precision of a search decreases. There are various ways of dealing with term matching when creating a query vector, e.g. calculate the similarity between the index terms and the query terms to retrieve index term positions. Examples of semantic similarity measures can be found on the Cog Works website [8]. The substring matching is easier to implement and were therefore chosen in this project.

6.4.3 Capacity and extensibility

With a 537 x 23 897 matrix there are capacity limitations in memory. Each cell in a JAMA Matrix object keep a 64 bit float value, so to keep the entire term frequency matrix in RAM requires a significant great amount of memory - the Java Virtual Machine[11] also seems to struggle to handle memory requirements above 2 GB of RAM due to the heap size. Since the JAMA package does not provide storage to disk operations, this application was tested with a test data collection of 537 records - the number of documents (537) was a random selection, optimally it should be exactly

11.http://java.sun.com/

500 documents, but with 537 the ratio between the different inheritance types is kept on a tolerable level.

The application is to some extent extensible but it requires work to overcome minor obstacles such as hard-coded file paths and etc.. The hard-coded file paths are mainly associated with the writing and reading procedures for the indexing, clustering and searching methods. A possible extensibility is to implement the application so it does not treat the document as a whole, but instead select the important fields (TI, TX, etc.) as foundation to cluster the documents. By omitting the less important fields, the document distribution may and probably will be different than the distribution this thesis is based on.

6.5 Evaluation of acceptance test results

In the acceptance test 15 various user queries were executed, see appendix E for the complete list of test queries. This testing reveals any defects and limitations of the system, it will also determine whether the system is ready to be used or not by biologists.

As mentioned in the beginning of this thesis, the biology field produce a significant amount of information and the need to store and retrieve it in a reasonable matter is of great interest. The proposed methods do have their limitations, mainly due to the clustering methods.

The LSI method works per se satisfactory since the documents one search for turns up high on the result lists, when retrieved. If the documents were not clustered, the results would most probably be different and with an improved degree of retrieval although decreased speed. Limitations in this system is mainly concerned around the clustering methods. When utilizing the two clustering methods implemented in this project, two conceptual similar documents may end up in two different clusters. With that only one of the documents may be retrieved when performing a search. If the application would perform a search without utilizing the clustering methods, it would most probability retrieve both of the documents and return it to the user, however the speed would decrease significantly. One example of this scenario is the test query "*syndrome*" that returns 19 conceptually similar documents when performing search on the standard k-means clusters, while with Two step k-means clusters 12 documents are returned. This implies that 7 conceptually similar documents are not a part of the result set returned by the Two step k-means clustering search, because they are located in an another cluster. This is a deficiency by the clustering methods and may produce unwanted results.

In the test where the application is supposed to retrieve documents with the term "*gastric*", two different results are returned based on the chosen clustering method; standard k-means retrieves only the record 602613 from cluster 3, while Two step k-means do not retrieve any records that matches the query term. The record 602613 contains the term "*gastrointestinal*" which is fairly conceptually similar to the query - the term "*gastric*" is at its word root and the document is therefore not retrieved by exact matching. Optimally the record 137130 would be retrieved but it is not located in cluster 3. In both cases the record 137130 is located in cluster 2, and in Two step k-means the record 602613 is also located in cluster 2. The fact that one could not retrieve the record 602613, in this case, indicates that the application utilized the wrong cluster and that it is not an error in calculation when calculating the similarity.

For each cluster the Two step k-means should optimally yield a more fairly distribution of the documents and discover a more optimal centroid vector, when based on the same initial basis, than the standard k-means do.

Which one of the two clustering methods performs best depends on which trade-offs one emphasizes, whether speed is more important than precision, and etc.. Although the system has its limitations the runtime of a search, performed on an clustered LSI index, is reduced to approximately a third of the time compared with a non-clustered index. This reduction in runtime comes from the reduction in search space - i.e. the number of dimensions is reduced considerably. The clusters created by the Two step method does also have a more uniform distribution than the clusters created by the standard k-means. This arise from the fact that the centroid vectors are forced towards the extreme points. As Faraoun and Boukelif [15] mentions in their article, to utilize the k-means method the best way is to place the initial centroid vectors as far away from each other as possible. In light of this the Two step k-means method may be viewed as an semi-automatic method for forcing the centroid vectors as far as possible away from each other. The goal of being less greedy is not met, since the method is still greedy when it switches to global mode.

There are no extensive use of error handling and may yield unsatisfactory results. For instance if one cluster do not contain any documents the cluster have to retain the old centroid vector in order not to be empty when the clustering process is finished, etc.. By doing it that way the method will most probably be more automatic and safe from any possible errors.

By utilizing conceptual search one may retrieve documents that does not contain the search queries. This has its advantages when queries like "*proteinase inhibitors serpin*" retrieve several documents containing the word protein, although "*proteinase*" is stemmed to "*proteinas*". In exact matching on one would not be able to retrieve documents containing the word protein when the query is stemmed to proteinas, the reason why these documents are retrieved in this application is because proteinase and protein is conceptual similar according to the structure created by the SVD reduction. Searches done on the Two step clusters in some cases the result sets are more concentrated than the standard k-means, while in other cases the results are not satisfactory.

Based on the test results it may be concluded that despite the drawbacks and the limitations, the system may be ready to be used, although it is recommended that the system should be polished before eventually launched.

7 Conclusion

As mentioned in the beginning of this thesis the NCBI's OMIM search engine is based on a Boolean retrieval model. Clusters based on an index created by the LSI model would have different result sets than on NCBI's search engine. This arise from the fact that search engines based on LSI retrieve documents that are conceptually similar to the query, in contrast to boolean models that performs exact matching. The LSI model manage problems with synonymy and to some extent polysemy. As shown in the test cases documents that do not contain the search queries are also retrieved.

The tests have shown that in some cases the Two step method performs better than the standard k-means, and vice versa. The main advantage of the Two step method is that it force the centroid vector towards the extremities and in that way the method is provided a completely different starting point than the case is for the standard k-means method. Although the method is provided with a completely different starting point, the location of the initial centroid vectors affects, however not in the same extent as the case in standard k-means, the final result. An another disadvantage is that if the cluster does not receive any documents, it will remain empty when the clustering process is finished. This is a deficiency by the system at this moment, if such situation should arise, however it is not a demanding task to correct it, so if a cluster does not receive any documents the cluster retain the old centroid vector.

The limitations that has been done regarding the tests affects their execution of them, for instance, to limit the test data to 537 documents, the marginal value was set to 0.80 in local mode in order to avoid empty clusters. This value determines how far the centroids are pushed away from each other.

Although the Two step method modifies the location of the initial centroid vectors in local mode, it is still sensitive to the location of them. By retaining the old centroid vector if no documents are added to the cluster, it will be less sensitive to it.

Regardless of the system deficiencies, it can be used to retrieve relevant records from the set of test data. It performs well in most of the cases, but also fails miserably some times. The main goal of developing this search engine was to find an optimal way to index and ease the retrieval of the information later, it may be concluded that the main objective of this project was nearly achieved.

7.1 Future work

In this part suggestions of further work and what features of information retrieval application can be improved are explained. First, the implemented search engine only deals with documents which are stored as plain text. The textual data are produced by extracting terms from the record information. If it is desirable to manage and search documents of other file types, PDF, HTML, MS Word, etc., the application requires a specific document parser for the individual types.

Secondly the structure of the individual documents may be extended to be able to index other types than OMIM records, for instance MEDLINE [36] abstracts. In order to achieve this the document parser have to be updated to be able to treat the data in the correct fashion.

The most interesting future work is probably not to treat each document as a whole, but rather just index the documents based on the more important fields as mentioned in chapter 6.4.2. This requires some modifications in the analyzer, but would most probably affect the retrieval precision noticeably.

Although the Two step method performs better than the standard k-means, and vice versa, in some cases the omission caused by the clustering methods are problematic. In worst case scenarios, such as the test with the query “*gastric*” where the Two step method could not retrieve any documents due to the document distribution, the application does not succeed in retrieving the correct documents. This occurs mainly when the documents the user desires lies near the border area of the clusters. An improvement on that area and the application would most probably improve the document retrieval and overcome the deficiencies caused by these clustering methods.

Finally, the retrieval speed ought to be improved along with the problems with the capacity when performing the SVD reduction. The index data in this project is stored as plain text and have to be preprocessed when performing a search, which results in an unnecessary latency. An improvement here would decrease the runtime significantly.

References

[1] Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K., Walter, P. 2002. *Molecular Biology of The Cell 4th edition*. Retrieved from http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?cmd=Search&db=books&doptcmdl=GenBookHL&term=%22all+cells%22+AND+mboc4%5Bbook%5D+AND+372023%5Buid%5D&rid=mboc4.section.4#23 (last visited xx.xx.xxxx).

[2] Altman, R. Bioinformatics in Support of Molecular Medicine. *AMIA*. 53-61.

[3] Baeza-Yates, R., Riberio-Neto, B. 1999. *Modern Information Retrieval*. Addison-Wesley. Boston, MA.

[4] Bayat, A. 2002. Bioinformatics. *BMJ*. 324:1018-1022.

[5] Berry, M. W., Dumais, S. T., O'Brien, G. W. 1995. Using Linear Algebra for Intelligent Information Retrieval. *SIAM*. 37(4):573-595.

[6] Brants, T. 2005. Test Data Likelihood for PLSA Models. *Information Retrieval*. 8(2):181-196.

[7] Campbell, N. A., Reece, J. B. 2005. *Biology 7th Edition*. Benjamin Cummings. San Francisco, CA.

[8] Cog Works. Measures of Semantic Relatedness (MSR). Retrieved from http://cwl-projects.cogsci.rpi.edu/msr/ (last visited xx.xx.xxxx).

[9] Dasgupta, A., Kumar, R., Raghavan, P., Tomkins, A. 2005. Variable Latent Semantic Indexing. *ACM SIGKDD*.

[10] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., Harshman, R. 1990. Indexing by Latent Semantic Analysis. *JASIS*. 41(6):391-407.

[11] Dictionary.com. Retrieved from http://dictionary.reference.com/ (last visited xx.xx.xxxx).

[12] Ding, C. H. Q. 2005. A Probabilistic Model for Latent Semantic Indexing. *JASIS*. 56(6):597-608.

[13] Efron, M. 2005. Eigenvalue-Based Model Selection During Latent Semantic Indexing. *JASIST*. 56(9):969-988.

[14] European Bioinformatics Institute. Retrieved from http://www.ebi.ac.uk/2can/bioinformatics/ (last visited xx.xx.xxxx).

[15] Faraoun, K. M., Boukelif, A. 2006. Neural Networks Learning Improvement using the K-Means Clustering Algorithm to Detect Neural Intrusions. *IJCI*. 3(2):161-168.

[16] Fraley, C., Raftery, A. E. 1998. How Many Clusters? Which Clustering Method? Answers Via Model-Based Cluster Analysis. *Computer Journal.* 41(8):578-588.

[17] Gleich, D., Zhukov, L. 2004. SVD Subspace Projections for Term Suggestion Ranking and Clustering. *ACM SIGIR'04*.

[18] Golub, G. H., Kahan, W. 1965. Calculating the Singular Values and Pseudoinverse of a Matrix. *SIAM Journal of Numerical Analysis*. 2:205-224.

[19] Gospodnetic, O., Hatcher, E. 2005. *Lucene In Action*. Manning Publications. Greenwich, CT.

[20] Hagen, J. B. 2000. The Origin of Bioinformatics. *Nature Reviews: Genetics*. 1:231-236.

[21] Hamosh, A., Scott, A. F., Amberger, J. S., Bocchini, C. A., McKusick, V. A. 2005. Online Mendelian Inheritance in Man (OMIM), a Knowledgebase of Human Genes and Genetic Disorders. *Nucleic Acids Research*. 33:514-517.

[22] Hendrickson, B.2006. Latent Semantic Analysis and Fiedler Embeddings. *Proceedings of SIAM Workshop on Text Mining*. April 2006.

[23] Hofmann, T. 1999. Probabilistic Latent Semantic Indexing. *ACM SIGIR*. 50-57.

[24] JAMA. Java Matrix Package. Retrieved from http://math.nist.gov/javanumerics/jama/ (last visited xx.xx.xxxx).

[25] Jing, L., Ng, M. K., Yang, X., Huang, J. Z. 2006. A Text Clustering System based on k-means Type Subspace Clustering and Ontology. *International Journal of Intelligent Technology.* 1(2):91-103.

[26] Karciauskas, G. 2005. Learning with Hidden Variables: A Parameter Reusing Approach for Tree-Structured Bayesian Networks. Master thesis.

[27] Krishnan, V. 2005. Shortcomings of Latent Models in Supervised Settings. *ACM SIGIR'05*. 625-626.

[28] Lavrenko, V. P. Generative Density Allocation. Dissertation chapter. Retrieved from http://ciir.cs.umass.edu/~lavrenko/ (last visited xx.xx.xxxx).

[29] Lesk, A., M. 2002. *Introduction to Bioinformatics*. Oxford University Press. Oxford, UK.

[30] Lu, G. 1999. *Multimedia Database Management Systems*. Artech House. Norwood, MA.

[31] Lucene. Retrieved from http://lucene.apache.org/ (last visited xx.xx.xxxx).

[32] Luscombe, N. M., Greenbaum, D., Gerstein, M. 2001. What is Bioinformatics? A Proposed Definition and Overview of the Field. *Methods of Information in Medicine*. 40(4):346-358.

[33] MacQueen, J. B. 1967. Some Methods for Classification and Analysis of Multivariate Observations. *Proceedings of 5th Berkley Symposium on Mathematical Statistical and Probability*. Berkley, University of California Press. 1:281-297.

[34] Matteucci, M. A Tutorial on Clustering Algorithms. Retrieved from http://www.elet.polimi.it/upload/matteucc/Clustering/tutorial_html/kmeans.html (last visited xx.xx.xxxx)

[35] McCutcheon, A. L. 1987. Latent Class Analysis. *Quantitative Applications in the Social Sciences Series No. 64*. Sage Publications. Thousand Oaks, CA.

[36] MEDLINE. Retrieved from http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?DB=pubmed (last visited xx.xx.xxxx).

[37] National Center for Biotechnology Information. Retrieved from http://www.ncbi.nlm.nih.gov/Entrez/ (last visited xx.xx.xxxx).

[38] Nielsen, J. 2000. *Designing Web Usability: The Practice of Simplicity*. New Riders Publishing. Thousand Oaks, CA.

[39] Online Biology Book. Retrieved from http://www.emc.maricopa.edu/faculty/farabee/BIOBK/BioBookTOC.html (last visited xx.xx.xxxx).

[40] Online Mendelian Inheritance in Man. Retrieved from http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=OMIM (last visited xx.xx.xxxx).

[41] Porter, M. The Porter Stemming Algorithm. Retrieved from http://www.tartarus.org/~martin/PorterStemmer/index.html (last visited xx.xx.xxxx).

[42] Pressman, R. S. 2001. *Software Engineering: A Practitioner's Approach 5th Edition*. McGraw-Hill. New York, NY.

[43] Robertson, S. E., Walker, S., Beaulieu, M. 1998. Okapi at TREC-7: Automatic Ad Hoc, Filtering, VLC and Interactive Track. *Proceedings of the Seventh Text REtrieval Conference*. Gaithensburg, MD.

[44] Wheeler, D. L., et al.. 2006. Database Resources of the National Center for Biotechnology Information. *Nucleic Acids Research*. 35:5-12.

[45] Zhukov, L. 2004. Technical Report Spectral Clustering of Large Advertiser Datasets Part I. *Overture R&D*.

Appendices

Appendix A: List of Stopwords

A	are	cd	et	her	itself
a	arise	cm	etc	here	
about	around	cn	ever	hereafter	**J**
above	as	come	every	hereby	just
abs	assume	compare	everyone	herein	
accordingly	at	could	everything	hereupon	**K**
across	av	cs	everywhere	hers	keep
after			except	herself	kept
afterwards	**B**	**D**		him	kg
again	be	de	**F**	himself	km
against	became	dealing	field	his	
all	because	department	find	how	**L**
almost	become	depend	for	however	last
alone	becomes	did	found	hr	latter
along	becoming	discover	from		latterly
already	been	dl	further	**I**	lb
also	before	do		i	ld
although	beforehand	does	**G**	ie	letter
always	being	done	gave	if	like
am	below	due	get	ii	ltd
among	beside	during	give	iii	
amongst	besides		go	immediately	**M**
an	between	**E**	gone	importance	made
analyze	beyond	each	got	important	mainly
and	both	ec	gov	in	make
another	but	ed		inc	many
any	by	effected	**H**	incl	may
anyhow		eg	had	indeed	me
anyone	**C**	either	has	into	meanwhile
anything	came	else	have	investigate	mg
anywhere	can	elsewhere	having	is	might
applicable	cannot	enough	he	it	ml
apply	cc	especially	hence	its	mm

mo
more
moreover
mostly
most
mr
much
mug
must
my
myself

N
namely
nearly
necessarily
neither
never
nevertheless
next
no
nobody
noone
nor
normally
nos
not
noted
nothing
now
nowhere

O
obtained
of
off
often
on
only
onto
or
other
others
otherwise
ought
our
ours
ourselves
out
over
overall
owing
own
oz

P
particularly
per
perhaps
pm
precede
predominantly
present
presently
previously
primarily
promptly
pt

Q
quickly
quite

R
rather
readily
really
recently
refs
regarding
relate
rf

S
sa
said
same
seem
seemed
seeming
seems
seen
seriously
several
sh0wn
shall
she
should
show
showed
shown
shows
significantly
since
slightly
so
some
somehow
someone
something
sometime
sometimes
somewhat
somewhere
soon
specifically
still
strongly
studied
sub
substantially
such
sufficiently

T
take
tell
th
than
that
the
their
theirs
them
themselves
then
thence
there
thereafter
thereby
therefore
therein
thereupon
these
they
this
thorough
those
though
through
throughout
thru
thus
ti
to
together
too
toward
towards
try
tx
type

U
ug
under
unless
until
up
upon
us
use
used
usefully
usefulness
using
usually

V
various
very
via

W
was
we
were
what
whatever
when
whence
whenever
where
whereafter
whereas
whereby
wherein
whereupon
wherever
whether
which
while
whither
who
whoever

whom
whose
why
will
with
within
without
wk
would
wt

Y
yet
you
your
yours
yourself
yourselves
yr

Appendix B: JAMA: Java Matrix Package

JAMA is a basic linear algebra package for Java, it provides user-level classes for constructing and manipulating real, dense matrices. It is meant to provide sufficient functionality for routine problems, packaged in a way that is natural and understandable to non-experts [24].

The JAMA package comprise six different Java classes: Matrix, CholeskyDecomposition, LUDecomposition, QRDecomposition, SingularValueDecomposition and EigenvalueDecomposition.

Fundamental operations of numerical linear algebra are provided by the Matrix class. In order to access the submatrices and matrix elements a set of various *gets* and *sets* are provided. Basic arithmetic operations are provided; matrix addition and multiplication, matrix norms and selected element-by-element array operations.

The five fundamental matrix decompositions consists of pairs of triples of matrices, permutation vectors, and the like, produce results in five decomposition classes. They also compute solutions of simultaneous linear equations, determinants, inverses and other matrix functions:

- Cholesky Decomposition of symmetric, positive definite matrices
- LU Decomposition (Gaussian elimination) of rectangular matrices
- QR Decomposition of rectangular matrices
- Eigenvalue Decomposition of both symmetric and nonsymmetric square matrices
- Singular Value Decomposition of rectangular matrices

The current version of the JAMA package deals only with real matrices and represents a compromise between the need for pure and elegant object-oriented design and the need to enable high performance implementations.

A summary of JAMA's capabilities are presented in table C1.

Object manipulation	constructors
	set elements
	get elements
	copy
	clone
Elementary operations	addition
	subtraction
	multiplication
	scalar multiplication
	element-wise multiplication
	element-wise division
	unary minus
	transpose
	norm
Decompositions	Cholesky
	LU
	QR
	SVD
	symmetric eigenvalue
	nonsymmetric eigenvalue
Equation solution	nonsingular systems
	least squares
Derived quantities	condition number
	determinant
	rank
	inverse
	pseudoinverse

Table C1: JAMA capabilities [24]

The JAMA package is planned to become the primary linear algebra package for Java.

Appendix C: List of tested OMIM records

OMIM ID:

100050	118920	139111	158170	182099	212540
100100	118945	139250	158310	182128	215550
100500	119550	139400	159950	182135	215850
100640	120050	139630	160200	182269	217600
100670	120180	142310	160990	182410	219000
100678	122100	142600	161000	182451	219095
101200	122780	142625	162091	182960	221600
102800	122920	142810	162662	183100	221750
102940	123670	142959	162830	184253	221995
103400	123841	142964	163980	184700	223340
104160	123856	145250	164180	184754	223400
104210	123880	145900	164220	185510	225060
104701	124070	146350	164760	186860	229000
105120	125635	146530	165020	186921	229050
107266	125650	146661	165280	187330	229800
107273	126180	146691	165340	189889	230400
107480	126390	146810	166300	189904	232400
107601	126410	147080	167220	189971	233650
107950	126950	147130	167405	190220	233800
108720	128800	147330	168300	190310	235600
108730	128992	147390	169720	190950	236700
108745	131230	147563	170950	191000	236800
109530	131600	147568	170998	191161	240500
109770	131850	147679	171750	191170	241540
110000	132800	148200	172250	191306	241800
111680	133020	148370	173120	192400	243080
111750	133250	150200	173325	193100	246470
112263	133535	150340	176270	193230	246555
112265	133600	150370	176888	194320	248190
113950	134520	151250	176930	194535	248310
114100	134830	152424	176944	194540	250950
114120	134934	152445	177075	194545	251000
114160	136140	153380	178500	202400	252350
114210	136590	153432	178630	203800	253230
114230	137130	153550	179600	206900	254000
115080	137150	154050	179800	207780	254130
116100	137164	154250	179837	208400	255500
116600	138150	155120	180231	208600	256100
116948	138297	156530	180630	209920	258150
117900	138322	156560	180660	210700	258360
118190	138680	157151	180740	211770	259600
118503	139100	157600	180990	211920	260530

260600	300580	600313	601298	602318	603443
263610	301790	600315	601363	602348	603448
264270	301815	600333	601418	602378	603463
264700	301830	600398	601420	602383	603465
264810	301845	600403	601423	602423	603493
265600	302900	600423	601443	602453	603558
267010	304350	600438	601473	602458	603563
267450	304700	600453	601478	602488	603593
268315	305660	600473	601508	602588	603658
271150	306995	600478	601583	602593	603663
271250	307200	600503	601588	602600	603693
272370	309555	600513	601608	602605	603758
272650	310300	600578	601623	602613	603763
274000	310350	600583	601653	602638	603793
274205	311810	600623	601658	602643	603823
276400	312600	600688	601663	602673	603828
277350	313850	600693	601668	602738	603863
277440	314850	600723	601698	602743	603873
277480	314998	600788	601725	602773	603908
300030	400023	600793	601728	602803	603913
300060	400028	600823	601808	602808	604013
300065	400033	600853	601838	602863	604078
300095	402500	600858	601843	602873	604163
300145	450000	600888	601883	602883	604183
300165	516040	600905	601913	602913	604248
300167	555000	600908	601918	602918	604393
300170	590015	600910	601948	602948	604413
300185	590020	600913	601998	603013	604463
300210	590105	600973	602000	603018	604563
300215	600013	601038	602023	603045	604588
300280	600023	601043	602053	603088	604643
300297	600025	601048	602058	603093	604668
300300	600053	601078	602073	603123	604698
300320	600083	601108	602083	603153	604718
300350	600113	601113	602098	603158	604818
300355	600118	601130	602163	603218	604883
300430	600148	601148	602168	603233	604958
300455	600173	601188	602173	603235	605023
300465	600175	601193	602190	603238	605043
300470	600190	601223	602195	603338	605163
300500	600218	601253	602213	603348	605168
300540	600223	601255	602218	603373	605193
300545	600288	601270	602283	603378	605273
300550	600308	601293	602313	603433	605308

605353
605418
605483
605583
605623
605633
605658
605723
605823
605868
605913
605928
606008
606073
606173
606238
606278
606288
606313
606393
606445

Appendix D: Document distributions

Standard k-means:

Cluster 1:

100640	138297	173120	300145	600503	601663
100670	138322	173325	300165	600578	601698
100678	138680	176888	300167	600583	601725
104160	139111	176930	300185	600623	601728
104210	139250	177075	300280	600688	601838
104701	142310	178630	300297	600693	601843
107266	142600	179837	300300	600723	601883
107273	142810	180231	300320	600788	601913
108730	142964	180630	300350	600793	601918
108745	146661	180660	300470	600823	601948
109530	146691	180740	300540	600853	601998
109770	146810	180990	300545	600858	602000
111680	147080	182099	305660	600888	602053
111750	147130	182128	314998	600910	602058
112263	147390	182135	400023	600913	602073
112265	147563	182269	400028	600973	602098
114160	147568	182451	400033	601038	602163
114210	147679	184700	402500	601043	602168
114230	150200	185510	450000	601048	602173
116948	150340	186860	516040	601078	602190
118190	150370	186921	590015	601108	602213
118503	151250	187330	590020	601113	602283
118920	152424	189889	590105	601130	602313
118945	152445	189904	600013	601148	602318
123670	153380	189971	600023	601188	602348
123841	153432	190220	600025	601193	602383
123856	154050	190950	600053	601253	602423
124070	154250	191161	600148	601255	602453
125650	155120	191170	600173	601270	602458
126390	156560	191306	600190	601293	602488
126410	162091	194535	600288	601363	602593
128992	162662	194540	600308	601418	602600
131230	163980	194545	600315	601423	602638
133250	164760	209920	600398	601443	602643
133535	165020	232400	600403	601473	602673
134830	165280	248310	600423	601508	602738
134934	165340	263610	600438	601583	602743
137150	167405	272370	600453	601588	602773
137164	170998	300060	600473	601623	602803
138150	172250	300065	600478	601658	602808

602863	604668
602873	604698
602883	604718
602913	604818
602918	604883
602948	604958
603013	605023
603018	605043
603045	605163
603088	605168
603093	605193
603123	605273
603153	605308
603158	605353
603233	605418
603235	605483
603238	605623
603338	605633
603348	605658
603373	605723
603378	605823
603433	605868
603443	605928
603448	606008
603463	606073
603465	606173
603493	606238
603558	606278
603593	606288
603658	606313
603693	606393
603758	
603763	
603823	
603863	
603873	
603908	
603913	
604013	
604078	
604163	
604248	
604463	
604588	
604643	

Cluster 2:

100050	147330	223340	300095	603663
100100	148200	223400	300170	603793
100500	148370	225060	300210	604183
101200	153550	229000	300215	604393
102800	158170	229050	300355	604413
102940	158310	229800	300430	605583
103400	160200	230400	300455	605913
105120	160990	233650	300465	606445
107480	161000	233800	300500	
107601	162830	235600	300550	
107950	164180	236700	300580	
110000	164220	236800	301815	
113950	166300	240500	301845	
114100	168300	241540	304350	
115080	169720	241800	304700	
116100	170950	243080	306995	
116600	171750	246470	307200	
117900	176270	246555	309555	
119550	176944	248190	310300	
120050	178500	250950	310350	
120180	179600	251000	311810	
122100	179800	252350	312600	
122780	182410	253230	313850	
122920	184754	254000	314850	
125635	190310	254130	555000	
126180	191000	256100	600083	
128800	193100	258150	600113	
131600	194320	258360	600118	
131850	202400	260530	600218	
132800	203800	260600	600313	
133020	206900	264700	600513	
133600	208400	264810	600905	
134520	208600	265600	600908	
136140	210700	267010	601298	
136590	211770	267450	601420	
137130	211920	268315	601478	
139100	212540	272650	601608	
139400	215850	274000	601653	
139630	217600	274205	601808	
142625	219000	276400	602023	
142959	219095	277350	602083	
145250	221600	277440	602218	
146350	221750	277480	602588	
146530	221995	300030	602605	

Cluster 3:

108720
114120
123880
126950
145900
156530
157151
157600
159950
167220
182960
183100
184253
192400
193230
207780
215550
255500
259600
264270
271150
271250
301790
301830
302900
600175
600223
600333
601223
601668
602195
602378
602613
603218
603563
603828
604563

Two step k-means:

Cluster 1:

100640	139111	177075	300280	600723	601998
100670	139250	178630	300297	600788	602000
100678	142310	179837	300300	600793	602053
104160	142600	180231	300320	600823	602058
104210	142810	180630	300350	600853	602073
104701	142964	180660	300470	600858	602098
107266	146661	180740	300500	600888	602163
107273	146691	180990	300540	600910	602168
108730	146810	182099	300545	600913	602173
108745	147080	182128	305660	600973	602190
109530	147130	182135	314998	601038	602195
109770	147390	182269	400023	601043	602213
111680	147563	182451	400028	601048	602218
111750	147568	185510	400033	601108	602283
112263	147679	186860	402500	601113	602313
112265	150200	186921	450000	601130	602318
114160	150340	187330	516040	601148	602348
114210	150370	189889	590015	601188	602383
114230	151250	189904	590020	601193	602423
116948	152424	189971	600013	601255	602453
118190	152445	190220	600023	601270	602458
118503	153380	190950	600025	601293	602488
118920	153432	191161	600053	601363	602593
118945	153550	191170	600148	601418	602600
123670	154050	191306	600173	601423	602638
123841	154250	194535	600190	601443	602643
123856	155120	194540	600288	601473	602673
124070	156560	194545	600308	601508	602738
125650	162091	208400	600315	601583	602743
126390	162662	209920	600398	601588	602773
126410	163980	229000	600403	601623	602803
128992	164760	232400	600423	601658	602808
131230	165020	240500	600438	601663	602863
133250	165280	248310	600453	601698	602873
133535	165340	272370	600473	601725	602883
134830	167405	300060	600478	601728	602913
134934	170998	300065	600503	601838	602918
137164	172250	300145	600578	601843	602948
138150	173120	300165	600583	601883	603013
138297	173325	300167	600623	601913	603018
138322	176888	300170	600688	601918	603045
138680	176930	300185	600693	601948	603088

603093	605353
603123	605418
603153	605483
603158	605623
603233	605633
603235	605658
603238	605723
603338	605823
603348	605868
603378	605928
603433	606008
603443	606073
603448	606173
603465	606238
603493	606278
603558	606288
603593	606313
603658	606393
603693	606445
603758	
603763	
603823	
603863	
603873	
603908	
603913	
604013	
604078	
604163	
604248	
604463	
604588	
604643	
604668	
604698	
604718	
604818	
604883	
604958	
605023	
605043	
605163	
605168	
605193	
605273	
605308	

Cluster 2:

100050	157600	260530
100100	158170	264270
100500	158310	264810
101200	161000	265600
102800	162830	267010
102940	164180	267450
103400	164220	274000
107480	166300	276400
107601	167220	277480
108720	169720	300430
110000	170950	300465
113950	171750	301815
114120	176944	301845
117900	178500	310350
119550	179600	311810
120050	182410	555000
120180	184253	600083
122100	184754	600113
122780	191000	600218
122920	192400	600313
123880	193100	600905
125635	193230	600908
126180	194320	601078
126950	207780	601223
128800	208600	601298
131600	210700	601420
131850	211920	601668
132800	215550	602605
133600	215850	602613
136140	217600	603793
136590	219000	603828
137130	219095	604413
139100	223340	
139400	225060	
139630	233650	
142625	233800	
145250	236700	
146350	241800	
146530	246470	
147330	253230	
148200	255500	
148370	258150	
156530	258360	
157151	259600	

Cluster 3:

105120	256100	602378
107950	260600	602588
114100	263610	603218
115080	264700	603373
116100	268315	603463
116600	271150	603563
133020	271250	603663
134520	272650	604183
137150	274205	604393
142959	277350	604563
145900	277440	605583
159950	300030	605913
160200	300095	
160990	300210	
168300	300215	
176270	300355	
179800	300455	
182960	300550	
183100	300580	
184700	301790	
190310	301830	
202400	302900	
203800	304350	
206900	304700	
211770	306995	
212540	307200	
221600	309555	
221750	310300	
221995	312600	
223400	313850	
229050	314850	
229800	590105	
230400	600118	
235600	600175	
236800	600223	
241540	600333	
243080	600513	
246555	601253	
248190	601478	
250950	601608	
251000	601653	
252350	601808	
254000	602023	
254130	602083	

Appendix E: Acceptance test results

Standard k-means:

Terms / stemmed terms	Documents	Cluster	Relevance/precision
aarskog (aarskog)	100050	2	yes (1/1)
aarskog syndrome (aarskog syndrom)	100050 101200 128800 164180 119550 191000 100100 122920	2	yes (1/8) (5/8)
syndrome (syndrom)	100100 158170 128800 600118 236700 309555 101200 274000 219000 119550 107480 601608 223340 602588 212540 223400 604183 122780 164180	2	yes (17/19)
wolff parkinson (wolff parkinson)	602743 604588 600858 600438 601048 601418 180990 400033 601443 601838 600148 172250 232400 600583 601253 150340 108730 600853 114230	1	yes, the records on top. the rest are not so relevant (3/19) (2/19)
gastric sneeze stomach reflex (gastric sneez stomach reflex)	600333 192400 145900 600175 183100 271250 604563 271150 602378 603218 600223 603563 182960 302900 159950 602195	3	yes, if only the term reflex is considered, not so relevant if all is considered. (0/16) (0/16) (0/16) (9/16)

sneeze (sneez)	133600 162830 102800 107601 148370 120050 148200 105120 125635 161000 131850 277480 122100 158310 182410 217600 310350 233650 194320 103400 233800 246470 126180 171750 131600 267450 137130 252350	2	yes, if one considers the retrieval of the record 137130 as essential. the rest of the retrieved documents are not relevant. (1/28)
gastric (gastric)	602613	3	yes, although the term retrieved are gastrointestinal it is relevant, however it is not the desired record. (1/1)
mental retardation seizures (mental retard seizur)	300210 300095 300355 277350 229050 268315 179800 212540 246555 202400 309555 601608 211770 114100 253230 221995 304700 251000 223400 243080	2	yes, the application retrieves many records containing the query terms. (13/20) (14/20) (3/20)
persistent polyclonal lymphocytosis (persist polyclon lymphocytosi)	171750 606445 146350 102800 166300 148370 162830 300550 158310 246470 193100 131850	2	yes, however the desired record, 606445, is listed too low. (2/12) (1/12) (3/12)

embryonic ciliary ganglia (embryon ciliari ganglia)	600053 605483 600308 109770 603013 142310 300167 118945 603593 601130 248310 163980 124070 300145 154050 601583 108745 173325 604588 151250 601728 108730 600173 111680 602883 109530 107266 147390	100640 601623 104701 601270 605868 152424 606288 601363 190220 138680 601255 516040 134830 114230 146661 107273 170998 191306 606313 601883 164760 185510 600025 142600 300165 153432 177075 600148	1	yes, the record 118945 was the desired record. some of the other retrieved records are relevant to the query. (17/56) (4/56) (4/56)
muscular dystrophy (muscular dystrophi)	310300		2	yes, the desired record was retrieved. (1/1)
mitochondrial myopathy (mitochondri myopathi)	601443 601048 600858 601253	602743 590015 600438	1	yes, the desired record was 590015. most of the records are relevant (2/7) (5/7)
demyelinating (demyelin)	604563 600223 145900 302900 183100 271250	600333 159950 600175 602378 271150 182960	3	yes, the desired record was 145900. (2/12)

abetalipoproteinemia (abetalipoproteinemia)	314850 168300 604393 105120 605913 312600	603663 602023 300500 252350 133020 306995	2	yes, the desired record was 314850 which is listed on top. some of the other retrieved records contain the word protein and can be viewed as relevant. (1/12)
proteinase inhibitors serpin (proteinas inhibitor serpin)	606313 602058 605723 107273 139111 604818 605193 601883 182099 602453 191161 604698 606008 600403 107266 604463 165340 123841 603348 606288 131230 603093 600623 605658 602213 604643 600823 605163 300545 602098 600023 300185 601188 601193 602673	603018 604013 600583 602423 602913 189971 114210 104160 152424 300540 164760 600888 605928 190220 601293 167405 606278 603823 604958 124070 600315 601838 603378 602458 118920 165020 602488 173325 601423 603448	1	yes, of the top 65 retrieved records they are conceptually relevant to the query. (55/65) (27/65) (1/65)

Two step k-means:

Terms / stemmed terms	Documents	Cluster	Relevance/precision
aarskog (aarskog)	100050	2	yes, the desired record whas retrieved. (1/1)
aarskog syndrome (aarskog syndrom)	100050 101200 128800 164180 119550 191000 100100 122920	2	yes, the desired record, 100050, was retrieved. most of the records are relevant. (1/8) (5/8)
syndrome (syndrom)	100100 223340 128800 164180 236700 158170 101200 274000 219000 119550 107480 122780	2	yes (11/12)
wolff parkinson (wolff parkinson)	602743 604588 600858 600438 601048 601418 180990 400033 601443 601838 600148 172250 232400 600583 108730 150340 114230 600853	1	yes, the records on top. the rest of the records are not so relevant. (3/18) (2/18)
gastric sneeze stomach reflex (gastric sneez stomach reflex)	600333 159950 145900 600175 183100 271250 604563 271150 602378 603218 600223 603563 182960 302900	3	yes, if only the term reflex is considered, not so relevant if all is considered. (0/14) (0/14) (0/14) (9/14)

sneeze	133600	137130	2	yes, if one considers the
(sneez)	102800	162830		retrieval of the record 137130
	148370	107601		as essential. the rest of the
	148200	120050		retrieved documents are not
	125635	161000		relevant. (1/26)
	131850	277480		
	122100	158310		
	182410	217600		
	310350	233650		
	194320	103400		
	233800	246470		
	126180	171750		
	131600	267450		
gastric	-		-	no, cannot retrieve any
(gastric)				records.
mental retardation seizures	300210	277350	3	yes, the application retrieves
(mental retard seizur)	300355	268315		many records containing the
	229050	212540		query terms. (12/19) (13/19)
	179800	202400		(3/19)
	246555	601608		
	309555	114100		
	211770	221995		
	304700	251000		
	223400	243080		
	300095			
persistent polyclonal	171750	193100	2	yes and no, the record 606445
lymphocytosis	146350	102800		was the desired record which
(persist polyclon	123880	148370		the application fails to
lymphocytosi)	166300	246470		retrieve. (2/11) (1/11) (2/11)
	162830	131850		
	158310			

embryonic ciliary ganglia (embryon ciliari ganglia)	600053	100640	1	yes, the record 118945 was the desired record. some of the other retrieved records are relevant to the query. (17/56) (4/56) (4/56)
	605483	601623		
	600308	104701		
	109770	601270		
	603013	605868		
	142310	152424		
	300167	606288		
	118945	601363		
	603593	190220		
	601130	138680		
	248310	601255		
	163980	516040		
	124070	134830		
	300145	114230		
	154050	146661		
	601583	107273		
	108745	170998		
	173325	191306		
	604588	606313		
	151250	601883		
	601728	164760		
	108730	185510		
	600173	600025		
	111680	142600		
	602883	300165		
	109530	153432		
	107266	177075		
	147390	600148		
muscular dystrophy (muscular dystrophi)	310300		3	yes, the desired record was retrieved. (1/1)
mitochondrial myopathy (mitochondri myopathi)	601443	602743	1	yes, the desired record, 590015, was retrieved. (2/6) (4/6)
	601048	590015		
	600858	600438		
demyelinating (demyelin)	604563	600333	3	yes, the desired record, 145900, was retrieved. (2/12)
	600223	159950		
	145900	600175		
	302900	602378		
	183100	271150		
	271250	182960		

abetalipoproteinemia (abetalipoproteinemia)	314850 168300 604393 105120 605913 601253	312600 603663 602023 252350 133020 306995	3	yes, the desired record was 314850 which is listed on top. some of the other retrieved records contain the word protein and can be viewed as relevant. (1/12)
proteinase inhibitors serpin (proteinas inhibitor serpin)	606313 602058 605723 107273 139111 604818 605193 601883 182099 602453 191161 604698 606008 600403 107266 604463 165340 123841 603348 606288 131230 603093 600623 605658 602213 604643 600823 605163 300545 602098 600023 300185 601188 601193 602673	603018 604013 600583 602423 602913 189971 114210 104160 152424 300540 164760 600888 605928 190220 601293 167405 606278 603823 604958 124070 600315 601838 603378 602458 118920 165020 602488 173325 601423 603448	1	yes, of the top 65 retrieved records they are conceptually relevant to the query. (55/65) (27/65) (1/65)