UiT THE ARCTIC UNIVERSITY OF NORWAY

Faculty of Science and Technology
Department of Computer Science

User profiling for diverse user contexts

—

Jan Tore Karlsen

INF-3990 Master thesis - June 2015

Contents

Acknowledgements

First I would like to express my gratitude to my supervisor, Professor Randi Karlsen for her guidance, feedback and availability.

I would also like to express gratitude to my family and friends for the discussions, good times and memories supporting me.

Finally, a special thanks to my girlfriend, Silje Skogli for all the support and motivation.

List of Figures

List of Tables

List of Algorithms

Abstract

The amount of content available for consumption online is increasing tremendously. This make the job of recommender systems more important, and at the same time, more demanding. Context-aware recommender systems might be a solution to this problem.

This work set out to discover user contexts dynamically by collecting contextual information from user actions and perform cluster analysis on the data collected. User interests are collected from user actions as well, and are sorted into groups based on the contexts discovered. These sorted interests are considered the users' user profile. The user profiles are in turn used to recommend news articles based on the interests of the users, where the users can select what context to receive recommendations from.

The results and evaluation of the system show that the approach used in this work is not very successful and adjustments are recommended to improve the results. The system designed and implemented in this work is only able to identify two very broad contexts based on user data.

Chapter 1

Introduction

This chapter is an introduction to the thesis, it presents the problem statement, which is the basis for this thesis, the motivation for doing this work, a brief summary of the work and finally the structure of the remaining thesis.

1.1 Problem statement

> "The goal of this thesis is to design, implement and test a system that generate user profiles based on user interests and available contextual information. The user profiles shall reflect the users' contexts or situations and provide the users' interests based on context."

1.2 Motivation

The volume of information on the internet is increasing at a tremendous rate[8]. For users this means more content to consume, but also more content to sift through before finding information of interest. People can struggle to find the intended information in the web, even with the help of search engines[11]. To locate content, filtering and recommendation systems are often used. This result in users discovering content the system deems relevant based on the users' interests or preferences stored in a user profile. The information contained in the user profile in non-intrusive systems is often collected by monitoring the users' actions in the system. There are also profiles built by the user manually by answering a questionnaire[19][28]. These techniques for building a user profile treats the user as one entity with a consistent set of interests and preferences. This is not necessarily the case in a lot of systems since users use the web and computers for a number of different tasks and in a number of different circumstances. Users searching for topics for a school project they have no interest in, can be plagued with suggestions related to the school project in a traditional user profile. If the user's interests had been gathered in regards to context and the user could choose to ignore the interests recorded in relation to the school project then the recommendations could be more accurate. Also if a user could specifically select a single context or situation to get recommendations from, the user could get better tailored recommendations instead of general ones representing all the user's situations. This could increase the user experience for systems generating a user profile bases on interests by providing a higher recommendation accuracy and better customization.

1.3 Approach

The creation of a context-aware user profile and a recommender system designed and implemented in this thesis can be divided into three parts. First there is a collection part where user interests with contextual information is collected and stored. The user interests and contextual information is a necessity for creating a user profile.

The second part is the process of creating the user profile with the user interests and contextual information. A major challenge here is the automatic discovery of user situations. This is done by utilizing cluster analysis on the contextual information to identify groups of similar contexts. These groups are treated as separate user contexts. The interests associated with the contextual information is grouped in the same manner and this constitute the user profile.

The third and last part is a recommendation system whose role is to utilize and test the user profiles created. The recommendation system collect articles from the web and presents them to the user using content-based filtering, based on the interests in the user profile and a chosen context.

The approach is discussed in detail in chapter 4.

1.4 Contribution

The contribution of this work is to explore the possibilities of user profiles to reflect users' context combined with interests. Identifying the different contexts based on simple information gathered from user web browsing is the core task of this work and the main contribution. To my knowledge there is no other work that discover user context based on clustering contextual information while treating computers and domains visited as a physical and digital location for the user actions. Further, this work contribute with the evaluation of the proposed system and ideas for further improvements.

1.5 Structure

The rest of this thesis is structured as follows. Useful background information is presented in chapter 2. Related work with elaboration on similarities and dissimilarities is in chapter 3. The approach and design of the work is presented in chapter 4. Architecture and implementation is in chapter 5. The results of the work done is presented in chapter 6. Future work is discussed at the end in chapter 7. The conclusion is in chapter 8.

Chapter 2

Background

This chapter cover some of the important topics that needs to be familiar for the rest of the thesis. The concept of context is covered in section 2.1, data cleaning in section 2.2, user profiling in section 2.3, content modelling in section 2.4, recommendations in 2.5, clustering in 2.6 and time series analysis in 2.7.

2.1 Context

When humans communicate they have the ability to use situational information to increase the communication bandwidth. This ability does not work well in communication between humans and computers. Computers does not have the ability to read and process the situational information without specifically being told to do so. Computers specifically told to read and process situational information is called context-aware computing[2].

Context is a concept used in a wide variety of fields for many different reasons. This result in context having a significant amount of definitions tailoring context to individual fields[3]. A broad definition of context can be found in the Merriam-Webster online dictionary. Merriam-Webster define context as "the situation in which something happens : the group of conditions that exist where and when something happens"[1]. Gregory D. Abowd and Anind K. Dey[2] define context as "any information that can be used to characterize the situation of an entity, where an entity can be a person, place, or physical

[1]`http://www.merriam-webster.com/dictionary/context` [Visited: 13. March 2015]

or computational object". Both are in essence the same. Context is the group of conditions or situation surrounding an entity.

Schmidt et. al[37]. categorizes context into six high level categories. They include three human factors and three physical factors. The human factors are the user (e.g. age, habits), social environment (e.g. social interactions) and tasks (e.g. active tasks, goals). While the physical factors are conditions (e.g. weather, light), infrastructure (e.g. resources) and location.

2.2 Data Cleaning

Data cleaning is often done as a stage in data processing to prepare the data for the next stage. Data cleaning can be performed on all types of data, but this section focus on cleaning textual data. Textual data or documents can be cleaned for a number of reasons. Some of the most popular are to reduce the size and noise of the dataset by removing words and characters that are not useful for the given purpose. Some popular techniques is covered in the following subsections.

2.2.1 Part-of-speech tagging

The process of identifying certain class of words in a sentence is known as part-of-speech tagging (POS tagging), and this is done while analysing sentences to understand them[30]. Part-of-speech tagging is part of the Natural Language Processing (NLP) field. The technique relies on analysing the entire sentence to determine what context words are used in. Even a simple word as "dogs", which usually is consider a simple plural noun[2], can also be a verb[3] in the sentence "The sailor dogs the hatch". Parsing a sentence for Noun phrases or unknown proper nouns without a large cohesive text for context is next to impossible since NPL requires a lot of context for the machine learning aspect to successfully function. There are a lot of scenarios where words get classified differently based on the meaning of the sentence. As an example the simple sentence "the car park" contains the words "the", "car" and "park". This sentence can be POS tagged in three different ways at least. The words "car" and "park" can be identified as nouns, or it can

[2]http://www.merriam-webster.com/dictionary/noun
[3]http://www.merriam-webster.com/dictionary/verb

be identified as a single noun "car park", or the word "car" can be identified as an adjective and "park" as a noun. After POS tagging is performed the desired classes of words can be kept and the other can be discarded. Before POS tagging can be performed the language need to be identified, without knowing the language, the system does not know what grammatical rules apply[30].

2.2.2 Stop words removal

A simple technique for removing unwanted words are stop words removal[4]. Stop words are commonly thought of as words without direct meaning, or supporting words in sentences. Table 2.1 contain a sample of common stop words. There is no absolute or complete list of stop words since the list of stop words should reflect the desired goal by removing them. A very comprehensive list of stop words can be found in MySQL database software[5]. Removing stop words from the user's interests is a very simple task, and to support several languages simply provide a list of stop words for each language. The system does not need to identify the language before removing the stop words.

2.2.3 Regular Expression

Regular Expression[6] (Regex) is often used to remove unwanted characters or character sequences from text. Regex can remove characters based on classes. For example can Regex be used to remove upper case characters, digits, hyphens, the letter "a" etc. Regex can remove patterns of characters, repetitions of characters or patterns and patterns based on the location of it. Regex is a powerful tool for removing or filtering text.

2.3 User profiling

This section cover fundamental information about user profiles and related subjects in regards to building them. For a system to be able to provide an

[4]`http://www.ranks.nl/stopwords`
[5]`https://dev.mysql.com/doc/refman/5.1/en/fulltext-stopwords.html`
[6]`http://pubs.opengroup.org/onlinepubs/007908799/xbd/re.html`

being	should	ourselves	further	ours	what
if	your	most	their	same	own
to	will	you	in	yourselves	until
here	both	against	these	it	or
so	myself	them	there	more	not
any	such	had	by	during	doing
am	a	than	its	on	she
between	only	were	some	because	which
that	where	he	did	our	above
was	out	can	no	having	as
after	are	themselves	him	before	just
again	me	with	they	those	then
once	few	the	yourself	when	why
other	into	theirs	now	too	itself
at	herself	up	very	off	an
be	i	and	for	been	himself
under	all	t	nor	whom	have
hers	below	do	s	does	this
we	his	each	my	is	yours
how	over	of	don	from	who
through	down	has	but	about	while
her					

Table 2.1: Small sample of common English stop words

experience tailored or adjusted for users' interests the system need to know something about each user. In personalization systems the user profile is a central component because the system need relevant information about the user to be able to recommend or filter content to that specific user. A popular way to build the user profile is with a list of keywords generated by high-frequency words from the user-browsed web pages[40] but this vary with the requirements for each specific system. Typically a user profile is created in regards to the information the system need or should know about the user. This information can typically be general user information (e.g. age, gender), user behaviour, context, interests and intentions, which is represented by terms of keywords[32][19].

If the goal is to provide the user with appealing and relevant advertising, as it often is for user profiles, the user profile should contain as much information as possible about a user's interests and other contextual information. If the user profile contains the user's current location, this can for be used to provide advertising about nearby stores and facilities. This should could be much more relevant to the user than advertisements of stores that might not be available to the user.

User profiles can either be constructed on the client or on a server receiving information about the user. Storing the user profile on the client has a few advantages, given that the system requirements allow for it. The profile can contain sensitive information about the user without this data being sent where the user no longer has direct control over it. The system can outsource the storage job to the client and relieve itself of this duty, simplifying the server requirements. Systems might have requirements that does not allow for client based user profiles. An example of this is recommender systems with collaborative filtering (covered in subsection 2.5.3) where user profiles are compared against each other to make recommendations. Other restrictions for storing the user profile on the client might be size limitation on the clients or computational requirements in regards to the user profiles that the client does not have. Finally storing the user profile on a server instead of the client give much greater data durability if the server is configured correctly.

Some information that can be typically stored in a user profile is covered in the following subsections.

2.3.1 Behaviour

Behaviour modelling can be constructed by analysing user history and discovering patterns[19]. The user history can be constructed by logging the different actions a user performs in the system. These actions could be general navigation like clicking hyperlinks and opening or consuming items or content in the system. By storing this data and maintaining a timely structure the system can analyse the data and try to determine what action the user might want to do next. By constructing a behaviour model for the user the system can as an example determine the effectiveness of the user interface or the effectiveness of advertisement campaigns.

2.3.2 Context

What contextual information a user profile contains reflect specific application requirements. Applications like Netflix[7] might use data like user age, gender and location to improve recommendations based on demographics. Mobile applications can be using location and connectivity information to improve the user experience. Context in it self is covered further in section 2.1.

2.3.3 Interest

Users' interests are the single most important information in recommendation systems. The users' interests are used to locate items or content that suits the users best[19]. Users' interests are often stored in keywords with a weight value to determine the importance of the keyword[28]. User profiles with interests as keywords are often used on documents or other items containing text. The relevant data can then be retrieved by comparing the interests in the user profile with item metadata containing relevant tags or keywords from document modelling techniques like TF-IDF[19] (covered in section 2.4). Systems with limited textual information often prefer to use collaborative recommendation filtering, which rely on a score users give items instead of interests represented by keywords[19]. You can read more about the collaborative recommendation filtering in chapter 2.5.3.

2.3.4 Intention

Intention modelling is done to identify the goal of the user in the system[19][18]. An example of this would be an online store identifying users with the intention of buying a product, and users without the intention of buying a product. A user intention can be classified into two levels: action intention and semantic intention[12]. Action intentions are low level intentions like a mouse click, keyboard stroke or page navigation. Semantic intentions are the intentions of what a user want to achieve on a higher level. Such an intention might be to buy a book on Amazon[8]. A high level semantic intention might include several low level action intentions to achieve its goal[29].

[7]`http://www.netflix.com/`
[8]`http://www.amazon.com/`

Intention modelling is largely based on a classification system that has a set of predefined categories[18]. There are several methods for identifying user intention. Some of the most popular are SVM (support vector machines) and Bayesian networks[18][19]. Both methods uses classification and supervised machine learning to achieve intention modelling. Intention modelling is at a higher level than behaviour and interest modelling. Analysing user intentions is based on user behaviour, interests and context[19]. Note that tasks, which is one of the six high level categories of context mentioned in section 2.1, can be interpreted to be the same as intention. So based on that classification, intention might be a part of the context.

2.4 Content modelling

For personalisation and recommendation systems, content modelling is an important aspect. The system need to be able to identify the content of documents or content. This is usually done through keywords, which is primarily obtained in two different ways[19]. The first is descriptive metadata, which is often used on content without text that can be easily parsed. Content like this might include videos, pictures and objects representing physical items (e.g. item on Amazon[9]). The second way is to use content modelling techniques like TF-IDF, which is short for term frequency–inverse document frequency. This is common to use on content containing text that can be analysed. This technique works by setting the term importance as the term frequency in the document, since the more the term occurs in the document, the better suited the term is for describing the document. Only relying on term frequency is not good since some terms are more common than others. For example the word "The" is very common, and is very likely to be found a lot in a document, but it is not suited to describe the document. Inverse document frequency is therefore important for highlighting terms that occur in a particular document compared to the rest. The importance of the word "The" would have gotten reduced significantly because of the number of occurrences in other documents.

Term frequency can be formulated as:

$$TF(t) = \frac{Number\ of\ t\ occurrences}{Total\ size\ of\ document}$$

Where the parameter t is the term the term frequency shall be found for.

[9]`http://www.amazon.com/`

Inverse document frequency can be formulated as:

$$IDF(t) = \log \frac{Total\ number\ of\ documents}{Number\ of\ documents\ t\ occur\ in}$$

Finally TF-IDF can be formulated as:

$$TF - IDF(t) = TF(t) * IDF(t)$$

2.5 Recommendation

Recommendation methods uses the information in the user profile to find the best matching results, be it movies, search results or advertisements. The user profile is typically modelled to fit the information the recommendation method requires to perform optimally.

Min Gao et al.[19] believes there are four filtering approaches for making recommendations. (1) rule-based filtering, where the results are determined by specified rules like "If user has selected math as an interest, show math related items", (2) content-based filtering, based on the users rating of content, recommend content similar to content rated high by the user, (3) collaborative filtering, which recommend content based on users with similar tastes rating of the item, and (4) hybrid methods of filtering, which uses techniques from some or all of the other approaches. The four filtering approaches is covered more in depth in the following subsections.

2.5.1 Rule-based filtering

The rule-based filtering is the simplest approach to recommendations. In this approach the system specifies rules that determine what results are shown based on certain information known about the user. The rules and classes of users are predefined and thus limits the complexity and flexibility of the system to a degree. Systems using rule-based filtering are most likely relying on users answering questions or filling out forms to provide the system with information like name, age and preferences. An example of rule-based filtering is when a user provides the system with information about his preferences by selecting among predefined topics provided by the system, and then the system return content related to the selected topics.

2.5.2 Content-based filtering

Content-based filtering relies on comparing the user profile with the description of items to calculate the degree of relevance[19]. The user profile can in this case consist of keywords or tags of movies that is weighted equally as the rating given to the movie by the user. The system can then recommend movies with keywords or tags similar to the users most weighted keywords. Content-based filtering is very effective in text heavy domains like news recommendation because the user profile can contain keywords from the users browsing activities and use these to match news articles that are relevant. For content-based filtering to work on multimedia content the content need to have describing metadata.

A negative side effect of using content-based filtering is the discovery of new content. If a new class of content is available in the system, the user might not have any keywords relating to it, and will as a result not be recommended items from the new class. This is also be true for undiscovered content in general, if there is no relevance between the user profile and the content, there is no match, so it is easy to be stuck with one type of content.

2.5.3 Collaborative filtering

While content-based filtering compares the user profile with the description of items, collaborative filtering compares user profiles with other user profiles and recommends items based on what is popular for similar user profiles[19]. This works by sorting users with similar interests into preference groups by comparing users likes and dislikes. The collaborative filtering approach relies on users discovering content by themselves, and with enough content discovered, the user can be matched to a preference group. Once this is done the user will be recommended items with high probability of being of interest since users that enjoyed the same content as the user have enjoyed it. A problem with the collaborative filtering approach is the lack of usage information on new items, this make them impossible to recommend[28]. Since the approach recommends only based on similar users preferences, the system does not need to know any information about the item being recommended other than the rating other users gave it. This results in this approach being very effective in domains where items might lack textual information such as e-vendors like Amazon[10]. With this approach the computation costs increases

[10]`http://www.amazon.com/`

linearly with the number of users and items[19]. This makes the approach not very scalable.

Collaborative filtering can also be used as an extension to content-based filtering by building an item to item similarity matrix rather than a user to user similarity matrix. This results in very fast recommendations because of the pre computed similarity model.

2.5.4 Hybrid filtering

Hybrid filtering is simply put a mixture of both content-based filtering and collaborative filtering. Using content-based filtering solve the new item problem and the complex computation problem that collaborative filtering suffers from. Using collaborative filtering on the other hand, solve the new item class problem that can occur in content-based filtering. A system using both content-based and collaborative filtering work very well in certain domains where the mentioned problems might occur, like e-commerce[19].

2.6 Clustering

This section cover clustering with a number of different clustering algorithms and distance measures as well as techniques for evaluating clusters. Clustering is the unsupervised task of grouping (clustering) items in a dataset such that each group or cluster contains the items in the dataset most similar to each other, and dissimilar to items in the other clusters. Clustering is a technique widely used in many fields for many purposes, and as a result have many different approaches. Some of the most popular approaches are Hierarchical clustering, K-means and DBSCAN[39]. These approaches is covered in subsection 2.6.1, 2.6.2 and 2.6.3. Clustering is a form of data analysis where the distance between the items in the dataset is calculated and the distance matrix is used to group items by different algorithms. The distance measure between items is a metric or quasi-metric on the feature space used to quantify the similarity of items[22]. The distance measure is discussed further in subsection 2.6.4. Cluster evaluation is the task of evaluating the quality of the clusters after performing cluster analysis. It is important since the quality of the results can vary greatly. Cluster evaluation is covered in subsection 2.6.5.

2.6.1 Hierarchical clustering

In Hierarchical clustering the objects in the dataset is clustered in a hierarchy of clusters. The hierarchy of clusters can be built Agglomerative or Divisive (also known as bottom up or top down)[39]. The different stages in Hierarchical clustering is based on the distance between the objects.

Algorithm 1 Basic agglomerative hierarchical clustering algorithm[39].

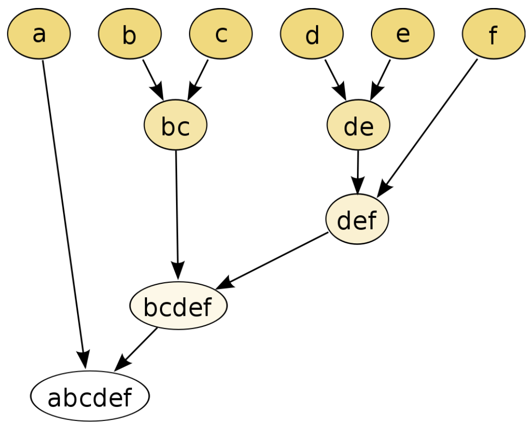
```
1: Compute the proximity matrix.
2: repeat
3:     Merge the closest two clusters.
4:     Update the proximity matrix to reflect the proximity between the
       new cluster and the original clusters.
5: until One cluster remains.
```

Object distance or similarity can be stored in a distance matrix and must be a numerical value. The algorithm for agglomerative hierarchical clustering can be found in Algorithm 1. For the agglomerative approach, first, objects in the dataset are clustered with each object placed in a separate cluster[39]. This can be seen in the top row in figure 2.1, each letter is placed in its separate circle. The clusters within a pre-determined distance interval are then joined and a new distance is calculated between the clusters that was joined and the other clusters. This last stage repeats until all clusters are combined in one single cluster. This stage can be illustrated by following the arrows in figure 2.1. For the divisive approach the same process as described is done in reverse, as the objects start in a single large cluster and are divided in steps. The distance between the clusters needed to define the ”closest clusters” can be measured in several ways. The most popular ways are to use the two closest objects in the clusters (single-linkage), the two furthest objects in the cluster (complete linkage) or to use the mean or average value of all the objects in the cluster (mean or average linkage). The clustering results may vary greatly depending on this.

2.6.2 K-means

K-means is a clustering technique that attempts to find a user specified number of clusters (K). These clusters are represented by their centroids. A centroid is in mathematics the arithmetic mean position of all the points in a given shape. This is the same in clustering where the centroid is the average

Figure 2.1: A dendrogram displaying hierarchical clustering results[13].



position of all the objects located in the given cluster. K-means is also one of the oldest and most used clustering methods[39]. The basic algorithm for K-means clustering can be found in algorithm 2.

Algorithm 2 Basic K-means algorithm[39].

```
1: Select K points as initial centroids.
2: repeat
3:     Form K clusters by assigning each point to its closest centroid.
4:     Recompute the centroid of each cluster.
5: until Centroids do not change.
```

First the user must specify the number of clusters (K) desired. Each object in the dataset is then assigned to the closest centroid in the cluster. The centroids for each cluster is then recomputed based on the objects in them. This is repeated until the centroids do not change, then all objects are allocated to the closest cluster.

2.6.3 DBSCAN

DBSCAN[16] is a density based clustering algorithm, this means the technique locates areas with high density separated by low density areas. Opposite to Hierarchical clustering and K-means, DBSCAN does not necessary cluster all objects in the dataset. Objects not in a high density area is regarded as noise, while objects with many neighbours are core objects. Objects with few neighbours but in range of core objects are regarded as border objects. This let it classify objects as noise, core or border.

Algorithm 3 Basic DBSCAN algorithm[39].

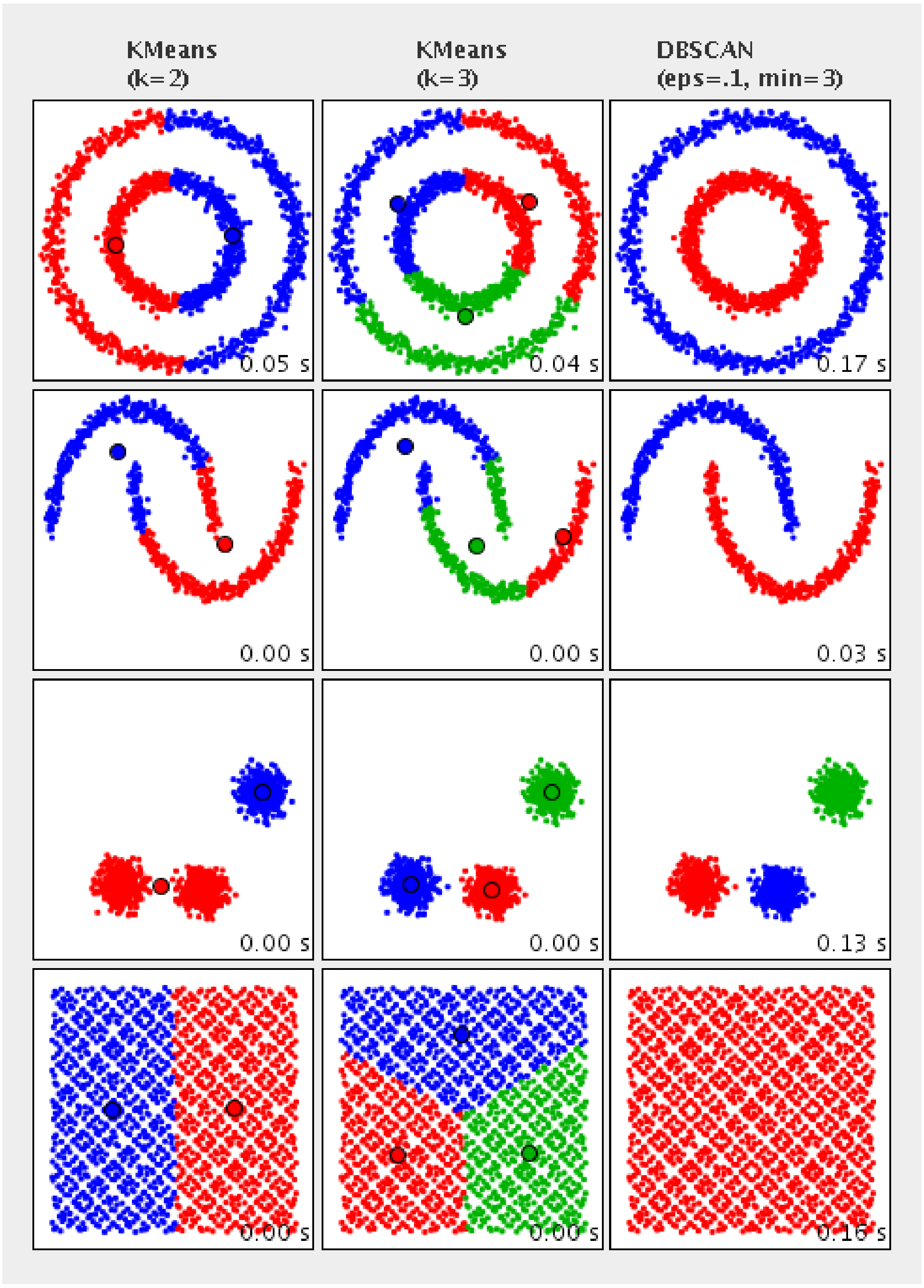
```
1: Determine object scan radius distance.
2: Label all objects as core, border or noise based on distance matrix.
3: Eliminate noise objects.
4: Make each group of connected core objects a cluster.
5: Assign each border object to the appropriate cluster.
```

A simple example of a DBSCAN algorithm is presented in algorithm 3. DBSCAN relies on two user specified values. The first one is the distance to scan for other objects in the dataset. This will impact the result in a significant way since a distance too short, relative to the object density, will result in no clusters found or a number of smaller clusters where the density is highest. Most object will be regarded as noise, and this is in most cases not good. The result of setting the distance too long is that too many objects will be part of the cluster, and data that originally would have been separate clusters can be grouped as one. The distance can and should be set in response to the density of the dataset, and how dense the objects should be to qualify to be clustered. The second user specified value is the minimum number of objects required to create a cluster. This is simply to avoid each pair of objects close enough creating their own cluster if this is not a wanted outcome. Since DBSCAN is a density based clustering technique it is able to identify clusters with many different shapes and sizes. It is also relatively robust in regards to noise. This make DBSCAN able to locate many clusters impossible to locate with hierarchical clustering and k-means. DBSCAN is however weak in regards to high variety of density since density is harder to define. A clustering comparison between k-means and DBSCAN highlighting scenarios DBSCAN is able to recognize clusters k-means is unable to, can be seen in figure 2.2.

2.6.4 Distance

In cluster analysis there is a fundamental need to measure the distance or similarity of the objects in the dataset. Without this distance or similarity it is virtually impossible to place the objects in vector space relative to other objects. Distance is a quantitative degree of how far apart two objects are. Distance measures that satisfies the metric property are called distance metric, while non-metric measures are called similarity[9]. In this paper the

Figure 2.2: A comparison between K-means and DBSCAN clustering[17].



term distance is used to cover both distance metric and similarity. The distance measure depends on the attribute type and scale of the data. Data can have attributes as presented in table 2.2. Since there is a variety of attribute types and scales for objects, the distance measure chosen impact the results significantly[22]. Objects are usually represented as points (vectors) in multidimensional space where each dimension represents a distinct attribute (variable) describing the object[38]. In a data matrix the objects are represented as a m x n matrix, where m represent the number of objects and n the number of attributes. Attributes in a data matrix is sometimes transformed before use. This is because different attribute values can be on different scales. Attributes using different scales can impact the result of the cluster analysis significantly. It is common to standardize the data so that all attributes are on the same scale. Another reason to transform the data can be to reduce dimensions or attributes. In high dimensional space distance between points become relatively uniform[38]. While some clustering algorithms use the original data matrix, many clustering algorithms use or can use a similarity matrix. A similarity matrix is a n x n matrix mapping the distance between the objects n.

Quantitative	**Continuous** (e.g., weight) **Discrete** (e.g., number of kittens in a litter) **Interval** (e.g., the duration of an event)
Qualitative	**Nominal** or **unordered** (e.g., color, religion) **Ordinal** (e.g., military rank, good-better-best)

Table 2.2: Attribute types.

There are a lot of distance measures to chose from. The following subsections briefly elaborate some distance measures that are commonly used for clustering.

Euclidean distance

Euclidean distance or Euclidean metric is the straight line distance between two points[9]. In a plane space with p (p1, p2) and q (q1, q2) the distance is:

$$d(p,q) = \sqrt{((p1 - q1)^2 + (p2 - q2)^2)}$$

For points in N dimensions, the Euclidean distance between two points can be found with the formula:

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(pi - qi)^2}$$

Where pi (and qi) are the coordinates of p (and q) in dimension i.

Manhatten distance

Manhattan distance or Block distance is the distance between two points measured along the axis at right angles[9]. In a plane with the points p (p1, p2) and q (q1, q2) the distance is:

$$d(p,q) = |p1 - q1| + |p2 - q2|$$

For n dimensional space, the distance is:

$$d(p,q) = \sum_{i=1}^{n} |pi - qi|$$

Jaccard distance

Jaccard distance or Jaccard index is used for measuring the dissimilarity between datasets. It is calculated by dividing the difference of the sizes of the union and the intersection of two sets by the size of the union[9]:

$$d(p,q) = \frac{|p \cup q| - |p \cap q|}{|p \cup q|}$$

Jaccard distance is useful for data objects contains binary attributes or in some cases for nominal attributes.

Cosine similarity

Cosine similarity is a measure of similarity between two vectors that measure the cosine of the angle between them[9]. Cosine similarity is most often used in high dimensional space. Applications producing data with many dimensions are typically text mining, where each term is assign a different dimension and a document is a vector where the value of each dimension is the number of times the term appears in the document. For the points P and Q the formula for cosine similarity is:

$$d(P,Q) = \frac{\sum_{i=1}^{n} PiQi}{\sqrt{\sum_{i=1}^{n}(Pi)^2}\sqrt{\sum_{i=1}^{n}(Qi)^2}}$$

2.6.5 Cluster evaluation

Cluster evaluation is a task performed to control the results of clustering algorithms. The two most important aspects of cluster quality is that inter-cluster distance is high, and that intra-cluster distance is low[25]. In other words that the distance or dissimilarity between clusters are as high as possible, and that the items in the clusters are as similar or close as possible.

The Silhouette index is an index to measure cluster quality[35]. The Silhouette index has the following equation for K samples in the dataset[34].

$$S = \sum_{i=1}^{k} \frac{B(i) - A(i)}{max\{A(i), B(i)\}}$$

Where B(i) is the average nearest-cluster (inter-cluster) distance between the sample and the nearest cluster, which measures how spread apart the sample and the closest cluster are. A(i) is the average within-cluster (intra-cluster) distance between the sample and all other data in the cluster.

The results from the Silhouette Coefficient is in the range of -1 to 1 where the best value is 1 and the worst value is -1[11]. Values near 0 indicate overlapping clusters and negative values generally indicate that samples are assigned to the wrong clusters. The higher value returned from the Silhouette index, the better the clusters are.

2.7 Time Series Analysis

This section cover time series analysis with different measures. To compare two time series and calculate the similarity between them, techniques from

[11]`http://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html`

time series analysis can be used. A time series is a sequence of data points typically consisting of sequential measurements made over a time period. Time series can be difficult to compare based on a number of factors. There can be a difference in length, noise points, misalignment and local time shifting[15]. There are several ways to compare the similarity of two time series, and the following subsections cover some of them.

2.7.1 Euclidean distance

Euclidean distance is largely covered in subsection 2.6.4, but not in regards to time series. Euclidean distance is a lock-step measure, this means that the points in the time series compared are fixed. Since each point is fixed to the corresponding point in the other time series the measure is extremely sensitive to noise and misalignments in time[15]. Euclidean distance is also unable to handle local time shifting or in other words similar segments that are out of phase[15].

2.7.2 Edit distance

Edit distance is the minimum number of operations needed to make one sequence similar to another[33]. The basic edit distance measure is not very well suited for time series analysis, so several distances based on edit distance are developed[33]. The best known such distance is LCSS, which is short for longest common subsequence[33]. This measure define a threshold parameter that allow two points from different time series to match if they are inside this threshold of each other[33]. The distance is based on the longest common subsequence between two time series, and can be a very good measurement if the data is not time shifting.

2.7.3 Dynamic Time Warping

Dynamic Time Warping (DTW) is a method used to compare the optimal match between two sequences[33]. It was originally created to be used in analysing speech in speech recognition since it allow time series to stretch and compress in order to provide a better match[7][31]. DTW is not restricted to comparing time series in lock-step, but can compare points one-to-many[33]. As can be seen in figure 2.3, DTW works by warping the time axis iteratively in order to find the best possible match for the time series. This make the measure capable of handling local time shifting and a varying degree of points very well. To measure the distance between two sequences, DTW need to create a M x N distance matrix based on a simple distance like Euclidean. The shortest path is the straight line from the bottom left corner to the top right corner. The warp path between the sequences is the path from the lower left corner to the top right corner with the lowest distance values. See figure 2.4 for an illustration on the lowest distance values. The sum of the distances in the lowest distance path is the distance between the sequences. There is no limit to how big this distance can be, so the resulting value is a continuous value.

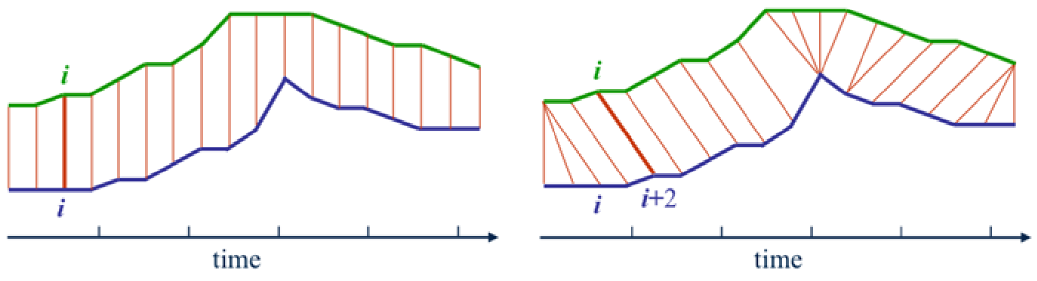
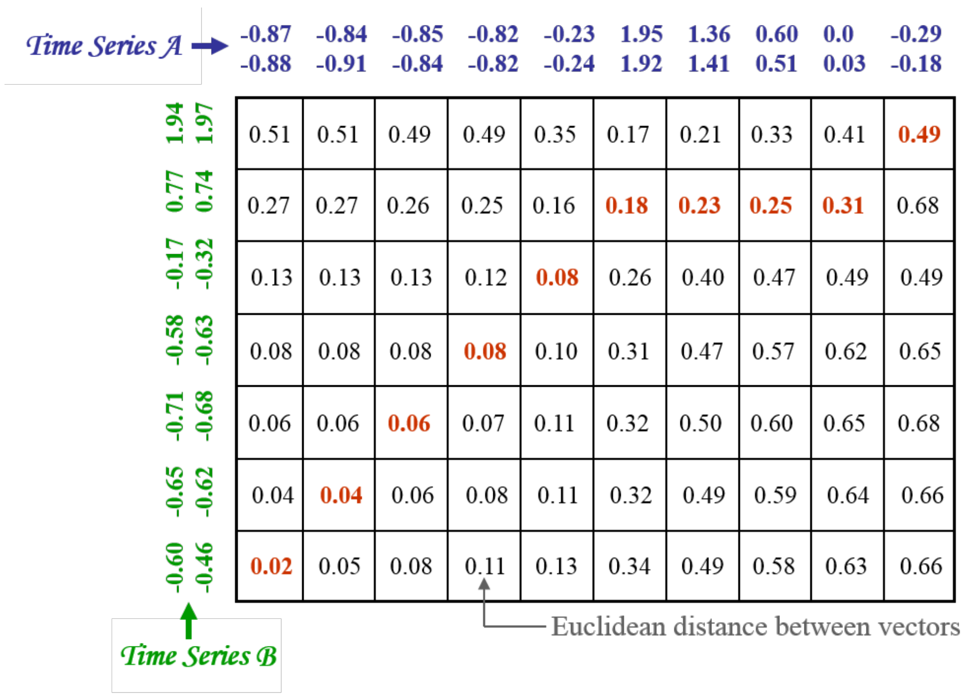


Figure 2.3: Dynamic time warping example[41].



Time Series B \ Time Series A	-0.87 -0.88	-0.84 -0.91	-0.85 -0.84	-0.82 -0.82	-0.23 -0.24	1.95 1.92	1.36 1.41	0.60 0.51	0.0 0.03	-0.29 -0.18
1.94 1.97	0.51	0.51	0.49	0.49	0.35	0.17	0.21	0.33	0.41	**0.49**
0.77 0.74	0.27	0.27	0.26	0.25	0.16	**0.18**	**0.23**	**0.25**	**0.31**	0.68
-0.17 -0.32	0.13	0.13	0.13	0.12	**0.08**	0.26	0.40	0.47	0.49	0.49
-0.58 -0.63	0.08	0.08	0.08	**0.08**	0.10	0.31	0.47	0.57	0.62	0.65
-0.71 -0.68	0.06	0.06	**0.06**	0.07	0.11	0.32	0.50	0.60	0.65	0.68
-0.65 -0.62	0.04	**0.04**	0.06	0.08	0.11	0.32	0.49	0.59	0.64	0.66
-0.60 -0.46	**0.02**	0.05	0.08	0.11	0.13	0.34	0.49	0.58	0.63	0.66

Figure 2.4: Shortest warping path for DTW[41].

Chapter 3

Related work

This chapter cover related work and some of the differences in regard to this work.

Hung-Jen Lai et. al. compare the results of tailoring the news based on a user profile to the standard editorial pick in their paper *Customized Internet news services based on customer profiles*[27]. In their system they determine the users' interests in an article based on the time used reading the article. They take the relevant keywords, nouns (identified by WordNet[1]), identify synonyms, and calculate the frequency among other things to eliminate noise. The recommendations is then determined by calculating a match between articles and the weighted keywords in the user profile for the user. Their results show that this approach outperforms the traditional editorial picks. Hung-Jen Lai et. al's work is related to this work in that both utilize indirect information retrieval to determine the popularity of recorded keywords to recommend articles to the user. Their approach to use reading time to determine the level of interests work for them because they are operating strictly on the website of China Times[2], while this work collect interests from all over the web. Measuring the level of interest based on reading time would be very inaccurate on websites with multimedia content, as time spent on the page would be long compared to the amount of text, which would inflate the perceived interest. Based on the user this could be common in this work, and thus this approach might not be appropriate.

Annie Chen's paper *Context-Aware Collaborative Filtering System*[10] use

[1]`https://wordnet.princeton.edu/`

[2]`http://thechinatimes.com/`

context in a collaborative filtering system to predict a user's interests in different contexts. Her work is related to this work in that both set out to recommend content based on the users' context and need to compare contexts in the process. Her work utilizes collaborative filtering to predict the users' interests in their given context, while this system use clustering to detect contexts and to sort the users' interests based on this, and then letting the user choose the context to get recommendations from.

Mark van Setten et. al. has created a system that combine context-awareness and interest recommendations into a system for tourists called *COMPASS*[42]. The system serve the users with information and services based on the tourists' context and interests. Their system is related to this work in that it combine context-awareness with recommendation and separate criteria into hard and soft, with context being a hard criteria in both their and this work, and interests being a soft criteria in both. *COMPASS* use the mobile network or GPS receivers to determine the users' location and show this on a map. The map display objects of interest for the user based on the user's profile, and can contain buildings, buddies, restaurants, shows or other objects. The system is built on a platform called WASP where third parties can register their services to be discovered. The items recommended are different for the two works, as well as the technique used to recommend items. The *COMPASS* system also use different contextual information to recommend items, such as last time visited.

Sofiane Abbar et. al. combine the ratings of users on items with the users' contextual information in their system[1] to provide more accurate recommendations based on user context. Similarly to this work, their system use clustering to determine the contexts of the users based on the contextual information collected. They describe in their approach that they use hierarchical clustering to determine the number of clusters, and then use k-means to cluster the data. This approach is different from using the elbow point to determine the number of clusters and use hierarchical clustering to cluster the data, as is done in this work. They deploy collaborative filtering to recommend content to the users while this work use content-based filtering with keywords describing user interests to recommend content.

Chapter 4

Approach and design

This chapter cover the approach for solving the problem statement along with the design details for the work. An overview of the work and the general approach can be found in section 4.1. The three main components of this work, information gathering, user profile creation and recommendation, is covered in section 4.2, 4.3 and 4.4.

4.1 Overview

The system's design can on a high level be divided into three main mechanisms. First there is a data collection mechanism that collect the interests and contextual information from the users' actions. The system gather and store information describing users' interests and contextual information from their web activities in a non-intrusive manner. The user interests and contextual information is gathered from the same events, so they are in relation to each other. This step include a process of cleaning the information collected.

Second there is a mechanism sorting the interests based on the contextual information to reflect the different contexts. In this step the system analyse the information gathered and create user profiles based on the results. This is achieved by using cluster analysis on the contextual information. The clusters created by the cluster analysis contain the most similar contextual information while the contextual information in different clusters are as dissimilar as possible. The clusters represent the different user contexts and by sorting the user interests by the clustering results the system can create a user profile that reflect the users' contexts.

Third there is a recommendation mechanism that utilize the user profiles generated in the second mechanism and recommend articles from the internet based on the interests registered. The recommendation system provide users with the option to choose a context from the user profile and recommend articles collected from the web based in the interests associated with the context chosen.

Figure 4.1 presents the different mechanisms and demonstrate a dependency between them. Clients send interests and other contextual data (described in subsection 4.2.2) to a server that store the information. The mechanism sorting the interests based on contextual information collects the data needed from the first server's storage and perform the clustering and store the result. When clients request recommended articles the recommendation mechanism check the separation mechanism storage for a user profile and present articles based on the users' interests. The recommendation mechanism has a component that continuously collect articles from the internet so the recommendation mechanism has content to present.

4.2 Information gathering

This section cover the information that is collected by the system to create the user profiles, and how to collect it.

There is two types of information to gather. The first type is the user interests. This information give the system the ability to find content that is related to the users' interests. Without the interest information the best a user profile could do is to predict interests based on demographics and similar data. The interest information is discussed in detail in subsection 4.2.1 The second type is the contextual information for the user. To create a context-aware user profile the system need to have contextual information related to the users. The contextual information is discussed in detail in subsection 4.2.2. Subsection 4.2.3 cover how this information is gathered. Finally subsection 4.2.5 cover the data model for the information gathering.

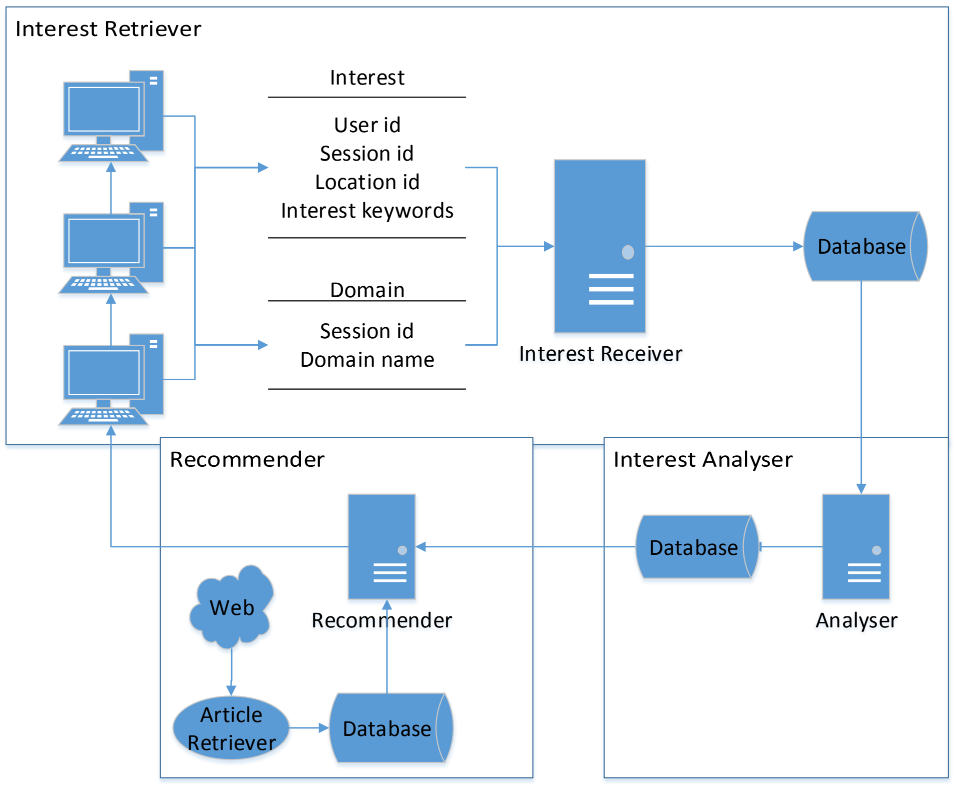


Figure 4.1: High level system architecture.

4.2.1 Interests

Users' interaction with something of interest on the web is often by clicking a link to access it. To illustrate, a user navigating a news website will click and navigate to the articles of interest by clicking on the headlines displayed. Similarly a user interested in content without knowing the location of the content will typically use a search engine. The results the user potentially click on after the search is finished will also contain keywords describing the interests, if it did not the user would not be inclined to click on the link. So in that sense it should be enough to collect interests just from links, and not from a query, since it would only be the same interest two times. If the client were to capture text from input element on websites the risk of the data captured being social security numbers, bank account numbers and personal information captured from chatting and social network services are high. More on this can be read under the subsection 4.2.4 Privacy.

Cleaning interests

The data collected through parsing link texts and search queries can contain a lot of unnecessary words that do not describe or indicate any interests. Trying to clean the text with Part-of-speech tagging (POS tagging) can be difficult, considering the input can be anything, not just properly constructed sentences. Another problem with the input is language. The interests are not exclusively in English and this result in an increased complexity since the POS tagging have to identify the language first, then process it if it support the language. Without supporting the language, the POS tagging would have a hard time identifying classes of words.

A simpler and more consistent solution is to use a technique called stop words removal. The downside of using stop words is the number of words not describing an interest, not being removed. There is also words that should be removed in certain situations and not in other. The most common problem here is Noun phrases where common stop words can be part of the Noun phrase. If a user is interested in The Who[1] and records the phrase "The Who", the list of stop words might contain both the word "the" and the word "who", so the entire name is removed. The only possibility is to add exceptions to certain stop word combinations with capitalisation, but this require the knowledge of these names in advance. This would in a lot of cases still be a problem using NLP instead, because successfully discovering unknown proper knowns is a very complex and hard task. Some margin of error is then expected when cleaning the interest keywords for unnecessary words.

Besides stop words removal it is good to also remove unwanted characters from the text with regular expressions (regex). This can for instance be used to remove all numbers and special characters, since they are most likely not of interest.

4.2.2 Contextual information

Context is captured either explicitly by requiring the user to specify it, or implicitly by monitoring the user and the user's activities[37]. To use the context in relation to the users' interests, there need to be an association between context and interests. This means that the context need to be cap-

[1]`http://thewho.com/`

tured for each interest[10]. Context can often be captured from multiple sources depending on the type of information. Personal assistant applications similar to Cortana[2] and Siri[3] might collect weather information from an external source, events from another external source and personal information like calendar events from the local device. Based on the infrastructure that is accessible to the current environment, different contextual information might be available and unavailable[10].

The basis for the contextual information in this thesis is the six high level categories from Schmidt et. al[37].

- User (e.g. age, habits)
- Social environment (e.g. social interactions)
- Tasks (e.g. active tasks, goals)
- Conditions (e.g. weather, light)
- Infrastructure (e.g. resources)
- Location (e.g. physical location)

Since the contextual information is collected on a per user basis, the user category is not relevant, since it would be identical each time it is collected. Social environment which is the second category, and cover social interaction and group dynamic among other things, is very difficult to collect indirectly. This information is typically not available without users' explicitly providing it. It could be available by possibly monitoring social network interactions, but this is beyond the scope for this work. The third category of contextual information is tasks. Tasks can be actions the users performs, goals the users try to achieve or users' intentions. Users' actions can be directly monitored since they are concrete, but not goals, as it is an abstract concept.

The last three categories are physical aspects of the context. Conditions, which can be weather, light conditions or other environmental variables, are difficult to collect indirectly. Weather can be collected from external weather services based on location, but this might not be that relevant since users will more than often be inside. If users were to be affected it would probably only be when there are extreme weather conditions. The time of day and day

[2]`http://www.windowscentral.com/cortana`

[3]`https://www.apple.com/ios/siri/`

of week are both conditions that can be easily collected simply by recording the current time. As for other conditions for the user, there is simply no way to collect information about the conditions in the room the user is located in without the user explicitly provide the information or without the use of sensors. Infrastructure is another contextual information category that is difficult to capture indirectly. Location is the last category for contextual information. It might be the absolute location or relative location. Absolute position might be represented by latitude and longitude values while relative location might be represented by a simple "is it the same location or not" or any other relational value. Relative location might be much more accurate in describing the location compared to absolute position, since two location (two computers) might be physically very close, but they have significant different uses.

The following subsections cover the contextual information recorded together with interests.

Time

A study by Karlson et al.[24] done on PC and smartphone usage for information workers show that work computers are primarily used in typical work hours. Time is then important for the indication of the user context. Most importantly is the time of day, since many user's context change based on what time of day it is. An example of this is a person that begins work at 8 am. This person's context will change at 8 am if the person is on time to work.

Another piece of information that is important for indication of user context is the day of week. It is still time, but on a slightly larger scale. What day it is in the week might determine if the user is at work or has a day off, or if certain activities take place for the user. These are tasks that might change the context considerably.

Physical location

For most users a change in location means that the user is at a new computer, possibly with a different context. A common scenario is users that use two different computers through out the day, where one is a computer at work, and the second is a personal computer at home. The physical location of the computers does not matter much. The important aspect here is the separation of the different contexts (e.g. work and personal) on two different computers. This is why it is reasonable to use a relative position for each computer the user uses and make the relative distance between locations either ”similar” or ”dissimilar”. The location can then be a unique machine id for each computer.

A study of device usage for industry and academia workers[14] show that separating work between a work computer and a personal computer was preferable for the subjects working in industry, but that they had trouble doing so in practice. The situation is then that not every user that have separate computers for work and private will have a clear distinction between them. Even though the difference in hardware can be a very good indication to the difference in context, it is not definite and exceptions may apply. The same way that a user may be doing personal tasks on a work computer.

Digital location

A user browsing the web will be visiting websites located at domains. This is a digital location the user visits, and is not the user’s physical location. A user going to a domain location is an abstract concept, since the user does not in any way visit the domain, but simply request content from the address. These domains often evolve around certain topics or activities, so users’ domains visited can contribute to the description of the users’ context.

4.2.3 Collection

There are three approaches for collecting user information, direct, semi-direct and indirect[19][36][28]. The direct approach would typically present the user with a questionnaire containing pre-defined interests the user can choose from. This is the simplest way to get the user’s interests, but not the most accurate, and certainly not the most convenient. The semi-direct approach can be conducted by allowing the users to rate the items they consume, and with this create a profile of what the user like and dislike. Letting the user provide a rating on items does, in general, work as intended and provide the system with data related to user interest. This solution does require user interaction, and it can be very complicated to recommend content if similar items are rated differently. Opposed to the direct and semi-direct approaches, the indirect approach does not require the user to actively provide the system with preference information. It will monitor the user activity and record consumed items as interests automatically. Recording consumed items as interests is not a fair representation of reality since the user can consume content and not enjoying it. This problem can be overcome by looking at the consummation frequency. If similar content is consumed by the user on a regular or continuous basis it is safe to assume that the user is interested in this content.

A flowchart of the information gathering process for the clients can be found in figure 4.2. The flowchart has two parallel paths, one for domains and one for interests.

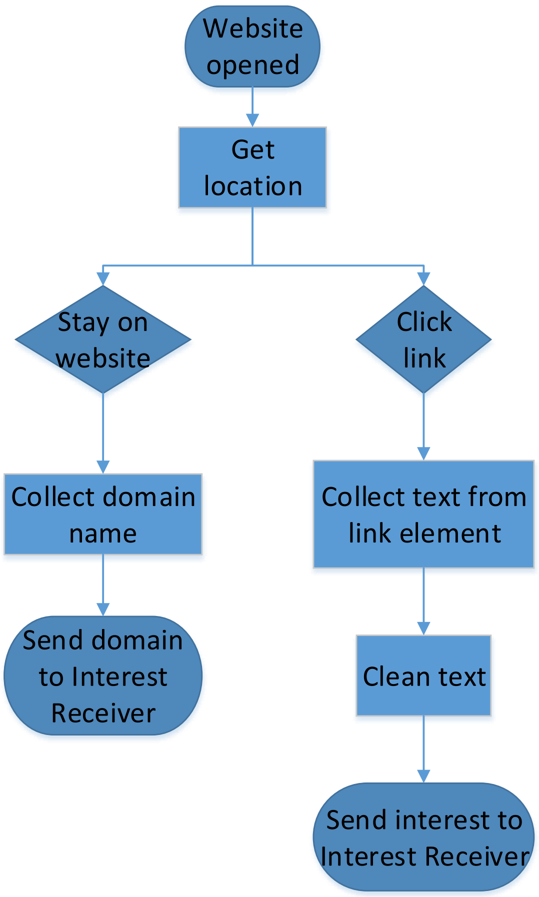

Website opened
Get location
Stay on website
Click link
Collect domain name
Collect text from link element
Send domain to Interest Receiver
Clean text
Send interest to Interest Receiver

Figure 4.2: Client information gathering flowchart.

As discussed in subsection 4.2.1, the interests will be captured as keywords from links clicked on by the users. Contextual information will be captured along with the interests so each interest have contextual information associated with it. The contextual information; time, physical location and digital location might not be enough to successfully reflect the different user contexts. To address this the contextual information will be collected in sessions, where sessions are simply a collection of interests collected in a small time period. The assumption is that more contextual data is better.

Session

A session is a collection of the information gathered in a limited time period. By gathering the information in distinct time periods there are more aspects to analyse then with single interests. The contextual information for a single interest is:

- Time
- Relative physical location
- Relative digital location

By collecting and analysing interests in sessions there are more information available to analyse. The information is given an extra dimension. A session will have a time frame, instead of a time point. This is represented by the session start and stop time points. The relative physical location will be the same for all interests in the session, and thus the same for the entire session. The relative digital location on the other hand will possibly be different for each interest, and this give a session a collection of relative digital locations, instead of a single relative digital location for a interest. The advantage of this is when two sessions are compared, the collections of relative digital locations can match in varying degrees (e.g. 50% match) while single relative digital locations will either be similar or dissimilar. Finally while analysing interests in sessions, the system can utilize the second dimension added, by analysing the series of time points for recorded interests. If a session contain ten interests, there is ten time points in the session. These time point series can be analysed as time series when comparing two sessions, to see to some degree if the usage pattern of the sessions are similar or dissimilar. The recording pattern will represent users' behaviour and be considered a part of the contextual information. The contextual information for a session is then as follows:

- Session start time
- Session stop time
- Relative physical location
- Multiple relative digital location
- Interest recording pattern

The drawback of capturing interests in sessions is the risk of collecting different contexts in the same session. This risk can be reduced by setting a time duration limit on sessions and not allow multiple physical locations in one session. This will limit the contextual differences between the interests in the session.

Sessions begin when an interest is captured and there exist no valid active session. A session timeout value is then set, so if the session is inactive for the duration of the timeout value, the session is retired. A timeout value for this can for example be one hour. The timeout value will limit the session duration to the amount of time a user is accessing the web, so when a user is done the session will timeout after a while. This can potentially be a long duration, if the user is continuously browsing the web. So a maximum time to live value is also set, so the session cannot be continuously kept alive for too long. Sessions should not be closed when a new session is created to accommodate situations where users use two computers at the same time or a virtual machine and is doing some browsing on each machine.

4.2.4 Privacy

This system will gather information about what users are clicking on, when they do it and what domains they visit. This is a major intrusion in a users' privacy, but an intrusion users implicitly agree to when they install the client recording the data and create a user account. For applications designed to collect user data, transparency in how and when it is done is very important to gain users' trust. It is also important that the software does not overstep the boundaries the user can expect the software to stay within. The source code for the client is available at Github[4] and the JavaScript will not be minified or uglyfied to hide the code from the user. Preferably there would be no manual oversight of the interest keywords captured, but without manual oversight there is no proper way to evaluate the success of the stage clustering the contexts described in section 4.3. While collecting data there are a lot of considerations to take, especially when it comes to privacy. There is a need to weigh the advantages of collecting certain data against the disadvantages. A simple approach would be to collect all keywords entered into input elements in the websites being monitored. This would result in the client collecting all the data and potential interests provided by a user. The problem with this approach is that is does not discriminate against websites like online banking, social networks and other websites that contains a lot of sensitive information.

4.2.5 Data modelling

The data model as can be seen in figure 4.3, is constructed based on logical separation of data. The session is the main entity in the data model. It is the core that combines the other entities. It combines the user with every interest recorded in a time series, as well as all domains recorded in the same time series. Each session contain several domains, and each domain can occur in several sessions, so there is a many-to-many relation between these entities through a session-domain mapping entity. Interests are recorded with possibly a collection of keywords. This is reflected with a one-to-many relation between an interest entity and interest keyword entities.

4.3 User profile creation

The core task in this project is to create user profiles with dynamically separated interests into different groups based on the context. The basic premise for this stage is to group the sessions with similar contextual information together and create a user profile from this data. This is only possible if there is a valid relation between similar contextual information and context. The premise for this is that there is a relation between the contextual information and the context, and by grouping sessions with similar contextual information, the sessions with similar context is also grouped. If a user's sessions

[4]`https://github.com/jtkarlsen/InterestsRecorder`

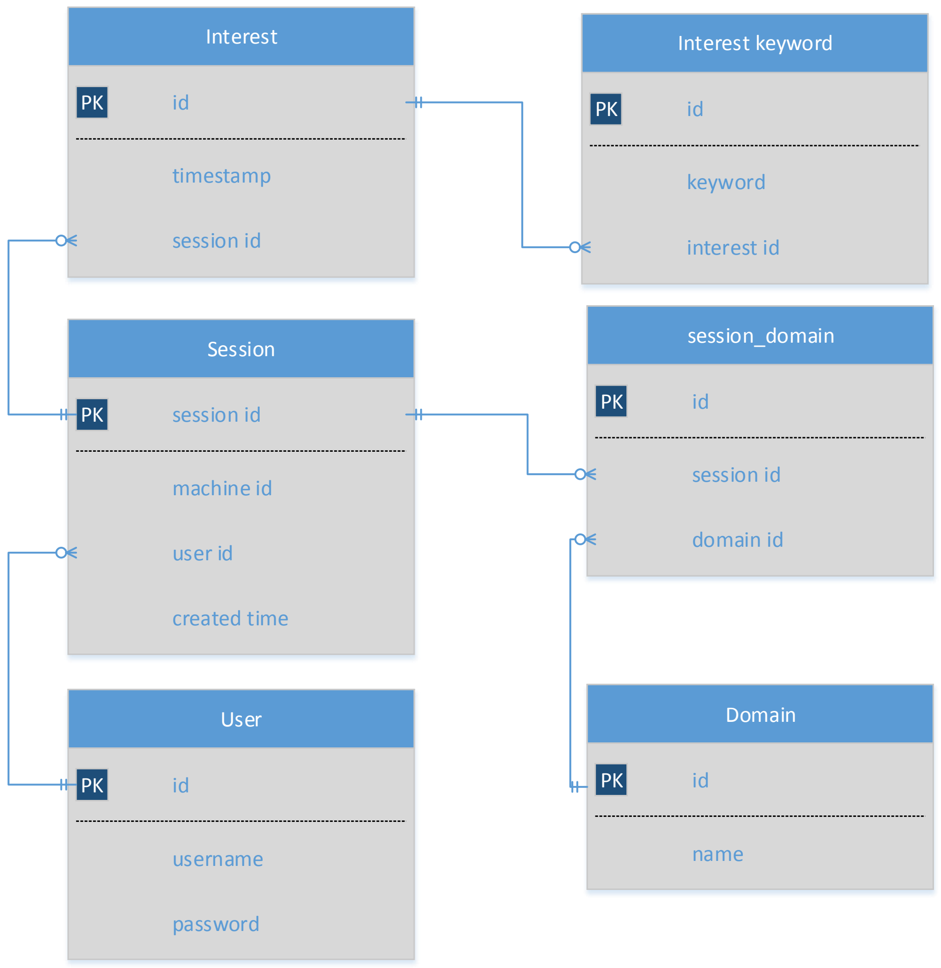
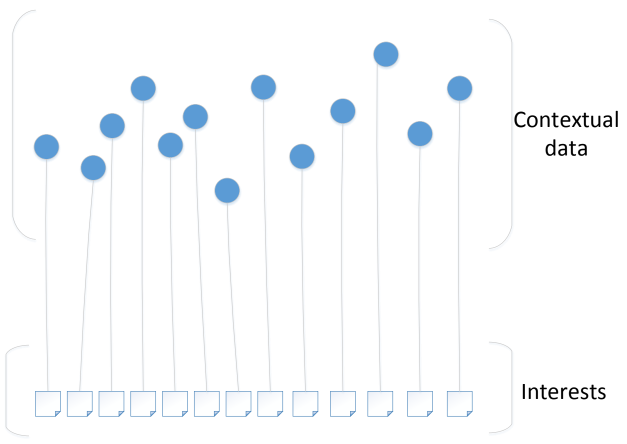
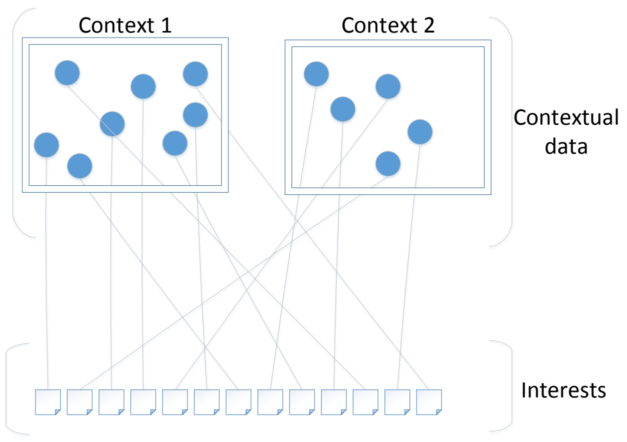


Figure 4.3: Data model as UML diagram.

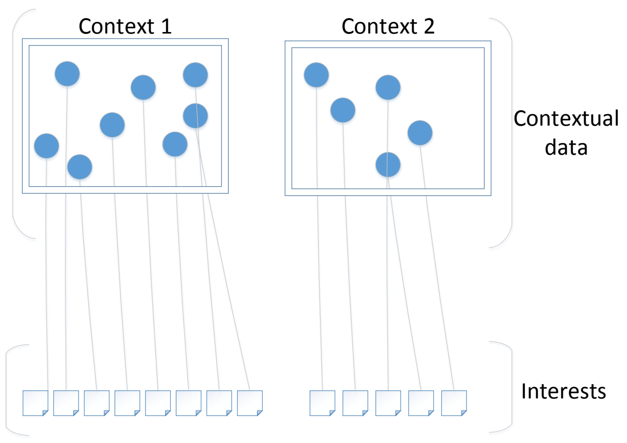
can be grouped in two groups, this means the user has two different contexts where interests have been recorded. When a user's sessions is grouped, the keywords representing the user's interests can be grouped based on the sessions, and the user profile can be created with the interest keywords sorted by contexts. This concept is illustrated in figure 4.4. Stage 1 (Subfigure a) illustrate the interests with the corresponding contextual data stored after collection, which is the initial stage for the process of creating the user profiles. Stage 2 illustrate the contextual data sorted in groups based on their similarity. The groups represent two different contexts for the user. Stage 3 is the completed stage where the interests are sorted by the contexts.



(a) Stage 1 - Initial state



(b) Stage 2 - Contextual data sorted by similarity



(c) Stage 3 - Interests sorted by context

Figure 4.4: General approach for creating user profiles that reflect context

A very popular technique for detecting similar data in a large dataset is cluster analysis. You can read more about the basics of cluster analysis in section 2.6. The technique locate similarities or patterns in a large dataset and group them together based on this. Clustering is a type of classification, and is often called unsupervised classification[38]. This will allow the system to detect the different contexts in the dataset dynamically. With simpler methods such as supervised classification, where data is sorted in pre-defined classes, it is impossible to discover new or undefined groups[38]. Clustering can be a subset of unsupervised machine learning. In clustering, similarity is perhaps the most difficult step to overcome. The next subsection 4.3.1 cover how similarity is defined and calculated for this project. The clustering algorithm cannot perform clustering on data without knowing the relation between the items in the dataset. The task of differentiating contexts then comes down to two steps. (1) Calculate the distance or similarity of the sessions based on contextual information. (2) Apply a clustering algorithm on the dataset with the distance from the first step. The clustering algorithm will then sort the dataset into clusters or groups based on the parameters provided. Clustering is covered in subsection 4.3.2.

4.3.1 Distance

For cluster analysis on a dataset to work, each data in the dataset need to have a defined metric separating them in distance or dissimilarity[22]. The distance measure should be chosen based on the attribute type of the data in the dataset, since different distance measures will be affected differently by attribute types. For Euclidean and Manhattan distance the attribute types should be normalised quantitative values[22][38]. In other words the values can be continuous, discrete or interval, as long as the type is consistent. A mixture of attribute types will result in unintentional weighting of certain attributes. Jaccard distance is a measure suited for attribute types that are binary or nominal[38]. Cosine similarity is a measure suited for attributes of quantitative types, but with a lot of attributes and thus with a lot of dimensions in vector space[38]. To identify the most suited distance measure, the data attributes of the data recorded in this project need to be examined.

The data objects to perform clustering on is the sessions recorded from user activities. These objects have a number of attributes that need to be taken in to account. From table 4.1 the attributes and their type is listed. There are two different attribute types in the data object and none of the distance

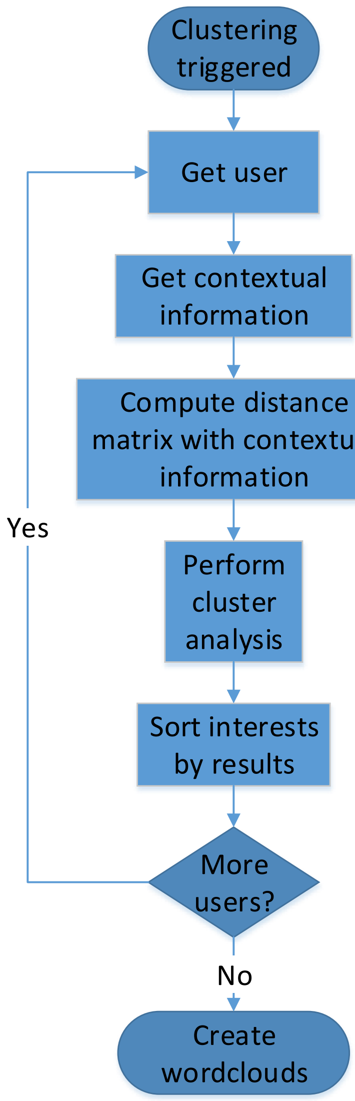

Clustering triggered
Get user
Get contextual information
Compute distance matrix with contextual information
Perform cluster analysis
Sort interests by results
More users?
Yes
No
Create wordclouds

Figure 4.5: User profile creation flowchart.

measures covered will work on the data as it is. The attributes correspond poorly with the attribute requirements for Jaccard distance, so that distance can be ruled out since it is not very meaningful to convert discrete or continuous data to nominal or binary values. The objects that shall be clustered has six attributes and does not qualify to be categorized as high dimensional, so the obvious advantages of cosine similarity measurement does not apply to this dataset[4]. High dimensional data usually have from dozens of attributes to several thousands[4]. Three of the attributes are nominal values, that makes the Euclidean and Manhattan distance also not usable without normalising the data. The book "Finding Groups in Data: An Introduction to Cluster Analysis"[25] by Kaufman et. al. says "The use of the Manhattan distance is advised in those situations where for example a difference of 1 in the first variable, and of 3 in the second variable is the same as a difference of 2 in the first variable and of 2 in the second.". This is not the case in our clustering analysis, since a comparison of a session starting at 10.00 and ending at 16.00 and a session starting at 14.00 and ending at 18.00 is not the same as a comparison of a session starting at 15.00 and ending at 17.00 and a session starting at 16.00 and ending at 18.00. Euclidean distance is the best fitted distance measure for the data, since converting nominal attributes to discrete attribute does work to a degree depending on the data.

Attribute	**Type**
Physical location	Nominal
Time of day start	Discrete
Time of day stop	Discrete
Day of week	Discrete
Domains	Nominal
Recording pattern	Nominal

Table 4.1: Session object attributes with type.

Based on the attribute types the session object have, three nominal and three discrete, the obvious choice is to convert the nominal attributes to discrete and try to normalise the values in a reasonable manner. Some attributes will matter more than others for the clustering to be as accurate as possible, so the values need to be weighted after importance as well. The values in the session object need to be broken down and converted to normalised discrete values. These values can be in the range of 0 to 100, without considering weighting.

After all the session's attributes have been normalised, they are all discrete values in the range from 0 to 100. This is very good values for the euclidean distance measure to calculate the distance between the different sessions. The different attributes for the session objects are all in the range from 0 to 100, and have with this the same effect on the distance results. In reality the different attributes have a different impact on the identification of the context. This is where weighting come in, and this will be discussed in subsection 4.3.1.

The following subsections cover the measurement of similarity between the different attributes associated with sessions and discuss the weighting of the distances.

Physical location

Physical location or machine id is a unique nominal value representing a specific machine. When comparing two machine ids the outcome is binary, it is either equal or unequal. It is either the same machine or not. The physical distance between machines does not matter as it would be impossible to use it to analyse the contexts. Since the result of comparing two machine ids is binary, it can be normalised to be either the minimum value or the maximum value. In our case this results in the value 0 for equal values and the value 100 for unequal values.

Time of day

Time of day start and Time of day stop are discrete values in the range of 0 to 24 hours. They are both in the same scale and in the same range so we can cover them together. When comparing the time of day values, the resulting difference can be in the same range as the values them self, 0 to 24. All we need to do for these discrete values is to change the range from 0 to 24 to 0 to 100, and this is easily done by dividing the resulting value with 24 and multiply with 100. For example the time 14.33 is 873 minutes, and the time 16.47 is 1007 minutes. In minutes the range is from 0 to 1440. The difference between the times are 134 (1007 - 873) in the range of 0 to 1440. In the range of 0 to 100 the difference is 134 divided by 1440, multiplied by 100, so 93. The tricky part is to take into account the circular nature of time. 00.05 and 23.55 can be either 23 hours and 50 minutes apart, or 10 minutes. Since the clustering algorithm is comparing session from different days there is no basis for using the date of the time to solve this. The best solution is to use the shortest distance, since it is the real distance in time between the values we are comparing, without considering the date.

Day of week

Day of week is also a discrete value in a different range than the proposed 0 to 100. The approach used for normalising the time of day start and stop applies to day of week as well. We need to change the range from 0 to 6 (zero indexed 7 day week), to 0 to 100. The approach is then to divide the difference between the day of week with 6 and multiply with 100 and the result is normalised. We need to remember that the day of week also is a circular number as the time of day is, and need to be handled accordingly.

Domains

Domains are a list of nominal values that are collected throughout the session. Since the domains are a list of nominal values, we can use the Jaccard distance to calculate the similarity. The Jaccard distance is based in the rate of overlapping items in two datasets and will suit this data. Jaccard distance is the difference of the sizes of the union and the intersection of two sets divided by the size of the union. When measuring the distance of the two set of domains that are listed in table 4.2 by using Jaccard distance, we get 0,875. We get this value by first figuring out the union of the two datasets, which are 16, and then the intersection, that is 2 (en.wikipedia.org and www.amazon.co.uk). Applying these numbers to the Jaccard distance formula give us $\frac{16-2}{16} = 0,875$. The distance returned from the formula is in the 0 to 1 range, so by multiplying with 100 we get the desired range.

Domains A	**Domains B**
elinux.org	en.wikipedia.org
www.amazon.co.uk	www.investopedia.com
www.sdcard.org	www.mathisfun.com
mega.co.nz	cs.bu.edu
support.google.com	stat.ethz.ch
kodi.wiki	cs229.standford.edu
www.raspberryp.com	mathoverflow.net
www.modmypi.com	www.amazon.co.uk
meta.stackexchange.com	
en.wikipedia.org	

Table 4.2: Two domain datasets.

Interest recording pattern

Recording pattern is a list of time values that represent the time each interest in the session was recorded. The list of time values or pattern is a nominal value seen as a single attribute, but each item in the list is a discrete number type. Time can be seen as either a continuous value or a discrete value based on the application. For this project a place in time will be viewed as a discrete number value, since it in computer science often is represented by a whole number. There are several different approaches to calculating the distance between two set of numbers. We can look at the sequence as a vector with each value as an attribute and measure the similarity between them with euclidean distance. This will result in a similarity measure that does not consider patterns in the sequence, but just compares the vector as an average combination of attributes. We can apply the edit distance algorithm to calculate the similarity by finding the number of edits required for the sequences to be equal. This is an approach that is more suited for words and unordered sets of numbers, and it will not take into account the frequency of the sequence. For two sequences [1, 2, 3, 4, 5] and [1, 3, 4, 5] the edit distance would at index 2 change the number and add one to the distance, and this will continue for every number since one of the sequences is lagging behind with one value. The distance measured would be 3 edits out of 4 items, so a large distance, while the sequences are very similar with only one number missing in one set. The last technique is to look at the sequence as a time series and to calculate the similarity between them with dynamic time warping (DTW). DTW adjusts for local time shifting and sequences out of phase when measuring the distance. This is good for the accuracy of the distance measure since the patterns recorded is in no way guaranteed to be with the same frequency and without acceleration or deceleration. The sequences are also not guaranteed to be synchronized and the DTW distance measure adjusts for this as well. DTW is explained further in subsection 4.3.1. The resulting value from the DTW distance measure is a continuous value type and does not fit in to the 0 to 100 range. This can be handled by setting a maximum distance and converting all values over this to 100, the maximum value, and normalising the rest in the range of 0 to 100.

Attribute weighting

The approach for setting the weight values was first to evaluate the importance of the attributes against each other. This is to have some initial values as a starting point. After some initial values have been selected, they are changed based on a trial and error approach to see what weighting values provide good results. I have yet to locate a measure that will successfully measure what value weight provide good results without having a pre-defined set of classes or results to compare to. Because of this a trial and error approach to determine the weighting is considered the best approach. Good results are relative, and different session data will react differently to different weight values, so the weight values are difficult to evaluate. The values used to weight the attribute distances when determining session distance is presented in table 4.3. This is the result of preliminary testing with trial and error based on initial values.

For the initial values *time of day start* and *domains* was set higher than the other attributes, because they were considered more important. The preliminary testing with the trial and error alteration of the weighting resulted in better results than the initial values when the other attributes were reduced further. So as a result *time of day start* and *domains* are very dominant attributes, while the others and especially *day of the week* is low.

Attribute	**Weight**
Physical location	2
Time of day start	6
Time of day stop	2
Day of week	1
Domains	4
Recording pattern	2

Table 4.3: Session attribute weighting

4.3.2 Clustering

This work will use cluster analysis to group similar sessions together and to keep dissimilar sessions apart. The clusters will represent the users' different contexts in the user profile. The idea behind this is that the different contexts, also have different contextual information associated with them. Then by clustering the interests with similar contextual information, the interests with similar context is clustered. It is also important to limit the number of contexts to a reasonable number while still representing the users' actual contexts. This is to reduce the complexity of the user profile. If the contexts were to be too precise, the user profile would have a context for every single action the user performed.

Clustering algorithms

Based on the euclidean distance metric used to measure distance between the sessions, we can choose between several clustering algorithms. Some of the most popular clustering algorithms are hierarchical clustering, k-means clustering and DBSCAN. These three clustering algorithms cluster the datasets in distinct different manners, and we will need to choose the one that fits our data best.

Hierarchical clustering is as the name indicates a technique that cluster the data in a hierarchy of clusters. Hierarchical clustering is explained further in section 2.6. The results from hierarchical clustering will vary depending on how "the closest pair of clusters" is defined. It can be defined by the closest points, furthest points or mean of all points in the clusters. To use the results from hierarchical clustering we need a notion of how many clusters we want, or a threshold for a maximum distance between clusters, or else the end result for the agglomerative clustering is one cluster. With a distance threshold the clustering algorithm can stop combining clusters after the closest clusters are further away than the threshold distance. With hierarchical clustering we can get robust clusters that are formed up to the threshold distance. This threshold is the maximum distance before clusters no longer are deemed similar, and no longer merged. By ignoring the clusters where there are too few members we can also ignore outliers from the results of this algorithm.

K-means clustering is a technique where the user specifies the number of clusters and the data is sorted into the closest cluster based on the centroid. The centroid is the mean position of all points in a cluster. K-means clustering is explained further in section 2.6. Since k-means divide all data items into the closest cluster, the algorithm has a hard time discovering shapes in the dataset.

DBSCAN is a density based algorithm that create clusters where there is enough data items in a given area. The user has to specify the range each item has to search for neighbours and how many items it is needed to create a cluster. DBSCAN is explained further in section 2.6. DBSCAN can be very sensitive to the parameters chosen and tiny changes in them can on some datasets result in massively different results. Furthermore the algorithm has a weakness for varying densities in the dataset, since the requirements for a cluster is based on hard values. The algorithm is on the other hand very robust in regards to outliers and noise and is one of the best algorithms for discovering arbitrary shapes.

All of the above mentioned clustering algorithms should perform sufficiently for the task of clustering the contextual information. All with their advantages and disadvantages. The data to perform cluster analysis on is generated by user actions, and has no clear structure. For different users the data can take completely different shapes, based on the users' usage. This means we cannot make any assumptions about the shapes or the density of the dataset. Since DBSCAN has a weakness for varying densities the results may vary greatly depending on the dataset, so DBSCAN might be the least feasible option. Between k-means and hierarchical clustering there is a significant difference in the result structure. K-means provide flat clusters while hierarchical provide a hierarchy of clusters. This hierarchy can easily be used to explore the accuracy of the clustering by increasing the depth. Based on this the choice of clustering algorithm fall on hierarchical clustering.

The next subsection discuss the linkage to use for the hierarchical clustering.

Linkage

Single-linkage hierarchical clustering risk making the entire dataset a cluster if the items in the dataset are close enough. Since single-linkage is measuring the distance between the closest items in the clusters, there is a risk of there always being a point in range of one of the outer points and in the end the entire dataset is one large cluster. There is no limitation to the size of the dataset, so this is a potential problem. For complete linkage hierarchical clustering the distance between the clusters is measured from the points furthest away. It will force clusters to be spherical in shape as long as there is enough data and it will force the clusters to have a maximum diameter equal to the threshold value if the threshold is based on distance. For average linkage hierarchical clustering the average distance between all items from one cluster to another is the distance. There is also a modified average linkage called weighted average linkage that is similar to average linkage except that the distance is weighted based on the cluster size. Average linkage is a middle ground between single-linkage and complete linkage hierarchical clustering. It can be computational intensive, but the result can potentially be significant better. It also is less sensitive to outliers in clusters, since the distance will not be based on single items on the edge, which is a major drawback for single-linkage and complete-linkage.

It can be hard to decide what linkage to use for hierarchical clustering but considering the shortcomings of single-linkage and complete linkage clustering for an unknown dataset, the best looking alternative is average linkage. Average linkage hierarchical clustering will simply put produce relatively compact clusters that are relatively far apart. Preliminary testing supports this assumption as well.

Number of clusters

Determining the number of clusters is done with the elbow method. This method consist of locating when the effectiveness of adding new clusters drop[26]. Optimally the drop in effectiveness will be significant at some point. This elbow point cannot always be unambiguously identified[26], but the best point will still be chosen. By using hierarchical clustering the hierarchy of clusters can be used to determine the increase in efficiency in a single calculation.

4.3.3 User profile modelling

A user profile is simply put a collection of relevant data to perform the desired actions related to the user. A user profile can be a metaphorical object that includes different user data from different parts of the system. The user profile for this system represent users' interests in regard to the different contexts they might have. After performing the cluster analysis covered in subsection 4.3.2 the user profile can be visualised as in figure 4.6. The user profile will contain a set of contexts that the sessions are sorted into. The sessions in the contexts each contain one or more interests and the interests contain one or more keywords describing them.

The user profile is dynamic and will change for each time the cluster analysis is performed with new data available. The sessions, interests and keywords will all stay static, with the exception of adding new ones. The aspect of the user profile that will change is the contexts. The user profile will be constructed on the backend of the system because of the computational requirements to perform the cluster analysis and the consistency it provides for multiple clients.

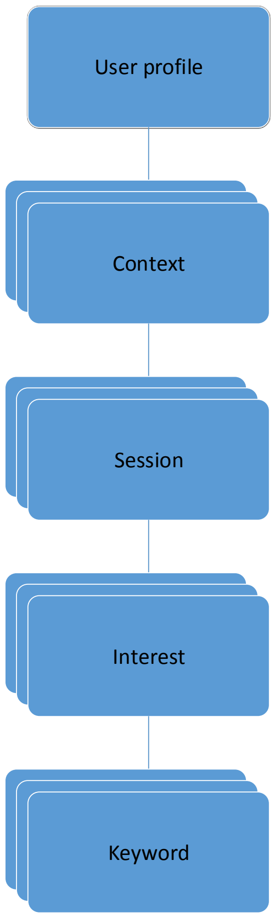


Figure 4.6: Overview of the user profile content.

Boosting and importance

If a specific keyword is recorded multiple times, the interest it represent is most likely of higher importance to the user than an interest with a keyword that only occur once. For a recommendation system to reflect this, a term weighting scheme need to be added. The simplest way of solving this is to multiply the keyword boost value by the number of occurrences for the keyword. A keyword that occurs twice is with this logic twice as important as a keyword that occur once. This solution does have a significant problem. Interests the users has had over a long period of time will potentially have a lot of keywords with multiple occurrences collected over time. Newer interests will not have the same amount of keyword occurrences for most likely a long time and will be seen as not very important in the user profile. Users also might have a lot of keywords for topics that they no longer find interesting, but the amount of keywords will be stable on the current level and it will be counted as important until the other interests surpasses it at a later time. This is not a good way to handle recommendations. To solve this, keywords will be associated with the time they were recorded by the system. The importance of the keyword will be decreasing by time passing since it was recorded. This will result in old keywords retiring with time and they will make room for new and possibly more important keywords. The age of the keyword will be used to decrease the boost value to create a more accurate representation of user's interests at the time of use.

4.4 Recommendation

To test the user profiles created, a recommendation system is created. The system will use the users' user profiles to provide relevant articles from selected sources on the web. The system will provide the option for users to select what context in the user profile the recommendations will be based on. Since users' interests are gathered based on links clicked, the users' interests should correspond well with articles from websites. A user's taste in movies might not have been such a good match since there is not necessarily a relation between interests from links clicked and movie interests. So with that assumption, recommending articles based on what users' click on the internet should work well, since users often click on links to read articles or other information. The recommendation system can be divided into three parts; collection, recommendation and presentation. These parts will be covered in detail in the following subsections.

4.4.1 Collection

For the system to be able to recommend articles to the users, it need to first have access to articles. The articles need to be available to search through to find matching keywords if they are to be used for recommendations. For performance reasons this require they are stored locally and indexed, or that a service taking a large set of keywords with weighting is available. There are several approaches to collecting and storing articles from the web. An obvious option is to crawl the web for articles and store the information needed. This approach either requires a predefined set of websites to crawl, or a crawler collecting articles from all over the web. A crawler can be a considerate amount of work to produce, but luckily there are several free and open source options available. Some alternatives include Nutch[5] and Scrappy[6]. Crawling the web for articles can be a complicated task, and the results varying depending on implementation. Another option to this is to use widely supported web feed formats to collect well formatted articles from multiple sources in a homogeneous way. The number of potential sources will rely on the number of websites that use and support a web feed standard to provide articles. This is not a big issue since the recommendation system is only to demonstrate the results of the user profile created. Another drawback from using web feed standards is that the content usually is not the entire article, but a headline and a summary. This should however be more than enough to query for the articles based on the user profile. Following web feed standards to collect articles is the simplest and perhaps best option in regard to implementation and our requirements for this system.

A web feeds is, simply speaking, websites that follow and present their content by a standard so feed readers can consume content from different sources the same way. This means that to collect articles from web feeds the system need a list of sources that can be checked for new content periodically. The system need a mechanism to check if the content has been consumed or collected previously to prevent duplicate data. A component that periodically check sources with a web feed standard and consume them based on the standard they are presented in should be enough for collecting articles for the recommendation system. This component is illustrated by a flowchart in figure 4.7.

As a source for the articles collected, Google News should be an excellent option. Google news already apply advanced mechanisms for providing a large selection of the most popular news articles[7]. Another great reason to choose Google News as a source is they collapse similar articles into a single article, so the system will have one article instead of possibly a lot of articles about the same situation from dozens of different sources. Because the alternative to Google News would be to manually compile a list of sources, and this could be skewed in favour of sources the author personally prefer.

[5]`https://nutch.apache.org/`
[6]`http://scrapy.org/`
[7]`https://www.google.com/intl/en_us/about_google_news.html`

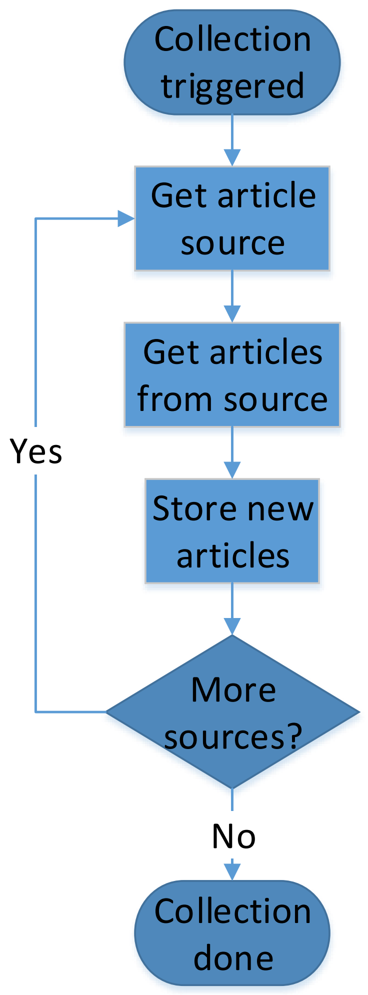


Figure 4.7: Article collection flowchart.

4.4.2 Storage and recommendation

To recommend articles to users based on their user profile containing keywords representing their interests, we need to be able to query the articles for matches with the said keywords. We need to store the articles in such a manner that they can be queried and preferably indexed beforehand. The best possible scenario is that we can query with a large selection of keywords and their weight and get a list of articles sorted by the degree of match to the keywords and weight. It is important that the solution support TF-IDF or similar methods for searching, so common words do not get overrepresented.

A good solution to solve the storage and recommendation aspect for the recommendation system is to use a full text search engine with support for term weighting. This would result in minimal implementation and a quality verified by many existing users. Figure 4.8 is a flowchart illustrating the recommendation process.

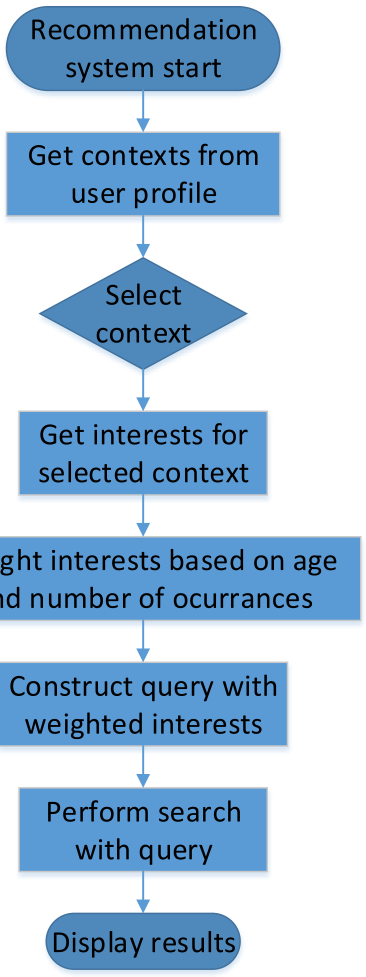

Recommendation system start
Get contexts from user profile
Select context
Get interests for selected context
Weight interests based on age and number of ocurrances
Construct query with weighted interests
Perform search with query
Display results

Figure 4.8: Recommendation flowchart.

4.4.3 Presentation

The goal for the recommendation system is to test and demonstrate the user profiles generated. Because of this, the articles recommended to users should be presented in a clean and structured matter so it is easy to locate the most recommended content and evaluate the user profile. Since the user profile is separated in several sections based on the contexts, the recommendation system must provide an option for users to select what context to get recommendations based on. Without this option it will be impossible for users of the recommendation system to assist in evaluating the correctness of the different clusters. There is no way for the system to determine *what* the different contexts are, just that they are different, so they will by default be without a descriptive label. They will need to be described by some available information like the most prominent keyword(s) in them. With a successful stop words removal the most prominent keywords should be descriptive of the content.

Since the different contexts are generated by contextual information there is no labels or key identifiers on the different contexts. The contexts will be presented to the user as a word cloud. A word cloud will in a good way illustrate the most prominent words in the group, and this will make it possible to identify them. Word clouds in itself are a very good tool to judge the quality of the groups since it make it easy to see if the prominent terms are somewhat related.

Chapter 5

Implementation

This chapter cover the architecture and implementation of the software for this thesis. Figure 4.1, on page 29, illustrate the different components. The software consist of five separate components, the Interest Retriever, Interest Receiver, Interest Analyser, Article Retriever and the Recommender. They are described in section 5.2, 5.3, 5.4 and 5.5 respectively. The Article Retriever and Recommender are both covered in section 5.5.

5.1 Server

The server hosting the solution is running on an Intel(R) Core(TM) i5-650 (3.2 GHz) processor with 4 GB of DDR3 memory and a 200 GB 7200RPM hard disk drive. The server is using Microsoft Windows Server 2012 64-bit as the operating system. The hardware and software is not chosen by any other reason than that it was already available for use and sufficient. At the time of testing the server had no other tasks than hosting this system.

5.2 Interest Retriever

The Interest Retriever is a Google Chrome browser extension[1] that collect interests and contextual information from users' link clicks in the web browser.

[1] `https://developer.chrome.com/extensions`

It is primarily written in JavaScript with a small amount of HTML. The choice of exclusively creating an extension for Google Chrome is based the browser usage of the potential application testers. I found Google Chrome to be the most used browser by a landslide. The browser extension work by injecting a script into every website that is rendered by the browser. The script add an event listener to the body of the website that check whether the tag clicked is a link or not. If the target is a link the text is read and an chrome.runtime.message[2] event is activated, sending the content of the target to a background script in the extension. The injected script will also send the current domain to the background script after ten seconds has passed. The ten second limit is to exclude domains that are visited briefly, since it is a high chance the user did not intend to visit it or saw that it was not of interest.

The background script evaluate the text from the link clicked and remove stop words. The script remove any individual numbers and special characters from the text. The background script store stop words from ranks.nl for both English[3] and Norwegian[4]. If the content is not empty after removing stop words, the interest is sent to the backend with the machine id, session id and user identification. The machine id is created only once and is a practically unique UUID (128-bit number). When it is created it is stored in the extensions local storage. For each time the machine id is needed, it is retrieved from the storage. If there is no machine id in the storage (e.g. first time use or storage deleted), the background script will generate a new UUID. This ensures the same physical machine has the same relative location in the system, unless the storage is deleted. The session id is similarly with the machine id a practically unique id that represent the current session.

Algorithm 4 Session timeout

```
1: Session start
2: repeat
3:     Interest recorded
4:     Session timeout value updated
5: until Session has timed out or max time to live reached
```

A new session (session id) is set when an interest is recorded and there is no existing session, or the existing session is expired. Each time an interest is submitted to the backend, the current session will refresh it's timeout value

[2]https://developer.chrome.com/extensions/messaging

[3]http://www.ranks.nl/stopwords

[4]http://www.ranks.nl/stopwords/norwegian

as shown in algorithm 4. This value will be checked when the next interest is submitted to see if the session has timed out. If the session is timed out a new session will start, but if not the current session will be used and the timeout value will be overwritten. Sessions also have a maximum time to live, so if this value is exceeded, a new session will begin even if the timeout value is still valid.

The background script send the domain received from the injected script to the backend. If there currently is no valid session, a new session will be created. This is to ensure the domain will be in the session with the interests that will potentially be collected from the website.

The user is authenticated through the Interest Retriever by entering the username and password in a small pop-up window available for extensions in Google Chrome. The credentials is verified with the backend, and when authenticated the Interest Retriever will submit interests recorded. Without an authenticated user the Interest Receiver will not submit anything to the backend. The small pop-up window mentioned is hidden behind an icon[5] licensed under creative commons[6]. This icon will change to display an additional red exclamation mark when the user is not authenticated. When the user is authenticated the username will be stored using chrome.storage.sync[7]. This ensure the user will be authenticated automatically on all Google Chrome browsers the user is logged in with a Google account.

The Recommender website is constantly listening for an event emitted by the browser extension with the user identification, and thus use the Interest Receiver for user authentication. This is a practical solution that allow users to log in only once for two separate applications.

5.3 Interest Receiver

The Interest Receiver is a backend component that provide a rest API to the Interest Retriever Google Chrome extension. It receive and store the information collected by the Interest Retriever. The component is developed in Python 2.7 with the Flask[8] package. Besides the Flask package the com-

[5]http://www.flaticon.com/free-icon/eye-close-up_61916
[6]http://creativecommons.org/licenses/by/3.0/
[7]https://developer.chrome.com/extensions/storage
[8]http://flask.pocoo.org/

ponent also relies on the MySQLdb[9] package for connection to the database and the json[10] package to easily interpret the data sent from the browser extension. The hashlib[11] package is used for a sha1 hash of user passwords in the database. The component has very basic storage requirements, so to use a MySQL database is simple, and provide the functionality the component need. The entity-relationship model for the database can be found in figure 5.1. The Interest Receiver provide an API with an endpoint for submitting interests, submitting domains, authenticating users, and creating new users.

- /interest - Post serialized interest object
- /domain - Post serialized domain object
- /user - Post serialized user credentials
- /user/new - Post serialized user credentials

These are the functionalities the Interest Receiver component provide. Interests received will create a new session in the database if there is none with the id already existing. The Interest Receiver will store interest keywords in many-to-one relation since keywords can be very diverse and while duplication will happen it is not very significant. For applications that are more than a proof of concept the interest keywords should be stores with a many-to-many relation to save storage. Domains received is on the other hand is a many-to-many relation since domains are often similar and there will be a lot of duplicates.

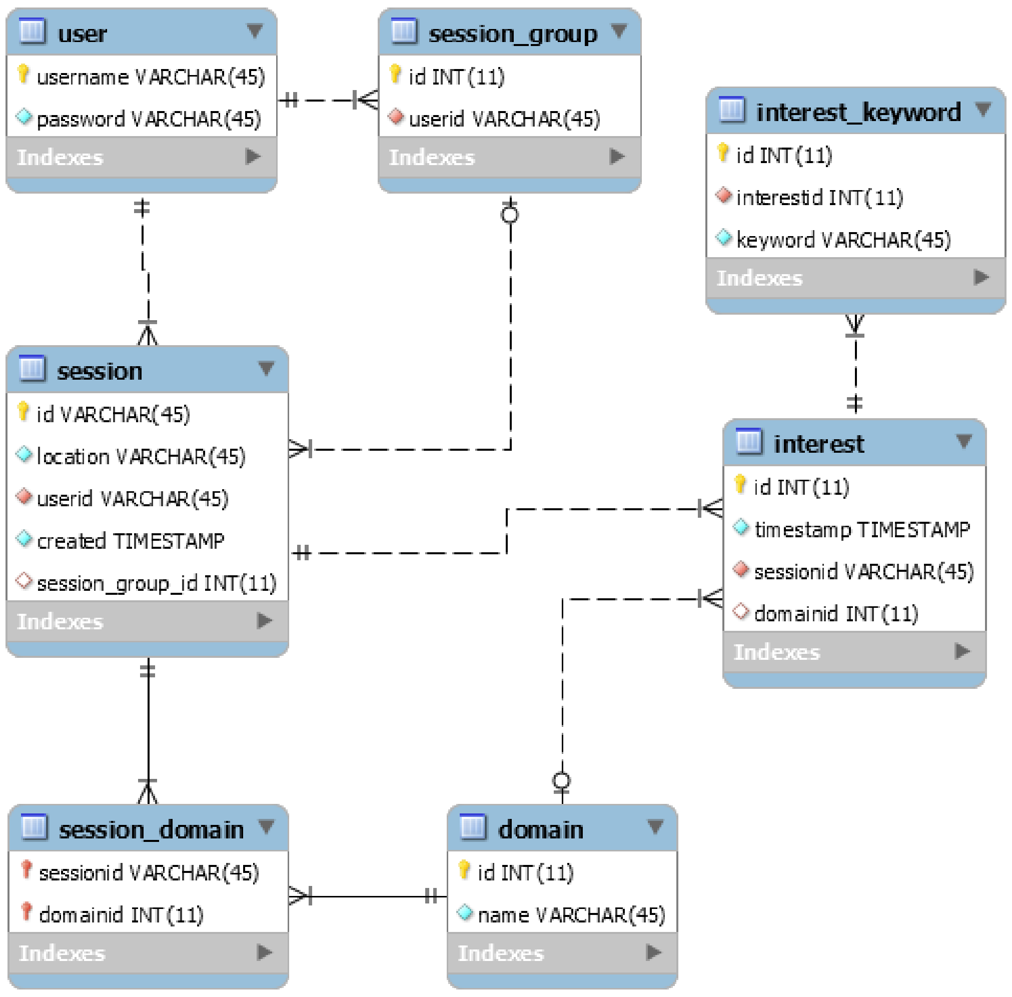
Sessions are a possession for users, and the database reflect this. Each session is associated with a user, and the user identity is verified for each request. Users' passwords are hashed with sha1 before storage for security and privacy concerns.

5.4 Interest Analyser

The Interest Analyser is a Python application that use cluster analysis to sort similar sessions into groups for each user. A group represent a context

[9]`http://mysql-python.sourceforge.net/MySQLdb.html`
[10]`https://docs.python.org/2/library/json.html`
[11]`https://docs.python.org/2/library/hashlib.html`

Figure 5.1: Database entity-relationship model.



for the user. The Interest Analyser also create word clouds based on the interests in each context when the cluster analysis is done. The application run in an interval of one hour, to ensure reasonable fresh contexts to use in the Recommender system (section 5.5).

The first thing the application does is to get all users from the MySQL database, since the clustering will be performed on the sessions for each user. For each user, the application collect the user's sessions. If the user has more than ten sessions, the system will perform cluster analysis on them, if not it will do nothing and go to the next user. If there is fewer than ten sessions it is not enough data to get any meaningful results. The distance measure, cluster analysis and the word clouds is covered in the following subsections.

5.4.1 Distance

The cluster analysis require a distance measure stored as a distance matrix. The distance matrix is created by looping over the sessions with two loops, one for each session, and another to calculate the distance between it and every other session. The distance measure between each session is performed by calculating the distance between each session attribute and summarizing it as the distance. The sessions have six attributes used to describe the context the session is captured in. These are presented along with their data type in table 5.1.

Attribute	**Data type**
Physical location	String
Time of day start	Timestamp
Time of day stop	Timestamp
Day of week	Timestamp
Digital locations	String array
Interest recording pattern	Timestamp array

Table 5.1: Session attributes

All distances calculated must be normalized so they have the same impact on the distance between the sessions compared, before they are weighted. If not they will be unintentionally weighted, instead of intentionally weighted and the results will be unpredictable. The distance range the results must fall within can be any value, since it is relative, so we set it to be 0 to 100. The weighting scale used in this application is from 1 to 10, where 1 is no boost in importance and 10 is a significant boost. All distance values are multiplied with their weight value discussed on subsection 4.3.1.

Relative location

The relative location described with a unique machine id can either be similar or dissimilar. The distance measure between two machine ids is then straight forward. The distance is 0 if they are equals, and 100 if they are not.

Time

Both time of day start and stop use the same distance measuring function. The function convert the Timestamp to seconds since midnight that day. If both times are equal, the function return 0 since they have no distance. The function subtract the smallest of the times from the largest, to get the difference in milliseconds between them. The returned distance is then the distance in milliseconds between the times divided by the total amount of milliseconds in a day multiplied by 100. To escape the problem of the distance not taking into account the circular nature of time, in other words that 23:59 is 120 seconds from 00:01, the function check if the distance is higher than half the day. If this is the case, the new distance must then be the distance subtracted from the total seconds of a day (86400 seconds) before normalizing it in the range from 0 to 100. Algorithm 5 illustrate the algorithm used to determine the distance between two time points just described.

Algorithm 5 Algorithm for distance between two time points

```
1: Convert t1 and t2 to seconds since midnight
2: if $t1 > t2$ then
3:     $distance = t1 - t2$
4: else if $t2 > t1$ then
5:     $distance = t2 - t1$
6: else
7:     return 0
8: if $distance > 86400/2$ then
9:     $distance = 86400 - distance$
   return $(distance/86400) * 100$
```

The day of week is calculated in the same manner since the nature of the distance is the same, the difference is the scale of the values. The Timestamp is converted to days since the start of week and the maximal value is 7 because a week has 7 days.

Domains

Domains are the locations visited on the web and it is represented as a string array. The strings in the arrays are compared similarly as the machine ids, either similar or dissimilar. The distance between the two lists of domains is then calculated by taking the number of similar domains and divide it with the length of the shortest list. The number of similar domains is calculated by comparing the lists in a double loop. Since the distance value from dividing the number of similar domains on the length of the shortest list is return the similarity between them, and not the distance the function normalize the result to the 0 to 100 scale and return the value subtracted by 100 to reverse it. This is illustrated in algorithm 6.

Algorithm 6 Algorithm for distance between two list of domains

```
1: $Count\ number\ of\ equal\ items\ in\ the\ two\ lists(list1,\ list2)$
2: if $number\ of\ equal\ items == 0$ then return 100
3: if $list1 > list2$ then
4:     $distance = 100 - (number\ of\ equal\ items/list2.length * 100)$
5: else
6:     $distance = 100 - (number\ of\ equal\ items/list1.length * 100)$
   return $distance$
```

Recording pattern

Calculating the distance between two time series is done with Dynamic Time Warping (DTW). Machine Learning Python[5] or mlpy[12] is a library providing a large set of both supervised and unsupervised machine learning functionality. The library has a great implementation of the Dynamic Time Warping algorithm[13] and this application use this function to calculate the distance between the recording patterns in two sessions. The DTW function does not return a result in a range, so to normalize the data a limit is set where any value higher than this result in the distance 100. I found the result 1 from the DTW function to be a good maximum value and the result is then simply multiplied by 100 to normalize it.

5.4.2 Cluster analysis

When the application has created a distance matrix as described in the previous subsection, it can perform the cluster analysis on the sessions. Hierarchical clustering is a common clustering technique, and because of this there exist a large number of implementations for it. In Python one of the most

[12]`http://mlpy.sourceforge.net/`

[13]`http://mlpy.sourceforge.net/docs/3.5/dtw.html`

popular implementations is in the SciPy[23] ecosystem. This implementation provide the configuration the application require to perform the cluster analysis and is well suited for the task. To use the scipy.cluster.hierarchy.fcluster[14] function, the application need to create a linkage matrix from the distance matrix to provide the function as a parameter. The function also require a threshold parameter to determine when to stop clustering. The linkage matrix can be obtained from the scipy.cluster.hierarchy.linkage[15] function. It require a condensed version of the distance matrix along with the linkage method, which is average linkage. For the threshold parameter of the function I use the elbow point, which is covered in subsection 4.3.2. Preliminary testing indicate the number of clusters usually amount between 2 and 3.

After the fcluster function has run, the application use the output from the function to assign sessions to the different contexts (clusters) based on the data. A new set of contexts is created in the database and sessions are assigned accordingly. The old and now empty contexts are then deleted as they are no longer of use.

5.4.3 Word cloud creation

After the cluster analysis has been performed and new contexts are created, the application will create word clouds for each context to be used as a label in the Recommender system. The creation of word cloud images is done with an external package called WordCloud[16]. The package have a lot of functionality like using an image to determine the shape of the cloud and setting the colours to be used. The result is a 400*200 png image with the words provided sized by number of occurrences with different colours. An example of a word cloud created can be seen in figure 5.2. The words in the image are not sized based on the number of occurrences directly, but also affected by the need to make the words fit nicely. The application get the words for the word cloud by collecting all keywords for each session in each context. After the word cloud images are created, they are stored to disk in a location on the web application in the Recommender. They are stored in a folder called images, so they can be accessed by the path 'images/{id}.png' where id is the context id from the database. Word clouds can also be used

[14]`http://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.fcluster.html`

[15]`http://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.linkage.html`

[16]`https://amueller.github.io/word_cloud/`

to evaluate the results from the clustering.

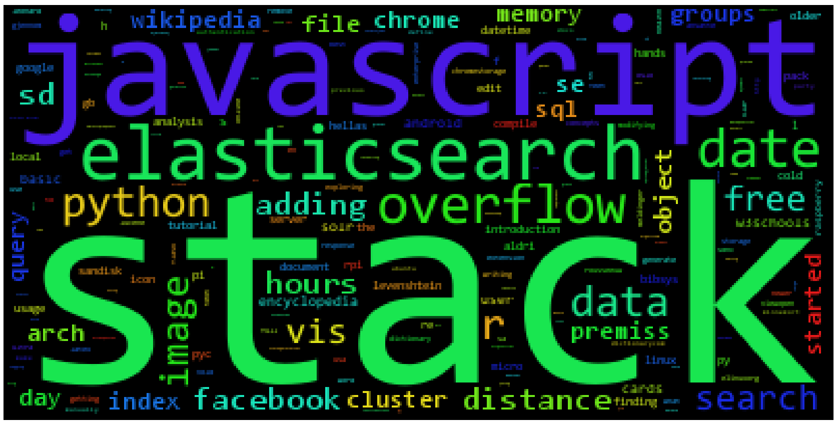


Figure 5.2: Example of word cloud from keywords in context

5.5 Recommender

The Recommender is a web application written in Java and JavaScript. It has a component that collect and store articles called the Article Retriever. There is two parts to the Recommender application, the back-end written in Java, and the front-end written in JavaScript. These parts will be covered separately after the Article Retriever in the following subsections.

5.5.1 Article Retriever

The Article Retriever is a Java application that collect items from web feeds and store them for the Recommender. The application use Maven[17] for simple dependency management. The dependencies for the application are ElasticSearch Java API[18], Apache HttpComponents[19] and Gson[20]. ElasticSearch is a full-text search engine the Recommender use. The Article Retriever will use the ElasticSearch Java API to store the articles retrieved in ElasticSearch.

[17]`https://maven.apache.org/`
[18]`https://www.elastic.co/guide/en/elasticsearch/client/java-api/current/`
[19]`https://hc.apache.org/`
[20]`https://code.google.com/p/google-gson/`

The Article Retriever is a continuous job that for each hour check the feeds specified for new items. The source or address for the feeds are specified in a .txt file that is read for each check. The Article Retriever parses the list of urls in the file and download the xml from each source. Each xml is a RSS representation of the source content. The application parse the xml and create java objects representing the feed and the feed items. The feed items is then stored in ElasticSearch. ElasticSearch uses json as data format, so the application uses Gson to convert the java objects to json before storing the items.

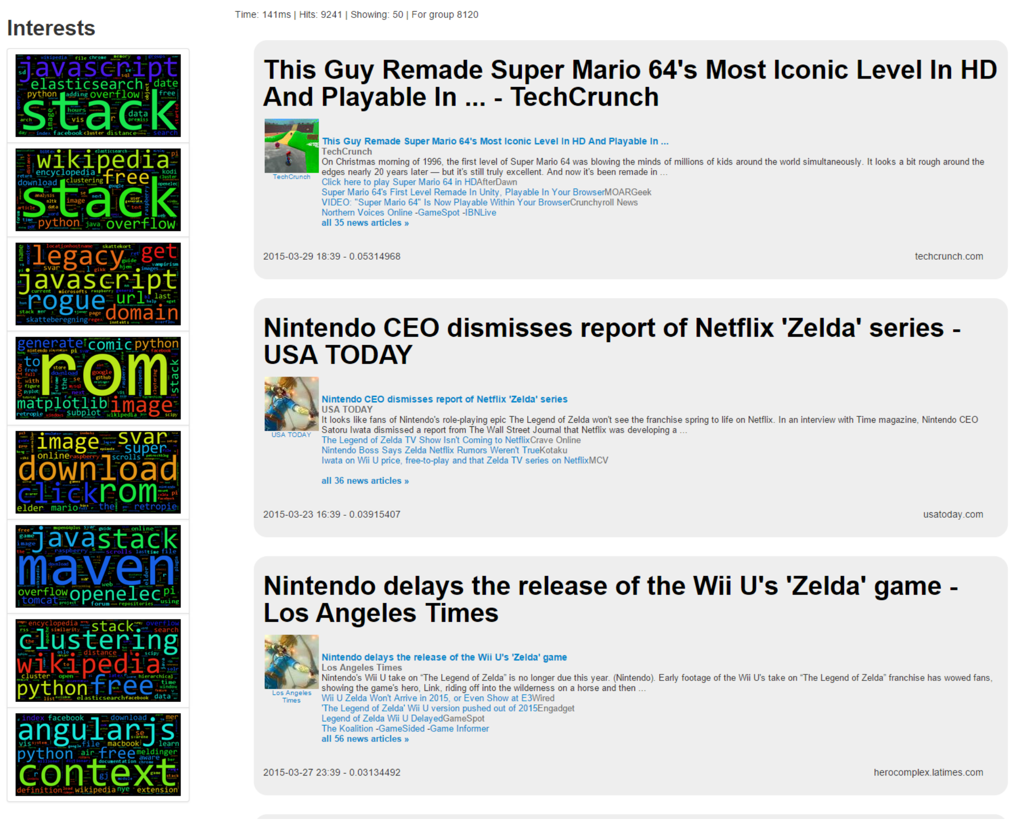
5.5.2 Front-end

The front-end of the recommender is using the AngularJS[21] framework and the jQuery[22] library. Using the AngularJS framework make it possible for two way data binding which make developing web applications significant more enjoyable and efficient. AngularJS is made for dynamic websites, and does this well because it extends html and JavaScript to provide powerful dynamic functionality. The front-end application consist of one module. A module in AngularJS is a container for controllers, services, and other components. The module contains a controller and a factory. The factory contains the http request methods for fetching contexts and articles from the back-end. The controller contains a listener for the user identification event sent by the Interest Retriever when the site is loaded. When the listener is triggered and the user identification is received, the application send a request to the back-end to get the contexts that belong to the user identification. If the contexts are returned, the application also requests the articles for the first context, so the websites have some initial content. The controller also contains some support functions and a method for fetching articles from the back-end based on the context id. This function is called when a user select a context to fetch articles for.

When the application get the user's contexts from the back-end, a list in the view is populated by images of word clouds. This can be seen to the left of figure 5.3. The word clouds are generated by the Interest Analyser (section 5.4) and are stored with the name set to the context id they belong to. So the list of images fetches the correct image based on the context id. The images have click event listeners and when pressed trigger the function to

[21]`https://angularjs.org/`
[22]`https://jquery.com/`

Figure 5.3: Screenshot of recommender front-end



Time: 141ms | Hits: 9241 | Showing: 50 | For group 8120

This Guy Remade Super Mario 64's Most Iconic Level In HD And Playable In ... - TechCrunch

TechCrunch

This Guy Remade Super Mario 64's Most Iconic Level In HD And Playable In ...
TechCrunch
On Christmas morning of 1996, the first level of Super Mario 64 was blowing the minds of millions of kids around the world simultaneously. It looks a bit rough around the edges nearly 20 years later — but it's still truly excellent. And now it's been remade in ...
Click here to play Super Mario 64 in HDAfterDawn
Super Mario 64's First Level Remade In Unity, Playable In Your BrowserMOARGeek
VIDEO: "Super Mario 64" Is Now Playable Within Your BrowserCrunchyroll News
Northern Voices Online -GameSpot -IBNLive
all 35 news articles »

2015-03-29 18:39 - 0.05314968 techcrunch.com

Nintendo CEO dismisses report of Netflix 'Zelda' series - USA TODAY

USA TODAY

Nintendo CEO dismisses report of Netflix 'Zelda' series
USA TODAY
It looks like fans of Nintendo's role-playing epic The Legend of Zelda won't see the franchise spring to life on Netflix. In an interview with Time magazine, Nintendo CEO Satoru Iwata dismissed a report from The Wall Street Journal that Netflix was developing a ...
The Legend of Zelda TV Show Isn't Coming to NetflixCrave Online
Nintendo Boss Says Zelda Netflix Rumors Weren't TrueKotaku
Iwata on Wii U price, free-to-play and that Zelda TV series on NetflixMCV

all 36 news articles »

2015-03-23 16:39 - 0.03915407 usatoday.com

Nintendo delays the release of the Wii U's 'Zelda' game - Los Angeles Times

Los Angeles Times

Nintendo delays the release of the Wii U's 'Zelda' game
Los Angeles Times
Nintendo's Wii U take on "The Legend of Zelda" is no longer due this year. (Nintendo). Early footage of the Wii U's take on "The Legend of Zelda" franchise has wowed fans, showing the game's hero, Link, riding off into the wilderness on a horse and then ...
Wii U Zelda Won't Arrive in 2015, or Even Show at E3Wired
'The Legend of Zelda' Wii U version pushed out of 2015Engadget
Legend of Zelda Wii U DelayedGameSpot
The Koalition -GameSided -Game Informer
all 56 news articles »

2015-03-27 23:39 - 0.03134492 herocomplex.latimes.com

get articles for the specific context. Articles populate a dynamic list that contains the headline, a html summary (that can contain images, urls etc.), date and time for the article, a score that tell how relevant the article is and the name of the article source. The articles can be seen visualised to the right in figure 5.3. Each article headline is a link to the actual article. A small top bar contain information from the ElasticSearch query results and include how much time the request took to perform, how many hits in total and how many of them are displayed.

5.5.3 Back-end

The back-end is a CRUD application written in Java with Maven as dependency manager. It use Jersey[23] to provide a RESTful API to the front-end.

[23]`https://jersey.java.net/`

It use mysql-connector[24] for connection to the MySQL database. It also use ElasticSearch Java API[25] for interaction with the ElasticSearch engine. The back-end application provide an API with three endpoints for the front-end to utilize.

- /groups/{id} - Get context list for a user id
- /images/{id}.png - Get word cloud image for a context id
- /articles/{id} - Get article list for a context id

The first endpoint return a list of contexts created by the Interest Analyser (subsection 5.4). These contexts are used by the front-end to provide users the choice of what context to get recommendations based on. The contexts are collected from the MySQL database since the Interest Analyser store them there. The front-end use the context ids for the second endpoint to get a word cloud image also created by the Interest Analyser. The word cloud image is used to describe and identify the contexts for the user. The image is stored on the web server in a folder and is simply accessed through the folder name (images) and by name (context id) extended with the image type.

The third endpoint is the most comprehensive and the main aspect of the recommendation system. The endpoint provide a list of recommended articles based on the interests in the context selected. To accomplish the recommendations, a full text search engine with support for query boosting is utilized. This is done because the efficiency and accuracy of existing search engines surpasses that of what I could have built with limited time and resources. The recommendation system is also for testing purposes, and using a system that is tested and used by a number of large enterprises is good to avoid the need for in-depth testing of the testing system.

Search engines

The system requirements for the search engine is that it is easy to use, provide good full text search results, support term boosting, and at last is reasonable fast with a reasonable low footprint. There are a few candidates available

[24]`https://www.mysql.com/products/connector/`

[25]`https://www.elastic.co/guide/en/elasticsearch/client/java-api/current/`

that would have done the job well. Solr[26], Sphinx[27] and ElasticSearch[28] are the most prominent candidates. They all cover the requirements reasonable well, but to a varying degree. Both Solr and ElasticSearch is built on top of Apache Lucene[29] and use the Apache Lucene core as the basis for the search functionality. Sphinx is on the other hand built from the ground up. Of these three candidates the most similar and also feature rich is Solr and ElasticSearch. Sphinx is a good candidate for very large indexing jobs on existing databases but in short that is the only significant advantage it holds over Solr and ElasticSearch, and this is not needed in this use-case. Both ElasticSearch and Solr would be a good match for the requirements and are quite similar functionality wise[30]. I have previous experience with ElasticSearch and this influences the choice to be ElasticSearch, since it will be easier and quicker to get started. I know from experience and recommendations that ElasticSearch is very easy to setup and use. It is also the system with highest popularity increase rate[31], which is positive even if it is currently below Solr in popularity.

ElasticSearch

ElasticSearch is a real-time search engine, and this means the data is available for search normally within a second of being indexed. In ElasticSearch data items is referred to as documents, and these are the items that are indexed and then searchable. A document is expressed in json, which is an internet data interchange format. An index is a collection of documents that is somewhat similar. In our case the collection of articles will be in a single index. ElasticSearch operates with clusters and nodes. A node is a server (virtual or physical) that contain data and participate in a cluster's indexing and search capabilities. A cluster is simply a collection of nodes with a common indexing and search mechanism. A cluster can be populated by just a single node. For redundancy and performance ElasticSearch also supports sharding and replication. An index can be divided between multiple nodes, this is called sharding. This will enable an index to exceed the capabilities in both storage and performance of a single node. The system can also create backup shards which will duplicate data from other shards for redundancy.

[26]`https://lucene.apache.org/solr/`
[27]`http://sphinxsearch.com/about/sphinx/`
[28]`http://www.elasticsearch.org/overview/elasticsearch/`
[29]`https://lucene.apache.org/`
[30]`http://solr-vs-elasticsearch.com/`
[31]`http://db-engines.com/en/ranking_trend/search+engine`

ElasticSearch provide a simple and powerful REST api to interact with the cluster. Alternatively there exist a number of language specific interfaces to use. For example the recommender system use the ElasticSearch Java API. With the api a user can: (1) Check cluster, node and index health, status and statistics. (2) Administer cluster, node and index data and metadata. (3) Perform create, read, update, delete and search operations against the indexes. (4) Execute advanced search options like paging, sorting, filtering, scripting, faceting, aggregations and more. Searching in ElasticSearch, which is the main attribute we are interested in, is simple and powerful. ElasticSearch provide a massive amount of query options to customize the search capabilities. The recommender system require the ability to search for a large number of keywords with individual term weighting. This is possible in the ElasticSearch query by utilizing a bool query with a boost attribute. The bool query is a query that allow multiple queries to be nested within. For the recommender system this could entail a series of multi match queries for each keyword with a boost attribute. A multi match query is a query that searches multiple attributes on a document, as for this system it will be the title and the summary of the article. By nesting a series of boosted queries for each keyword, ElasticSearch will return a list of documents by the degree of match with the query, with a score value for how well of a match the document is. The query is constructed and executed with the ElasticSearch Java API. This can be done quite simply by creating a bool query from the QueryBuilder provided by ElasticSearch and for each keyword (and boost value) create a multi match query object and add it to the bool query.

Chapter 6

Results and evaluation

This chapter present and evaluate the results of the system developed in this thesis. The evaluation is performed on user data collected from the author of this thesis over a 77 day long period and on generated fictive user data from use cases. Some of the results is also compared with the results achieved from using different linkage and number of clusters for the clustering algorithm.

The method used to evaluate the results is presented in section 6.1. The results and evaluation of the user profiles created with user data is presented in section 6.2. Section 6.3 contain the results and evaluation of the generated data results. Section 6.4 contain the results and evaluation using different linkage on the full user dataset. Section 6.5 contain the results and evaluation from increasing the number of clusters on the full user dataset. A summary of the evaluations is in section 6.6.

6.1 Method

The evaluation compare and evaluate the clusters in the user profiles created by the system. There are two general approaches for evaluating the results of a clustering algorithm, internal and external validity indices[21]. Internal validity will measure the quality of the clusters in regards to the dataset the clustering is performed on. External validity will measure the quality of the clusters in regards to the expected results. Both these measures will be explained further in subsection 6.1.1 and 6.1.2.

The evaluation will look at the results of clustering with different linkage to see if there is configurations that yield better results. The linkage for the hierarchical clustering algorithm used in the system is *average linkage*. The clustering performed on the user data and synthetic data described above will also be performed with *single*, *complete* and *weighted average linkage* to see if the results improve.

The evaluation will also look at the results from increasing the number of clusters to see if the method of determining the number of clusters work well, and if the higher number of clusters result in an increased accuracy.

6.1.1 Internal validity

Internal criteria can be seen as evaluating the clusters based on the vectors of the dataset themselves[20]. The dataset would constitute the distance matrix the clustering is performed on. Typical internal criteria evaluation is to look for high intra-cluster similarity and low inter-cluster similarity[25]. This is when the items in the clusters are similar, but not similar to items in other clusters.

Based on "An extensive comparative study of cluster validity indices"[6] by Olatz Arbelaitz et.al., the Silhouette Coefficient is generally the best score to use for evaluating internal clustering validity. The silhouette score is covered more in depth in section 2.6.5.

6.1.2 External validity

External criteria rely on comparing the results to a pre-defined expectation of the results[20]. This is in its most basic form to see if the results are random or not. An example of this could be the clustering of "jaguar" objects. We may expect the results of the clustering to be three clusters, one for each class of object, jaguar the animal, the car and the operating system.

Performing cluster evaluation on external criteria is difficult and not an accurate measure. This is because the system is performing unsupervised machine learning and there is no correct data to compare the results to in this case. The only pre-defined expectation to compare the results to, is the assumption that similar interests will, to a degree, be contained in sessions that are clustered into groups. The expectation is that the keywords, representing interests, should be somewhat semantically related in each cluster. The system visualise the keywords by creating word clouds for each cluster. These word clouds will be used to evaluate the clusters based on the pre-defined expectation. Evaluating the correctness of the interests represented by word clouds for each context require knowledge of the actual interests and contexts they represent. The external validity evaluation is a subjective evaluation based on how accurate the author of this thesis think the results look based on his own perception of the reality.

Common external clustering validity measures like entropy, purity and F-measure cannot be utilized without a pre-designated structure or classes to compare the results to [34]. Since the generated data have a correct answer to compare the results to, the precision and recall for the clusters will be calculated. This is covered more in depth in section 6.3.

6.2 User data

The user data is evaluated based on three user profiles created. They are generated from the data recorded by the author of this thesis in a time period of 77 days. Each is generated based on a portion of the data collected. The first user profile with only the first 25 days of the dataset. The second user profile with the first 50 days of the dataset and the third with the full 77 days.

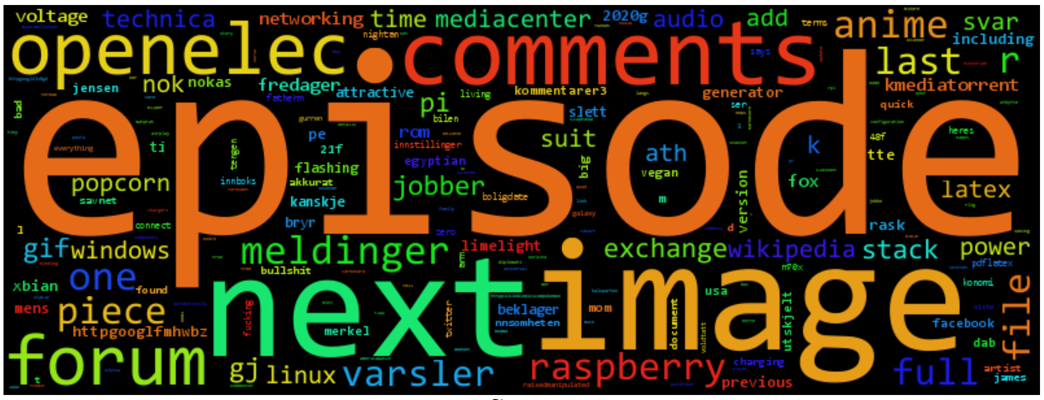
There is only user data from a single user, the author of the thesis. There was no testers willing to let the system monitor their actions for an extended period of time. The reason for this is that the evaluation require manual overview and analysis of the word clouds generated to evaluate the success of the clustering. This is however not that bad. It would have been problematic to evaluate the word clouds for other users in some instances without prior knowledge of the usage of the testers.

6.2.1 Results

The results presented is marked with the result number, the dataset and the cluster number. The results contain a word cloud visualizing the keywords in the cluster based on the number of occurrences. Note that the size of the keywords are not directly related to the number of occurrences, but might vary to fit the words nicely. For a more accurate presentation of the keywords and the number of occurrences, a word count list is included with the 80 most popular terms in the cluster. The results from the complete dataset also contain the headlines from the articles recommended by the recommender system, along with the source and accuracy score given by the system. The accuracy score measure the relevance of the article to the keywords and weighting in the user profile.

The silhouette score of the results is presented in table 6.1.

Result 1: 25 days
Cluster 1/3



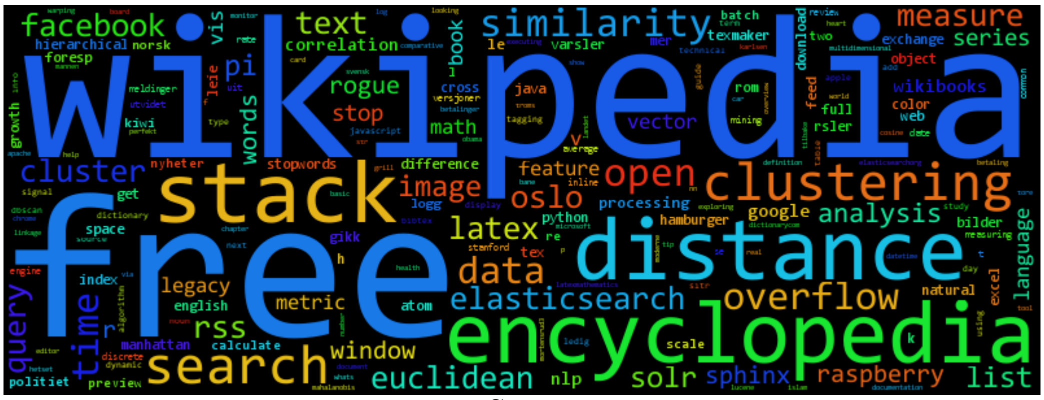
10 Sessions

Ordered word count list*

7 next	2 popcorn	1 episode	1 siste
7 episodes	2 time	1 norsk	1 uken
6 comments	2 windows	1 barnevern	1 commits
6 images	2 linux	1 mens	1 raspberrypi-firmware
5 openelec	2 gjør	1 obama	1 github
4 meldinger	2 jobber	1 merkel	1 airplay
4 anime	2 wikipedia	1 snakket	1 doesnt
4 one	2 mediacenter	1 diplomati	1 work
4 piece	2 power	1 virkeligheten	1 nightan
4 raspberry	2 add	1 ukraina	1 egyptian
4 forum	2 latex	1 oppskriften	1 satirist
4 file	2 audio	1 bedre	1 americawatch
3 varsler	2 technica	1 økonomi	1 mac
3 last	2 nok	1 piother	1 nokas-utbytte
3 full	2 svar	1 mini	1 ti
3 pi	1 forespørsler	1 computers	1 år
3 gif	1 quick	1 version	1 måtte
3 stack	1 slett	1 mer	1 james
3 exchange	1 facebook	1 innstillinger	1 gå
2 suit	1 previous	1 søket	1 km

*List contain the 80 most popular words out of 231 in total

Result 1: 25 days
Cluster 2/3



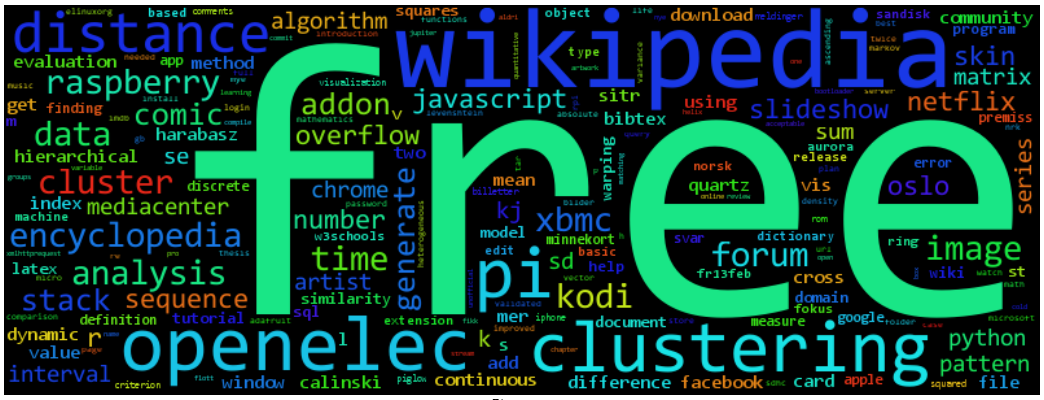
15 Sessions

Ordered word count list*

44 wikipedia	10 list	7 books	5 java
42 free	10 text	7 hierarchical	5 tex
35 encyclopedia	10 analysis	7 google	5 growth
29 distance	10 raspberry	6 python	5 manhattan
27 stack	10 pi	6 get	5 measure
27 clustering	10 solr	6 english	5 bilder
24 search	9 words	6 excel	5 vector
19 similarity	9 sphinx	6 vs	5 download
18 open	8 windows	6 correlation	5 difference
17 overflow	8 rogue	6 space	5 varsler
16 time	8 legacy	6 feature	5 les
15 data	8 images	6 measures	5 mer
15 elasticsearch	8 vis	6 metric	5 full
13 rss	8 stop	6 atom	5 exchange
12 latex	8 language	6 color	5 index
12 facebook	8 math	6 texmaker	5 leie
12 oslo	8 series	6 hamburger	5 rom
11 cluster	7 processing	5 gikk	5 web
11 euclidean	7 nlp	5 stopwords	5 forespørsler
11 query	7 wikibooks	5 natural	4 date

*List contain the 80 most popular words out of 1411 in total

Result 1: 25 days
Cluster 3/3



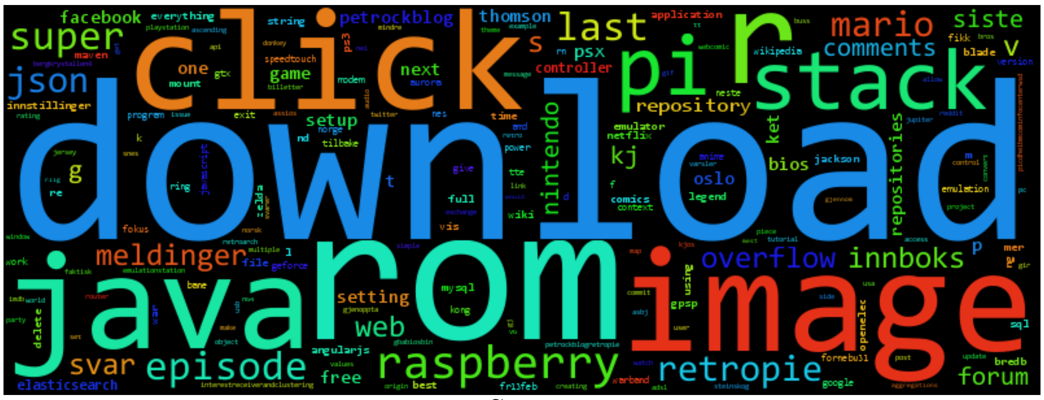
36 Sessions

Ordered word count list*

38 wikipedia	12 netflix	7 get	5 svar
38 free	11 overflow	7 vis	5 windows
37 openelec	11 slideshow	7 dynamic	5 latex
36 clustering	10 oslo	7 quartz	5 apple
30 pi	10 addon	7 file	5 program
29 encyclopedia	9 artist	7 pattern	5 premise
28 distance	9 mediacenter	7 sequence	5 dictionary
25 raspberry	9 python	7 hierarchical	5 methods
22 kodi	9 number	7 value	5 finding
21 data	8 se	6 facebook	5 clusters
21 analysis	8 using	6 mer	5 minnekort
20 images	8 download	6 difference	5 edit
20 forum	8 matrix	6 continuous	5 definition
18 generate	8 algorithm	6 sitr	5 help
17 time	8 sd	6 bibtex	5 addons
17 comic	8 community	6 cards	5 wiki
16 xbmc	8 series	6 two	5 warping
15 stack	8 index	6 skin	5 interval
15 cluster	8 sum	6 evaluation	5 similarity
14 javascript	8 chrome	5 w3schools	5 calinski-harabasz

*List contain the 80 most popular words out of 1635 in total

Result 2: 50 days
Cluster 1/2



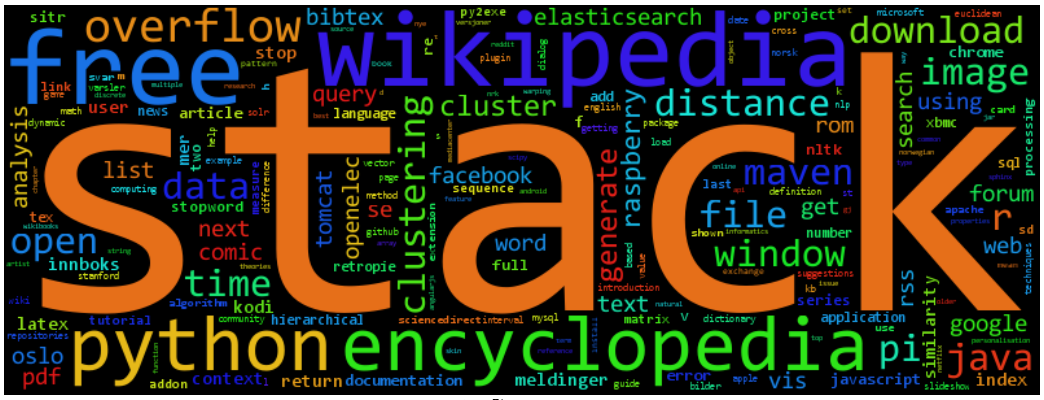
34 Sessions

Ordered word count list*

32 download	8 next	6 string	5 mysql
25 rom	8 one	6 vs	5 jackson
23 images	8 comments	6 sql	5 best
22 click	8 søket	6 vis	5 google
19 java	8 siste	6 thomson	5 bredbånd
17 stack	8 repositories	6 delete	5 neste
15 pi	8 web	6 war	5 context
14 raspberry	8 repository	6 ps3	5 games
13 last	8 petrockblog	6 controller	5 comics
13 super	8 psx	6 emulator	5 mount
12 json	7 oslo	6 gpsp	5 blade
12 retropie	7 forum	5 netflix	5 wiki
12 mario	7 free	5 everything	5 kong
11 meldinger	7 elasticsearch	5 time	5 legend
11 svar	7 bios	5 full	5 zelda
11 overflow	6 mer	5 openelec	4 kjøp
10 innboks	6 innstillinger	5 wikipedia	4 billetter
10 nintendo	6 application	5 file	4 program
9 episodes	6 using	5 angularjs	4 aurora
8 facebook	6 maven	5 javascript	4 fokus

*List contain the 80 most popular words out of 1375 in total

Result 2: 50 days
Cluster 2/2



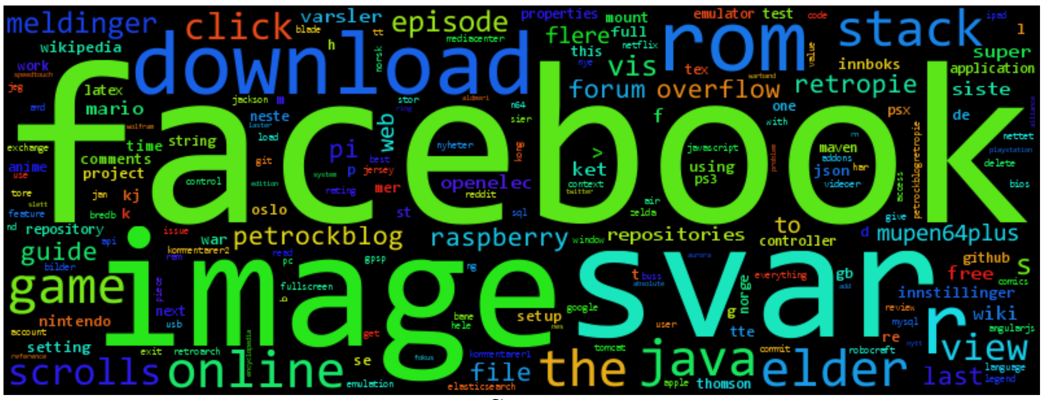
97 Sessions

Ordered word count list*

123 stack	34 using	24 similarity	17 two
114 wikipedia	34 google	23 return	17 nltk
113 free	33 analysis	23 index	17 hierarchical
90 encyclopedia	33 next	23 rss	17 roms
88 python	32 facebook	23 tomcat	16 news
86 overflow	32 se	22 mer	16 stopwords
70 clustering	32 web	22 kodi	16 language
63 java	31 comic	22 forum	16 processing
63 distance	31 vis	21 meldinger	16 full
45 download	31 search	21 latex	16 series
44 data	30 text	21 stop	16 sql
41 time	30 cluster	21 list	16 retropie
41 pi	29 pdf	21 innboks	15 svar
40 maven	28 get	20 javascript	15 last
39 generate	28 oslo	20 context	15 image
39 file	27 bibtex	20 documentation	15 matrix
38 raspberry	27 query	19 chrome	15 article
38 elasticsearch	26 windows	19 sitr	15 use
37 openelec	25 open	17 project	15 xbmc
35 images	24 words	17 tex	15 error

*List contain the 80 most popular words out of 3995 in total

Result 3: 77 days
Cluster 1/2



68 Sessions

Ordered word count list*

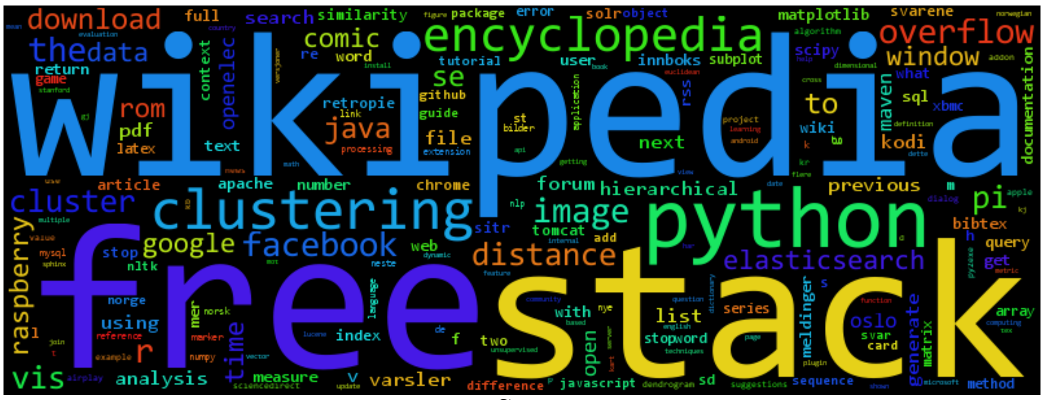
80 facebook	18 last	12 using	8 maven
64 images	18 retropie	12 json	8 repository
44 svar	17 mupen64plus	12 games	8 properties
39 download	17 to	12 nintendo	8 neste
32 java	16 varsler	11 wikipedia	8 ps3
32 the	16 openelec	11 web	8 psx
31 rom	16 repositories	11 se	8 this
31 stack	16 guide	10 oslo	7 anime
31 online	16 flere	10 episode	7 work
31 elder	15 wiki	10 full	7 latex
31 scrolls	15 super	10 war	7 bilder
30 view	14 søket	10 de	7 får
28 click	14 siste	10 gb	7 javascript
27 meldinger	14 forum	9 mer	7 elasticsearch
24 vis	14 free	9 comments	7 tomcat
21 petrockblog	14 file	9 time	7 string
20 raspberry	13 episodes	8 next	7 vs
20 pi	13 innstillinger	8 one	7 google
20 overflow	13 innboks	8 github	7 kommentarer1
19 game	13 mario	8 application	7 project

*List contain the 80 most popular words out of 2498 in total

Top headlines from recommender system

Headline	Source	Score
This INSANE Facebook letter is outraging everyone online	komando.com	0.06397697
Slik «snikleser» du Facebook-meldinger	aftenposten.no	0.06322565
Google now lets you download your search history	cnet.com	0.059937824
Facebook brings mobile Messenger app to desktops	cnet.com	0.056345195
Facebook introduces Hello, an app to replace the Android dialer	theverge.com	0.05602319
Viber vs Facebook Free Messenger on iOS, Windows PC and Android	thefusejoplin.com	0.05502811
Facebook reveals the logic behind its forced Messenger split	pcworld.com	0.053710956
Facebook Messenger blir så mye mer	dinside.no	0.051333155
New spectroscopic images of Mercury are a rainbow of colour	cnet.com	0.048418887
OnePlus's Android Lollipop-based ROM, OxygenOS, is available to download now	venturebeat.com	0.045409285
Facebook Launches a Mobile Ad Exchange on Top of LiveRail	recode.net	0.043863643
First Click: Nintendo could learn a lot from Netflix	theverge.com	0.043093774
Facebook Removes 'Feeling Fat' Status Expression, Replaces it With 'Feeling ...	socialmediaseo.net	0.04280837
Facebook-tabbe: Gotland har blitt norsk	e24.no	0.04249633
Facebook wants solar drone to bring Internet far and wide	cnet.com	0.042136293

Result 3: 77 days
Cluster 2/2



133 Sessions

Ordered word count list*

156 wikipedia	54 comic	31 web	24 clusters
154 free	48 time	31 with	24 apache
148 stack	44 cluster	30 query	23 chrome
129 python	43 openelec	29 mer	23 sitr
127 clustering	43 google	29 text	23 index
122 encyclopedia	40 generate	29 forum	23 rss
109 overflow	40 pdf	28 meldinger	22 two
90 the	39 using	28 bibtex	22 full
83 facebook	39 search	27 windows	22 solr
66 distance	38 maven	27 get	21 stop
64 vis	37 previous	27 matplotlib	21 wiki
64 pi	34 oslo	26 innboks	21 tomcat
61 se	34 analysis	26 sql	21 svarene
61 java	34 next	26 scipy	21 retropie
60 raspberry	33 varsler	25 number	20 context
59 download	33 kodi	24 vs	19 javascript
59 elasticsearch	33 file	24 open	19 latex
58 data	32 list	24 words	19 stopwords
57 to	32 hierarchical	24 similarity	19 matrix
54 images	32 documentation	24 return	18 svar

*List contain the 80 most popular words out of 5308 in total

Top headlines from recommender system

Headline	Source	Score
Wikipedia to file lawsuit challenging mass surveillance by NSA	ca.reuters.com	0.03325456
Viber vs Facebook Free Messenger on iOS, Windows PC and Android	thefusejoplin.com	0.02752973
The Strange Thing About 'Free-Range Parenting' Is That The Phrase Exists	inquisitr.com	0.024754943
How to Get Free 12 Krispy Kreme Donuts Today	time.com	0.024412466
7 awesome paid iPhone apps on sale for free for a limited time	bgr.com	0.024213756
Facebook lets you choose what to share with 3rd party apps	engadget.com	0.024091031
Google now lets you download your search history	cnet.com	0.024028001
How do you find a Burmese python in the Everglades?	cbsnews.com	0.023992304
Play Cards Against Humanity On the Web, for Free	pcmag.com	0.02348086
3 ways to free up space on your smartphone or tablet	postandcourier.com	0.021198826
An Open Google Now Is About to Make Android Super Smart	wired.com	0.020137079
Adobe's new Slate app aims to turn anyone into a web designer for free	mashable.com	0.019818576
Here's what happens when John Oliver hosts a 'Monty Python' panel	entertainthis.usatoday.com	0.019171933
Yahoo shows off password-free logins and new encrypted email technology	theverge.com	0.019086389
Even pirates will get a free upgrade to Windows 10, Microsoft says (+video)	csmonitor.com	0.018920645
Google takes on real-time big data analysis with new cloud services	pcworld.com	0.018224718

Result	Silhouette score
Result 1	0.362
Result 2	0.377
Result 3	0.430

Table 6.1: Silhouette score for clusters in results

6.2.2 Evaluation

The clusters in the user profiles from the user data is broadly divided into two main contexts based on the results. The two main contexts are work and everything else. This can be seen as a trend through results 1 to 3.

22 days of user data

For the first third of the user data, the user profile has three clusters as presented in result 1, so the system identify three contexts. The first cluster contains 10 of the 61 total sessions. Some of the most prominent words in this cluster is ”next” and ”episodes” as well as ”comments”, ”images”, ”openelec”, ”meldinger” and ”anime”. With the assumption that the context of this cluster is as broad as ”free time activities” or ”everything other than work” the terms fit very well. ”next”, ”episodes” and ”anime” are terms that result from me watching several episodes of an anime series. The terms recorded also include ”one” and ”piece” which combined is the name of the show. The terms ”comments” and ”images” come from the popular news aggregation website Reddit[1] which I frequent in my spare time. The term ”meldinger” is from the social media site Facebook[2]. ”openelec” is an operating system I partially use on my raspberry pi[3], and the terms ”raspberry” and ”pi” are also quite prominent in this cluster. Most of the terms and possibly all of the terms in this cluster is related to my spare time activities and interests. The terms with only one word count seem to be related to news articles I have clicked on. The only possible misplacement in this cluster is the terms ”stack” and ”exchange”, which can be related to work. There is however still a significant chance that the terms could come from

[1]http://www.reddit.com/
[2]http://www.facebook.com/
[3]https://www.raspberrypi.org/

me looking up something on ”raspberrypi.stackexchange.com” or other stack exchange sites. With the assumption that this cluster represent a broad context containing my spare time interests the content of it fit well.

The second cluster contain 15 sessions and have a much higher word count with a total of 1411 words compared to the first cluster which had 231 words. The most prominent terms in this cluster are related to me working on my master thesis. The terms ”wikipedia”, ”enclycopedia” and ”free” are the three most recorded terms in the cluster, and this is most likely because Wikipedia[4] include ”Wikipedia, the free encyclopedia” at the end of the link text in Google[5] results. Terms like ”distance”, ”clustering”, ”search”, ”similarity”, ”time”, ”elasticsearch”, ”rss”, ”latex” and more are directly related to my work, and very prominent in the cluster. The terms that do not fit in this cluster representing the context of work, is ”facebook”, ”oslo”, ”raspberry”, ”pi” and also ”rogue” and ”legacy”. These terms are not insignificant in the cluster in terms of word count values, as they have between 12 and 8 counts each, but they are significantly outnumbered by work related terms.

By not looking at the terms ”wikipedia”, ”free” and ”encyclopedia” the third and last cluster is a mixture of work and spare time interests based on the most popular terms. The terms ”openelec”, ”pi”, ”raspberry”, ”kodi”, ”comic”, ”xbmc” and ”netflix” are obvious spare time activities related terms, and these are quite prominent in this cluster. Equally prominent terms in this cluster are ”clustering”, ”distance”, ”data”, ”analysis”, ”time”, ”cluster” and ”javascript”, which are work related. The last cluster is a mixture of work and other interests without any obvious context.

The first two clusters contain each distinct broad context that was easily identifiable, work and spare time, while the last cluster contained a mixture of the first two contexts.

50 days of user data

For two thirds of the user data presented in result 2, there are two clusters. At first glance it looks like the data is clustered into two contexts, work and spare time interests.

[4]http://en.wikipedia.org
[5]http://www.google.com

The first cluster look like it contain the spare time interests with terms like "download", "rom", "images", "pi", "raspberry", "super", "retropie", "mario", "meldinger", "innboks", "nintendo", "episodes" and "facebook" as very prominent terms in the cluster. The terms reflect my new interest, since the one third user data results, well. The new interest is emulating old consoles on the raspberry pi. This is reflected by the terms "download", "rom", "retropie", "mario", "super" and "nintendo". The cluster also include prominent terms that relate to the context of work, and not spare time interests, such as "java", "stack" and "overflow". Among the less prominent terms there is a mixture of work related and spare time interests related terms. The cluster reflect the context of spare time interests, with flaws.

The second cluster's most prominent terms reflect the context of work neatly, with some exceptions that should optimally have been in the first cluster. The exceptions include "pi", "raspberry", "openelec", "images", "comic" and "oslo". The most prominent terms related to work are "stack", "python", "overflow", "clustering", "java", "distance", "data", "time", "maven" and "elasticsearch". The cluster is based on the word count list mostly populated with work related terms, but there is many terms related to spare time interests.

The two clusters are somewhat successful of representing two very broad contexts. There are too many terms placed in the wrong cluster and the contexts are too broad to call it a complete success.

77 days of user data

Using all user data provide similar results as using two thirds of the user data. There are two clusters, as shown in result 3, which on a broad term represent work and spare time interests.

The first cluster represent the spare time interests context, with a lot of similar terms as with two third user data. The main difference is that the term "facebook" is the most popular term by a large margin compared to previously, and the addition of terms describing the game "The elder scrolls online", which was a new interest in the last 27 days of user data. The terms "java" and "stack" which were the most prominent terms that did not belong in the cluster is still present and fairly prominent.

The second cluster which evidently represent the work context is very similar to the cluster representing the work context with two thirds of the user data.

For the clustering to be successful it should have detected contexts in more detail and with better accuracy.

Silhouette score

The silhouette score (shown in figure 6.1) increase with the size of the dataset used. It does not necessarily mean anything since the score is an indication of how good the clustering is based on the dataset, and the dataset changes, so the results are not directly comparable. The clustering performed with the full dataset score 0.43 with 1 as the best possible score and -1 as the worst possible score. This indicate the clustering is fine based on the dataset.

Recommendations

The recommendations from the recommender system is heavily influenced by the most popular terms in the clusters. In the word count list for cluster 1 in result 3, the word ”facebook” is by far the most popular term, and this is reflected in the recommendations. Of the 15 highest recommended articles based on the user profile from cluster 1 in result 3, the term ”facebook” is present in 11 of them. The results from the recommender system is not affected by time, so articles from today is not more recommended then articles from yesterday. This result in a lot of articles containing popular topics or more specifically terms like ”facebook”. Out of 23 398 articles in total (at the time of receiving the recommendations in results), it is somewhat surprising that not all 15 articles contained the term ”facebook”. This is of course explained by the fact that a combination of other relevant terms might generate a higher score.

The recommendations for the second cluster all have a significant lower score than the first cluster. This is caused by the higher number of terms in the second cluster, since it is harder to find a good match with more terms to match. The recommendations for the second cluster also have much more diverse content than the first cluster, and this can be explained by the most popular terms being relatively equal in their count compared to the first cluster. The recommendations also contain several headlines which contain multiple relevant keywords boosting their importance and making the recommendations more mixed then for the previous cluster.

6.3 Generated data

To evaluate the system further than using only the collected user data, some fictive user scenarios was generated and tested on the system as well. The generated user profiles will be generated based on data ranging from completely structured to completely random, based on fictive use cases. The results from the generated data will be presented with a word cloud for each cluster in the user profile, and with a precision and recall value for each context in the cluster. This is possible since the generated data has pre-designated classes for the data to be grouped by. The precision is the number of terms related to the context in the cluster, divided by the total number of terms in the cluster.

$$Precision = \frac{Terms\ related\ to\ context\ in\ cluster}{Terms\ in\ cluster\ in\ total}$$

The recall is the number of terms related to the context in the cluster, divided by the total number of terms related to the context in total.

$$Recall = \frac{Terms\ related\ to\ context\ in\ cluster}{Terms\ related\ to\ context\ in\ total}$$

Precision and recall together give a great indication of the success of each cluster in the user profiles. Because precision and recall can be calculated, there is no need to list the keywords for each cluster.

6.3.1 Generating fictive user data

There are four different user scenarios that was generated, and they range from structured to unstructured. The scenarios contain fictive contexts that the system tried to detect as accurately as possible. Some of the attribute values is chosen at random in a value range. To ensure that the results used are representative the generating was done several times for each user scenario to see if the results are representative.

The first of the four user scenarios is a distinct work and interest routine, where the fictive user only has two contexts, work and an interest. The second scenario is similar to the first but with two interests in the afternoon with overlapping time ranges. The third scenario is four interests in the same time frame spanning the entire day. Finally, the fourth scenario is four interests in the same time frame and with the same domains, so there is no way to separate them.

- Structured work and single interest
- Structured work and two interests
- Four interests in same time frame
- Four interests in same time frame and with same domains

The following subsections will cover how the sessions in the different scenarios are generated.

Structured work and single interest

The first user scenario consist of two completely different contexts the system will try to detect. The first context is work and the second context a hobby, in this case animals. The attribute values selected for the work context is displayed in table 6.2, and the attribute values for the animals hobby is displayed in table 6.3. Note that in addition to having different time periods and digital locations, they also have different relative locations.

Attribute	**Data type**	**Data range**
Relative location	String	Static value
Time of day start	Timestamp	Random between 07.30 and 08.30
Time of day stop	Timestamp	Random between 15.30 and 16.30
Day of week	Timestamp	Random between Monday and Friday
Digital locations	String array	Random 20 out of 26 total
Interest recording pattern	Timestamp array	Random between 20 and 40 timestamps with between 30s and 60s interval
Keywords	String	Random 10 out of 30 IT work related terms

Table 6.2: Session attribute values for fictive work context

Attribute	**Data type**	**Data range**
Relative location	String	Static value (Different then work context)
Time of day start	Timestamp	Random between 16.30 and 20.00
Time of day stop	Timestamp	Random between 20.00 and 24.00
Day of week	Timestamp	Random between Monday and Sunday
Digital locations	String array	Random 20 out of 26 total
Interest recording pattern	Timestamp array	Random between 20 and 40 timestamps with between 30s and 60s interval
Keywords	String	Random 10 out of 38 animal related terms

Table 6.3: Session attribute values for fictive animal interest context

Structured work and two interests

The second fictive user scenario consist of a work and hobby context similarly to the previous user scenario, but with an additional hobby in the same time frame as the other hobby. The added hobby is of the topic football, and the attribute values is displayed in table 6.4. The contexts for work and animals is identical so the attribute values can be found in table 6.2 and 6.3.

Attribute	**Data type**	**Data range**
Relative location	String	Static value (Same as animal hobby)
Time of day start	Timestamp	Random between 16.30 and 20.00
Time of day stop	Timestamp	Random between 20.00 and 24.00
Day of week	Timestamp	Random between Monday and Sunday
Digital locations	String array	Random 20 out of 26 total
Interest recording pattern	Timestamp array	Random between 20 and 40 timestamps with between 30s and 60s interval
Keywords	String	Random 10 out of 38 football related terms

Table 6.4: Session attribute values for fictive football interest context

Four interests in same time frame

The third fictive user scenario is a scenario with work and three interests at the same relative location, and in the same frame of time. This can be seen as a person working from home and work sporadically along with looking up interests. The attribute values for work, and the three interests can be found at table 6.5. They all share the same attribute values, except for context related terms and digital locations.

Attribute	**Data type**	**Data range**
Relative location	String	Static value
Time of day start	Timestamp	Random between 09.00 and 18.00
Time of day stop	Timestamp	Random between 21.00 and 24.00
Day of week	Timestamp	Random between Monday and Sunday
Digital locations	String array	Random 20 out of 26 total
Interest recording pattern	Timestamp array	Random between 20 and 40 timestamps with between 30s and 60s interval
Keywords	String	Random 10 context related terms

Table 6.5: Session attribute values for fictive interest context

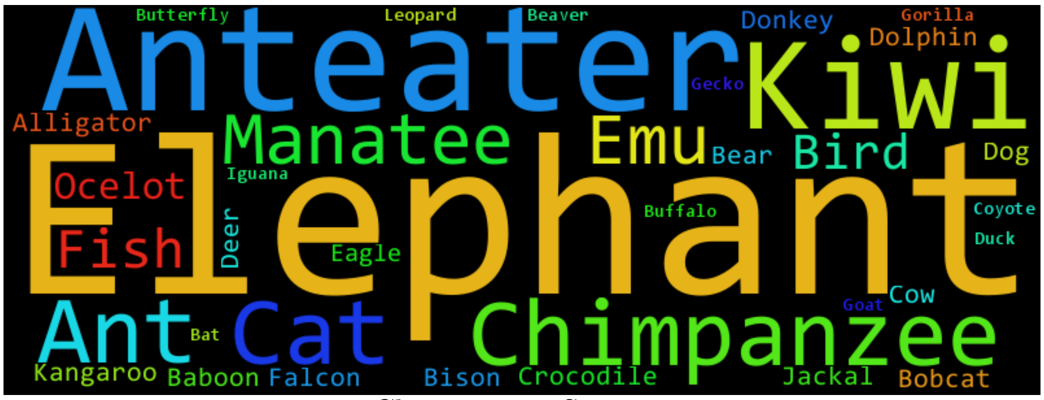
Four interests in same time frame and with same domains

The fourth and final fictive user scenario consist of five different interests with identical time frames and domains/digital locations. This user scenario can be seen as someone with absent of leave from work and use the computer whenever they want to do whatever they want, and with interests that share a similar location on the web. There are five different interests in this scenario; work, animals, football and cars. The session attribute values are identical to the attribute values in table 6.5 used for the previous scenario, but with all interests sharing the same pool of digital locations.

6.3.2 Results

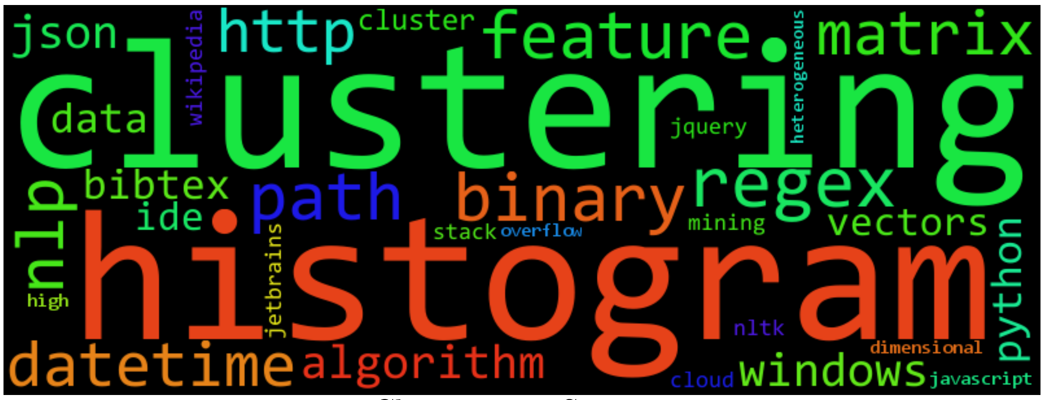
The results listed here have a result number and a title for the user scenario. Each result contain a word cloud and a table with precision and recall for each cluster. At the end the silhouette scores is listed in table 6.6.

Result 4: Structured work and single interest



Cluster 1: 50 Sessions

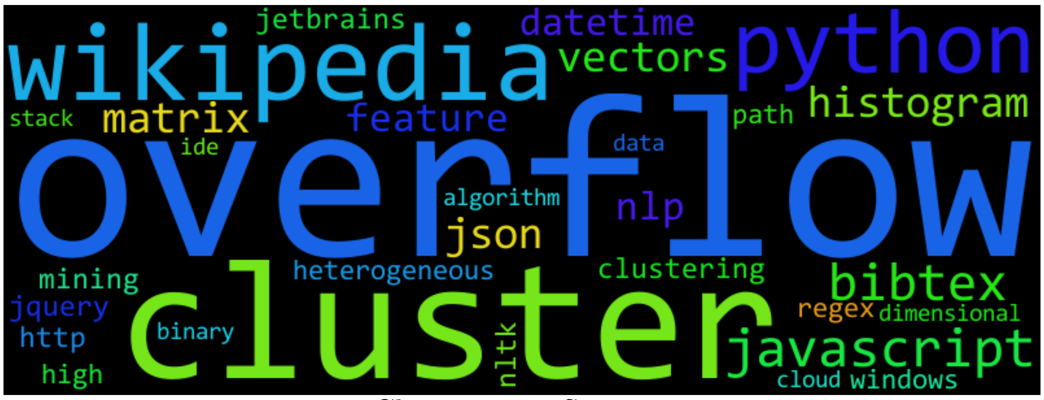
Context	Work	Animals
Precision	0	1
Recall	0	1



Cluster 2: 50 Sessions

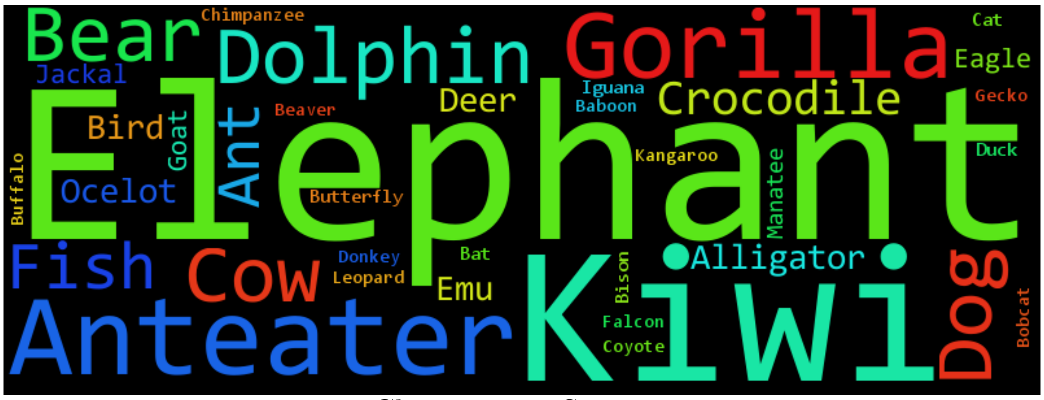
Context	Work	Animals
Precision	1	0
Recall	1	0

Result 5: Structured work and two interests



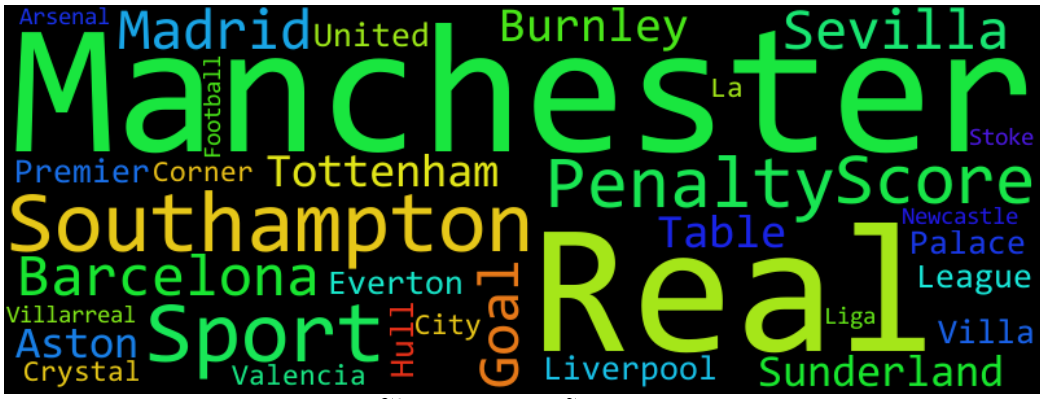
Cluster 1: 50 Sessions

Context	Work	Animals	Football
Precision	1	0	0
Recall	1	0	0



Cluster 2: 25 Sessions

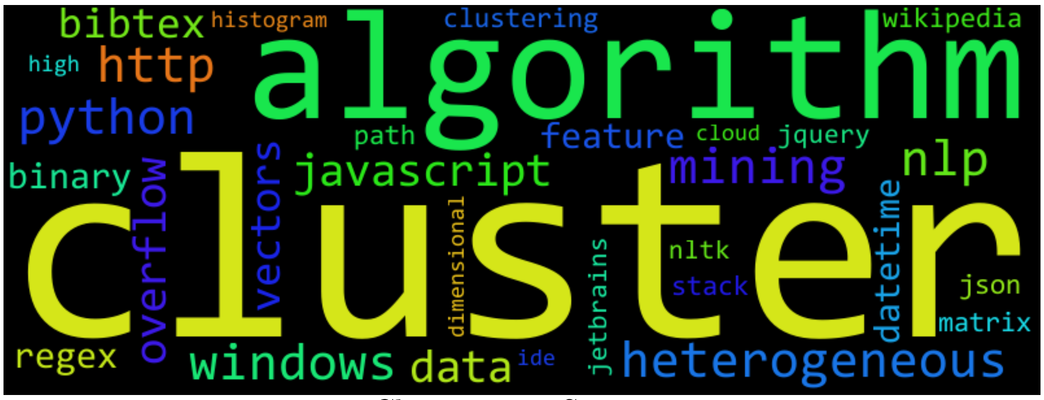
Context	Work	Animals	Football
Precision	0	1	0
Recall	0	1	0



Cluster 3: 25 Sessions

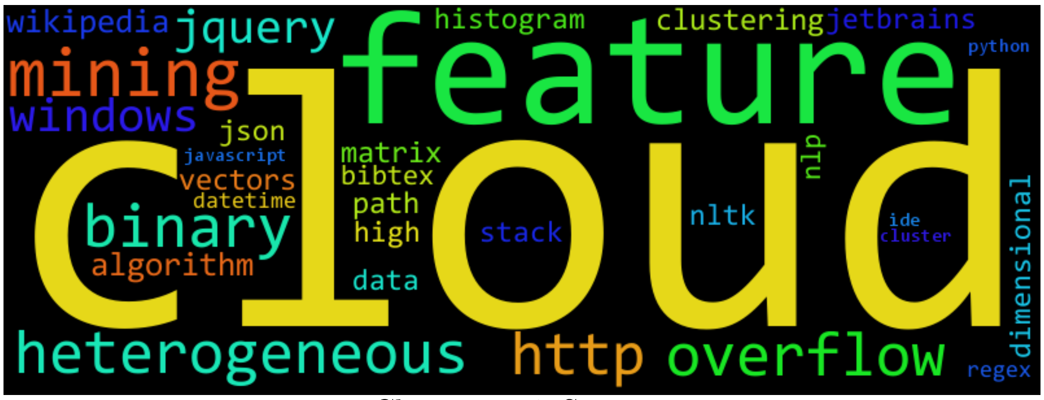
Context	Work	Animals	Football
Precision	0	0	1
Recall	0	0	1

Result 6: Four interests in same time frame



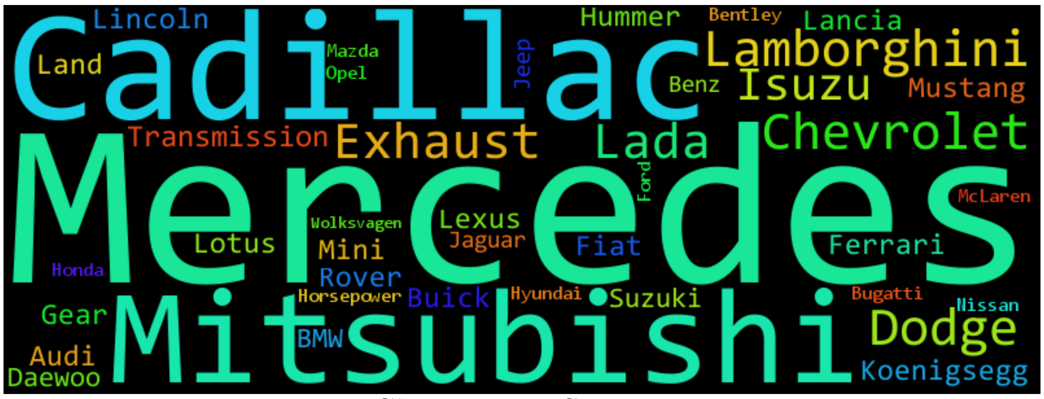
Cluster 1: 13 Sessions

Context	Work	Animals	Football	Cars
Precision	1	0	0	0
Recall	0,26	0	0	0



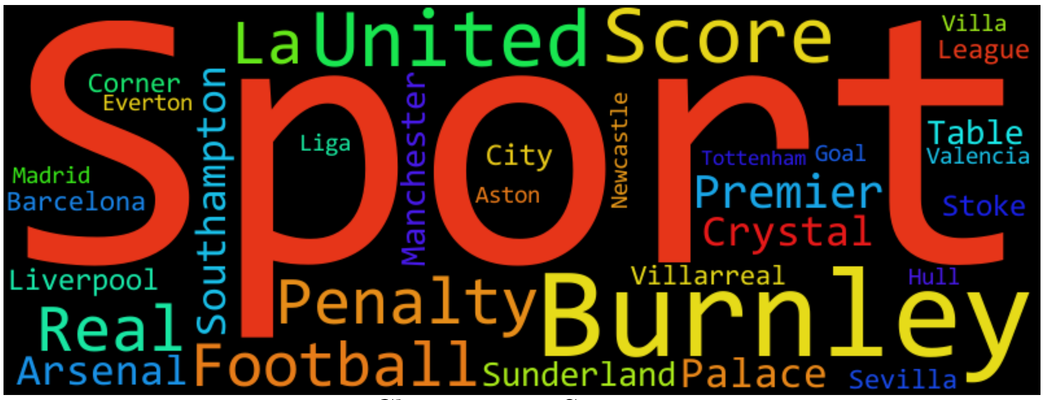
Cluster 2: 37 Sessions

Context	Work	Animals	Football	Cars
Precision	1	0	0	0
Recall	0,74	0	0	0



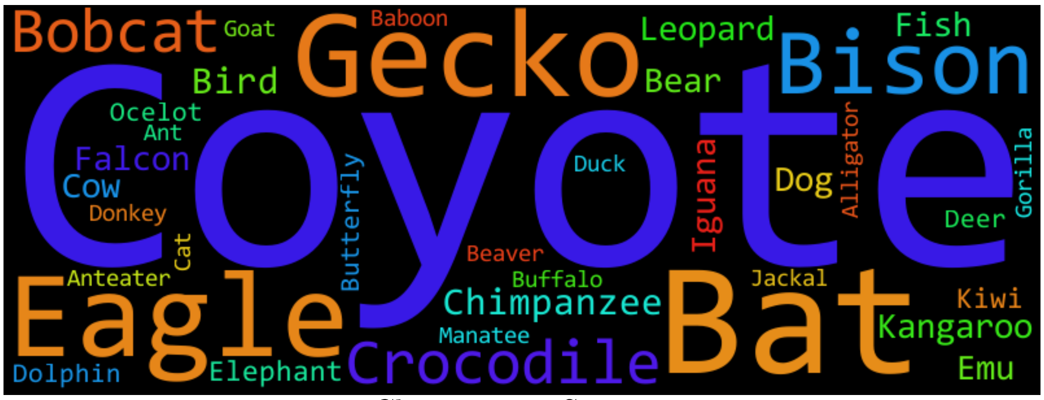
Cluster 3: 25 Sessions

Context	Work	Animals	Football	Cars
Precision	0	0	0	1
Recall	0	0	0	1



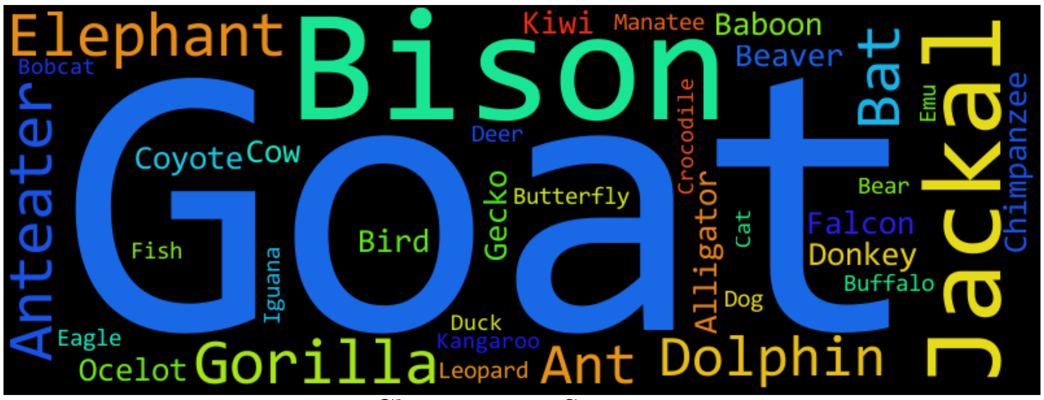
Cluster 4: 25 Sessions

Context	Work	Animals	Football	Cars
Precision	0	0	1	0
Recall	0	0	1	0



Cluster 5: 12 Sessions

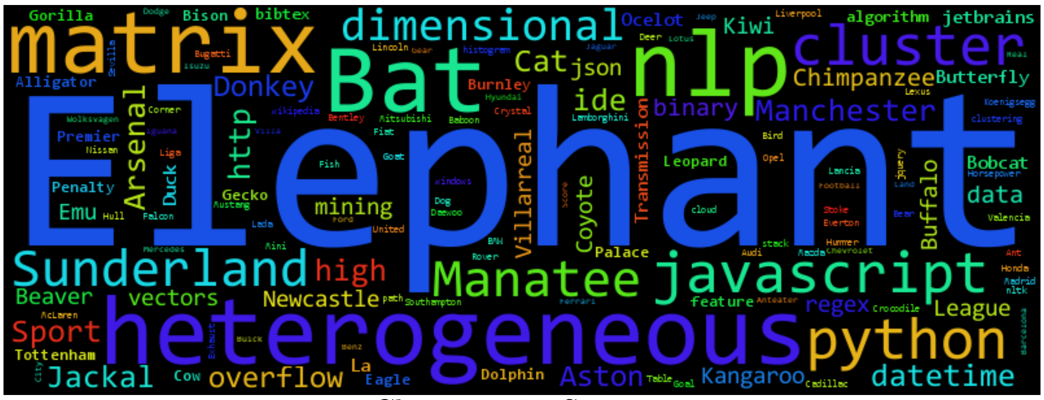
Context	Work	Animals	Football	Cars
Precision	0	1	0	0
Recall	0	0,48	0	0



Cluster 6: 13 Sessions

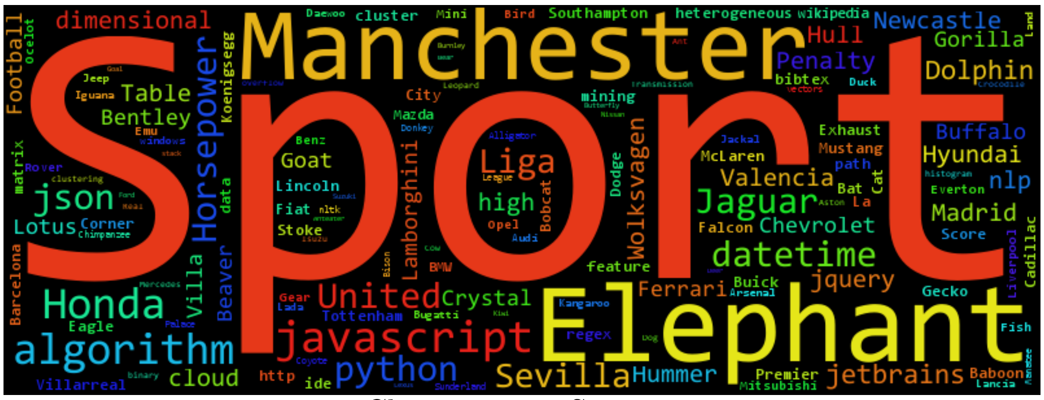
Context	Work	Animals	Football	Cars
Precision	0	1	0	0
Recall	0	0,52	0	0

Result 7: Four interests in same time frame and with same domains



Cluster 1: 77 Sessions

Context	Work	Animals	Football	Cars
Precision	0,3	0,3	0,22	0,19
Recall	0,46	0,46	0,34	0,3



Cluster 2: 123 Sessions

Context	Work	Animals	Football	Cars
Precision	0,22	0,22	0,27	0,28
Recall	0,54	0,54	0,66	0,7

Result	**Silhouette score**
Result 4	0.951
Result 5	0.790
Result 6	0.456
Result 7	0.482

Table 6.6: Silhouette score for clusters from generated data

6.3.3 Evaluation

Result 4 and 5 present the results of the generated data with structured work as one context and one and two other interests. The system successfully detect the different contexts with this data. The work and interests in result 4 and 5 have distinct different times, relative physical location and digital location. So differentiating work and the interests should be, and is, without problems. The system also successfully differentiate between the two interests in result 5, even though the only difference is the digital locations. This is to be expected since the difference in digital location should be enough to differentiate the two when the other attributes are similar. Both result 4 and 5 is completely successful with the best possible score in precision and recall, which is values that indicate how well the clustering is based on the pre-defined contexts.

Result 6 is from four interests in the same time period. The start and stop times have a wide range, and the real difference between all four interests is similarly to the interests in result 5 the digital location. Because of the wide range there is a bigger variance in the time attributes for the sessions, and detecting similar contexts will be more difficult. The system detect six contexts in result 6, while there are only 4 contexts in the dataset. The precision value for all four contexts are as good as it can get, while the recall value for two of the contexts are 0,26 and 0,74, and, 0,48 and 0,52. This combined with the word clouds for cluster 1 and 2, and cluster 5 and 6 show that two of the contexts are split into two clusters each. The system has probably split two of the contexts because the time values have a wide range and some of the sessions values have thus had a large distance, making them too distant to naturally be in the same cluster. The result is not very positive as the system should be able to successfully locate interests with such distinct digital locations. The result indicate that the time attributes of the sessions are weighted too much when the distance is measured, since two of the contexts are split in two clusters, and time is the only major difference.

For result 7 the four interests are constructed similarly to result 6, but the digital location is similar between all interests. This means there is practically no difference between the different contexts. The system behave as expected and cluster the sessions seemingly random in two clusters.

The silhouette score, as shown in table 6.6, for the results are not directly comparable since the dataset is different for each result. A trend in the score is that the score decreases as the complexity increases. For result 4 where the difference between work and the interest was significant, the score is almost perfect at 0.951. It is however curious that the score increase from result 6 to 7, but the data in result 6 was not very good since contexts with similar data was split up.

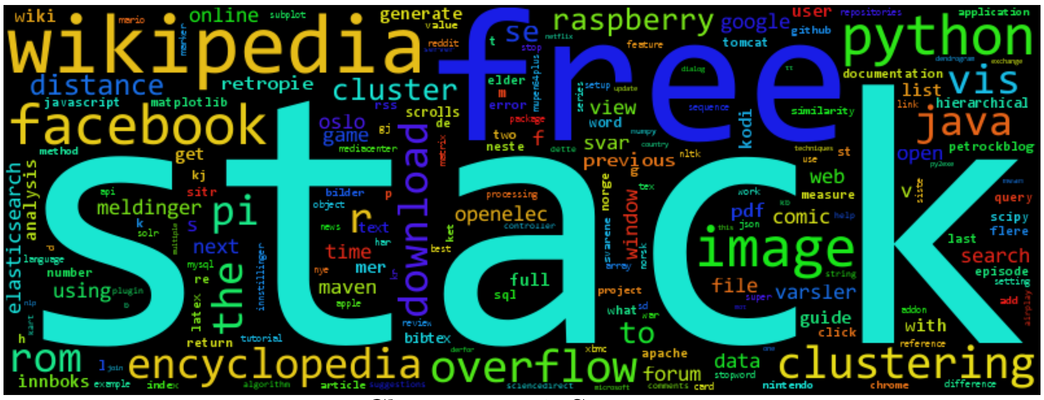
6.4 Different linkage

This section present and evaluate the results from using different linkage in the hierarchical clustering algorithm. Previously average linkage was chosen as the best linkage based on evaluation of their properties and preliminary testing. The other alternatives evaluated here is single, complete and weighted average linkage. This section will evaluate if that choice was correct. The results presented here are from using the full user data.

6.4.1 Results

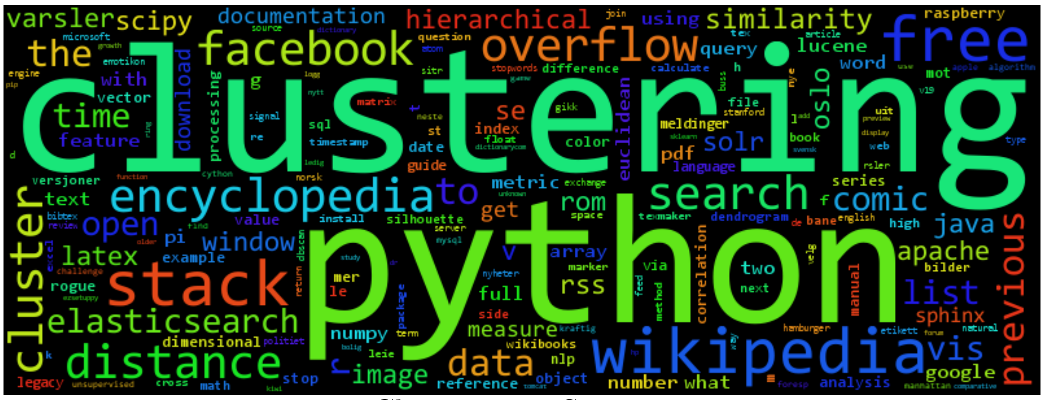
The results listed here have a result number and a title describing the linkage. Each result contain a word cloud for each cluster. At the end the silhouette scores is listed in table 6.7.

Result 8: Single linkage

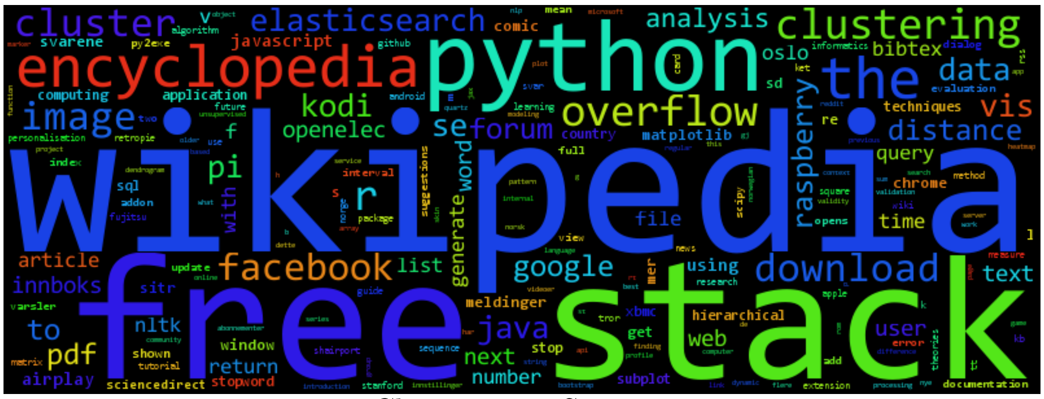


Cluster 1: 197 Sessions

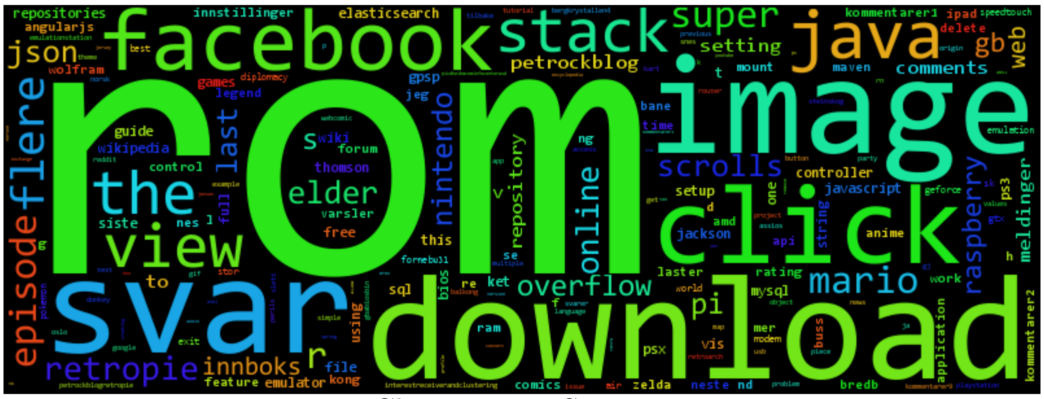
Result 9: Complete linkage



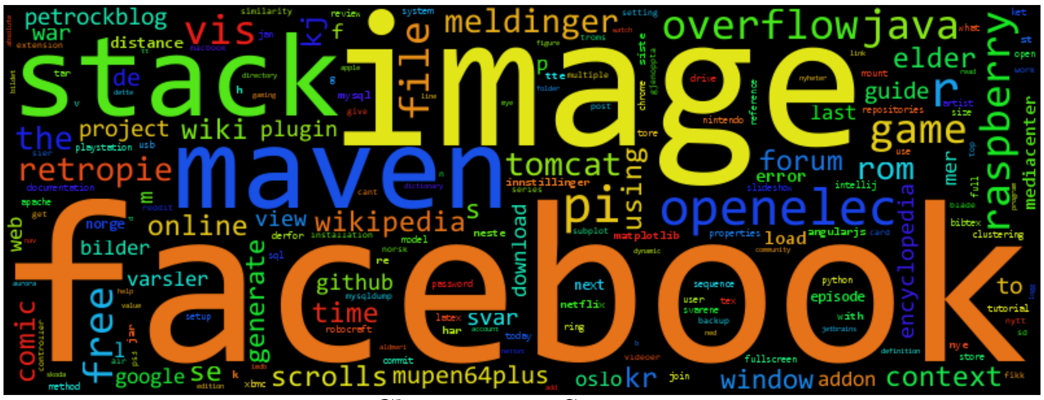
Cluster 1: 28 Sessions



Cluster 2: 70 Sessions

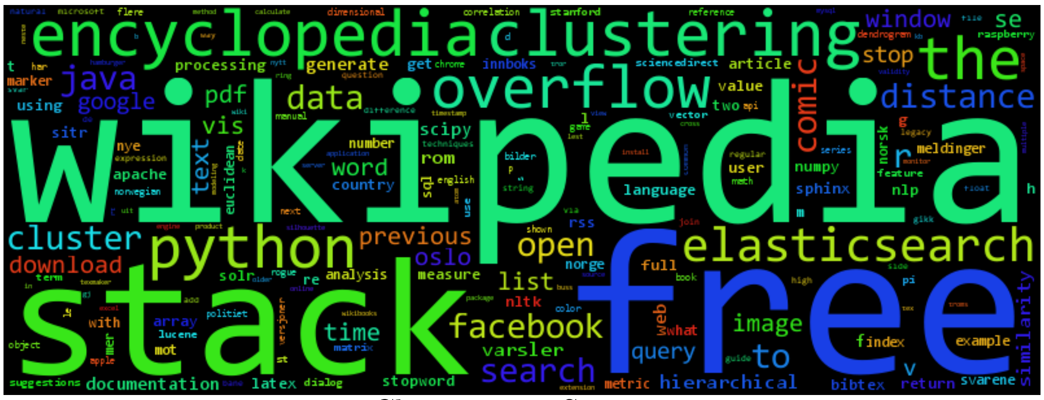


Cluster 3: 34 Sessions

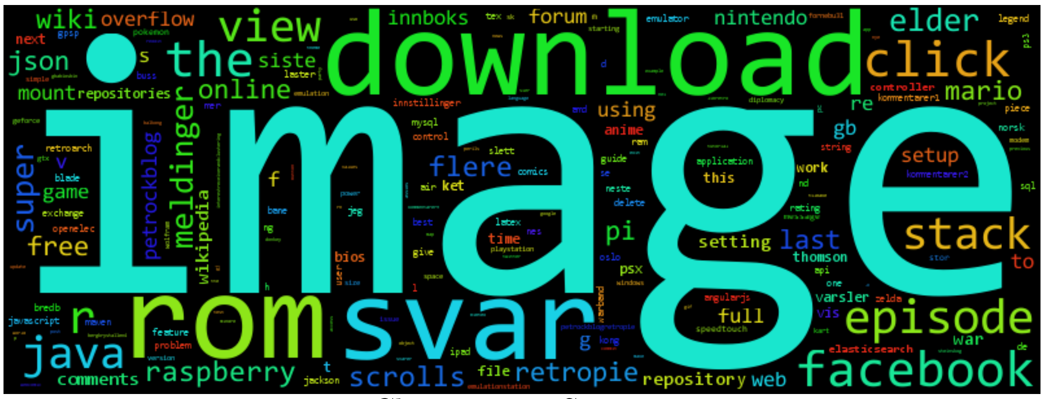


Cluster 4: 69 Sessions

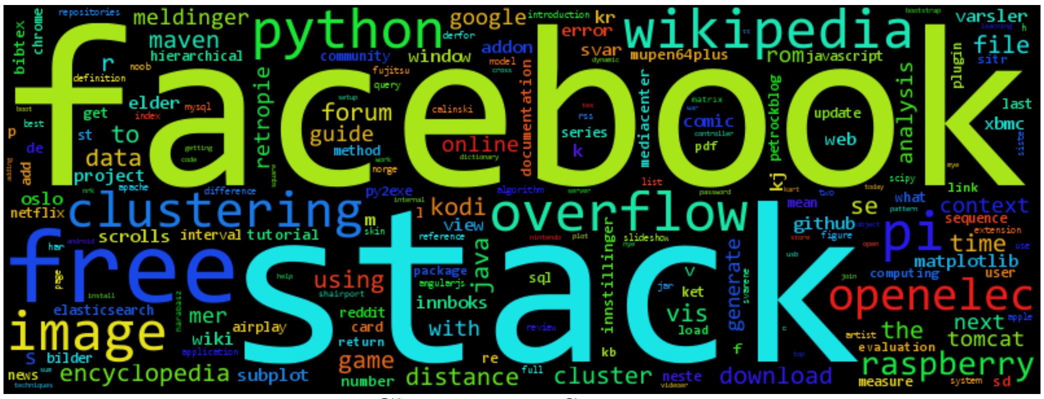
Result 10: Weighted average linkage



Cluster 1: 56 Sessions



Cluster 2: 52 Sessions



Cluster 3: 93 Sessions

Result	Silhouette score
Original: Average	0.430
Result 8: Single	-0.213
Result 9: Complete	0.269
Result 10: Weighted average	0.389

Table 6.7: Silhouette score for clusters with different linkage

6.4.2 Evaluation

Using different linkage in the clustering algorithm as presented in result 8, 9 and 10 with single, complete and weighted average linkage offer no improvement based on the results.

Single linkage

As expected from the discussion in subsection 4.3.2, the single linkage combine all sessions into one cluster except for some outliers that are excluded in the results. This is not a good result in any way as it is obvious more than one context.

Complete linkage

Complete linkage, shown in result 9, provide four clusters which looks like the results from average linkage (result 3) except with both the work context and the spare time interests context split in two clusters each. Where cluster 1 and 2 cover the work context and cluster 3 and 4 cover the spare time interests context. There are differences between the clusters representing the broad context of spare time interests. For instance in cluster 3 the terms ”rom”, ”download”, ”super”, ”nintendo” and ”mario” is much more prominent then cluster 4, and these terms are related to the context of old console emulation. Cluster 3 and 4 still share a lot of terms such as both having ”raspberry”, ”pi”, ”elder”, ”scrolls”, ”online”, ”retropie”, ”petrockblog” and ”java”. The terms ”retropie” and ”petrockblog” is related to old console emulation, so this context is not exclusive to cluster 3 either. The term ”java” is still fairly prominent in both cluster 3 and 4 so they still contain terms from the work context. Cluster 4 have the term "maven" as well, which is a new addition of work context terms added to the spare time interest context cluster. So to some degree the split of the spare time interests context has advantages, but also disadvantages. Cluster 1 and 2 seem very similar based on the word clouds they produce, and I see no advantage of there being two clusters instead of one as in average linkage.

Weighted average

For weighted average linkage the result is similar to average linkage, except for a third cluster added which is a combination of both the work context and the spare time interests context. There is no advantage of having cluster size weighting when trying to identify different contexts in the data, this would only result in more mixture of context between clusters.

Silhouette score

The silhouette score for the results can be found in table 6.7. The scores in this table are all based on results from the same dataset, so the scores are directly comparable to each other. The original result which is with average linkage has the highest score, and support my evaluation that none of the other linkages was an improvement. The silhouette score is however not necessarily the best tool to judge this, since I thought the complete linkage was better than the weighted average, but this is not supported by the silhouette score.

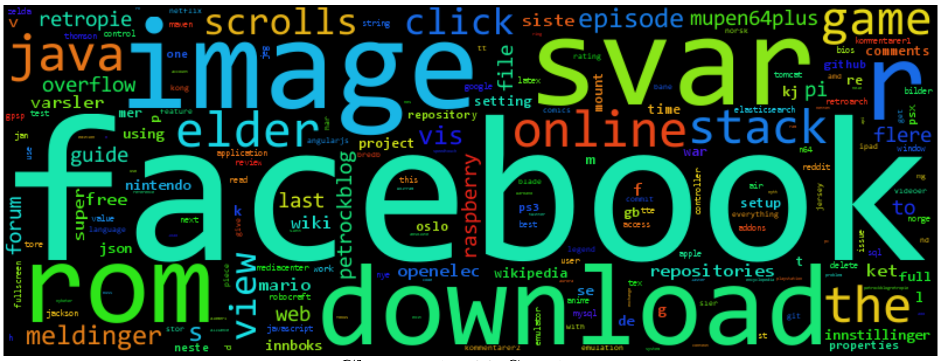
6.5 Higher resolution

This section will present and evaluate the results from increasing the number of clusters wanted out of the clustering algorithm. Originally the number of clusters are determined by the elbow point described in subsection 4.3.2. This sections will evaluate if the results improve with a higher number of clusters than the two clusters in the results from the user data. It will also evaluate of the contexts can be more accurate by increasing the number of clusters and thus the number of perceived contexts. The results presented here are from using the full user data.

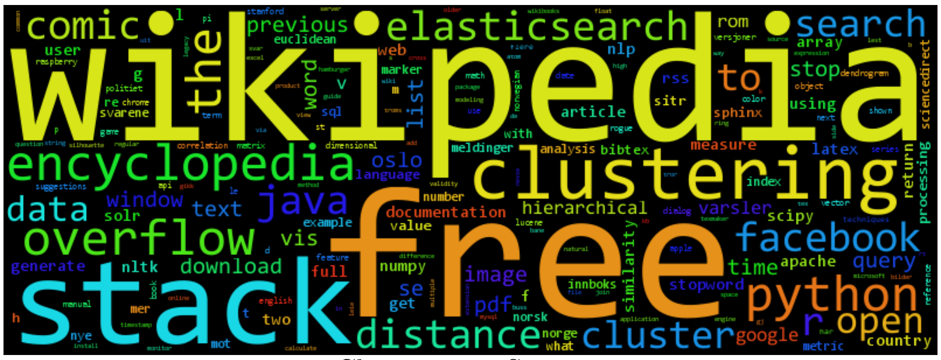
6.5.1 Results

The results listed here have a result number and a title for describing the number of clusters. Each result contain a word cloud for each cluster. At the end the silhouette scores is listed in table 6.8.

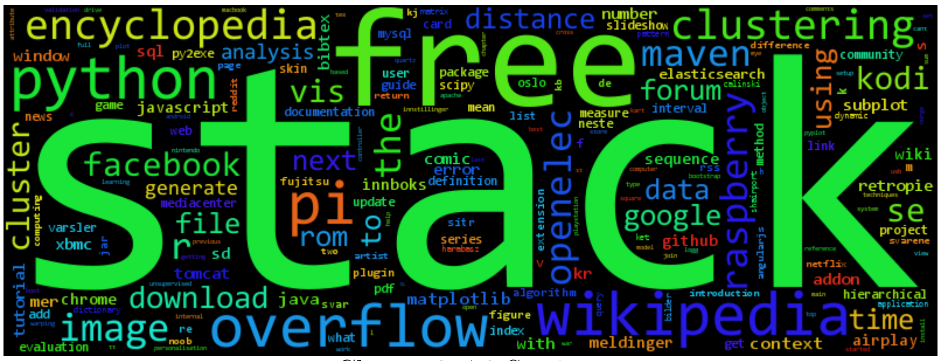
Result 11: 3 clusters



Cluster 1: 68 Sessions

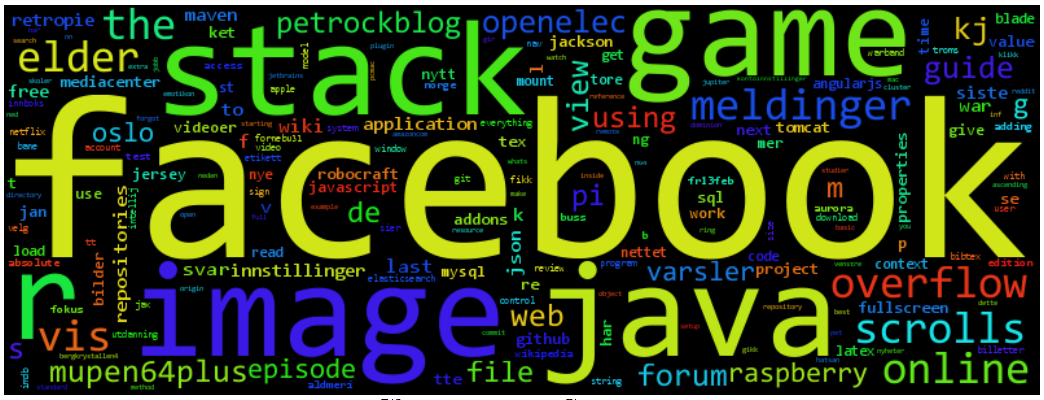


Cluster 2: 55 Sessions

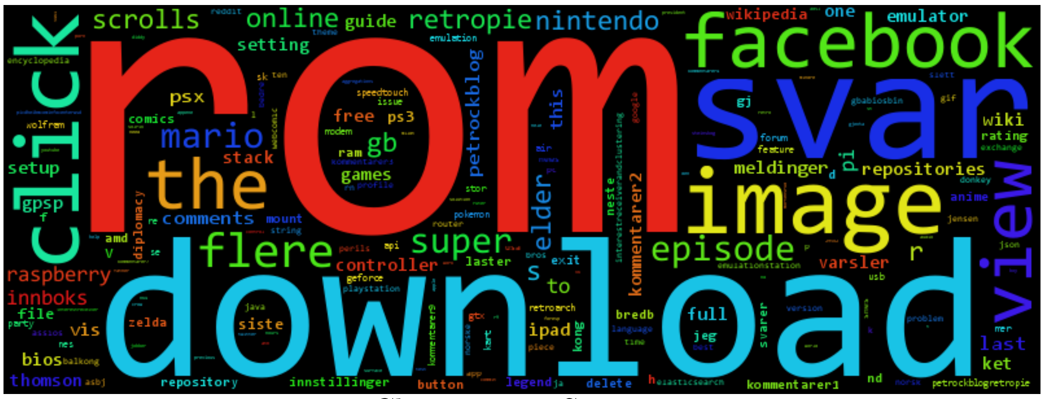


Cluster 3: 78 Sessions

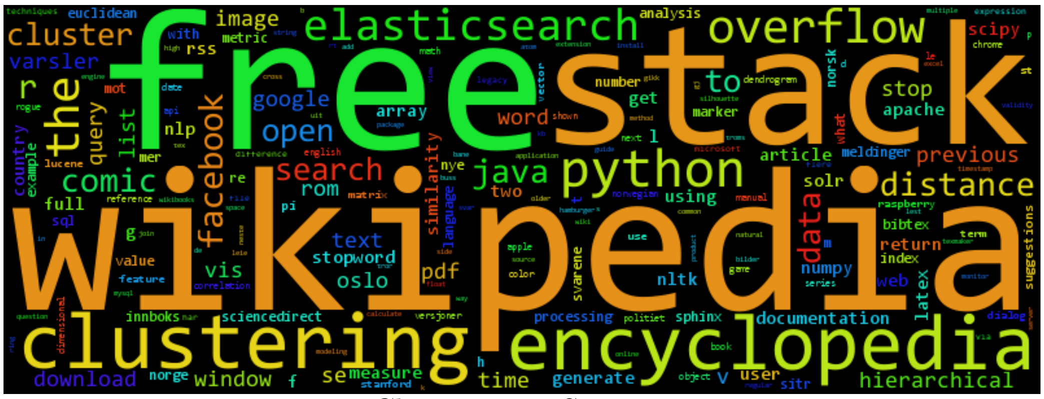
Result 12: 4 clusters



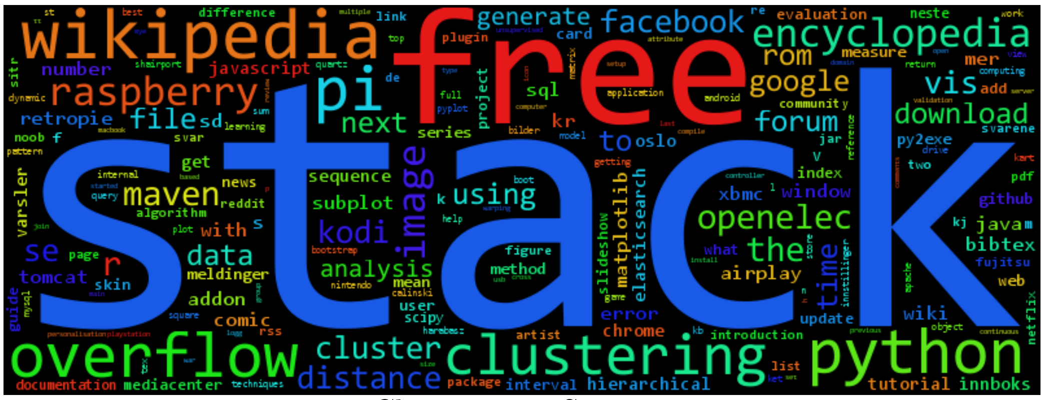
Cluster 1: 35 Sessions



Cluster 2: 33 Sessions

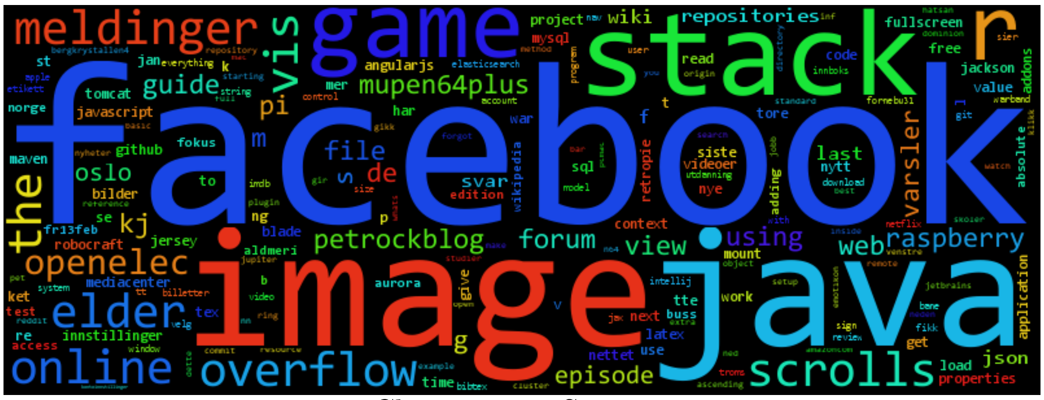


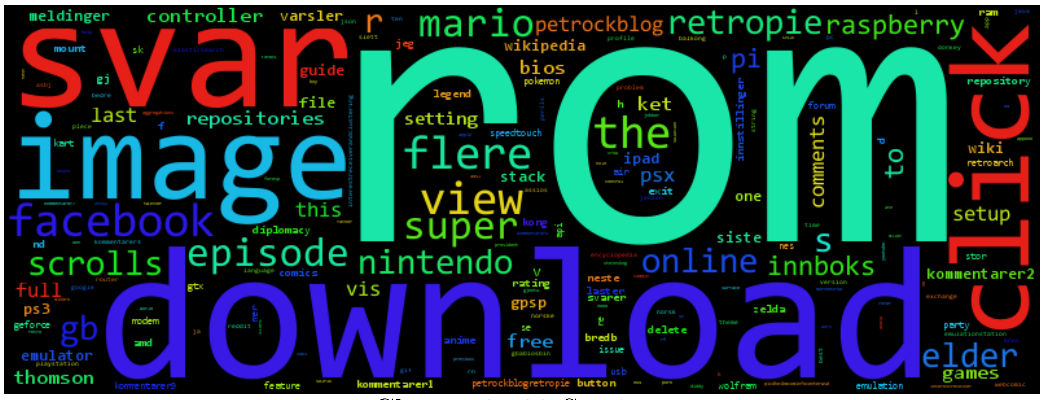
Cluster 3: 55 Sessions

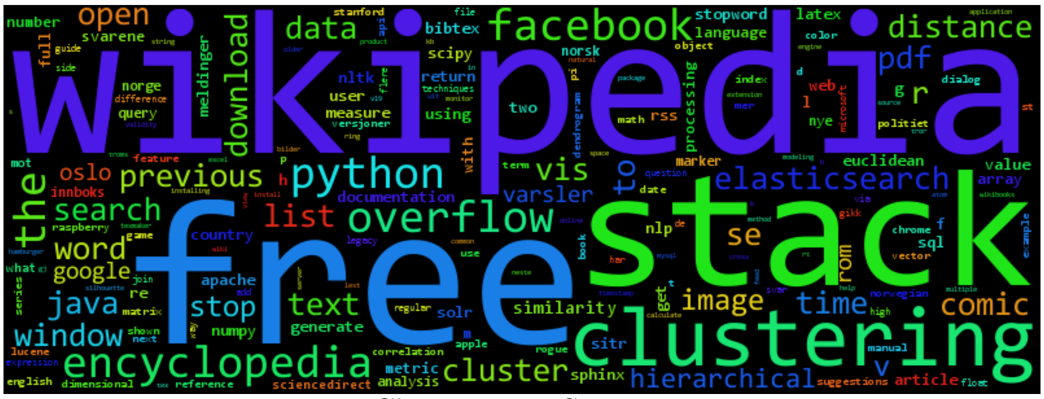


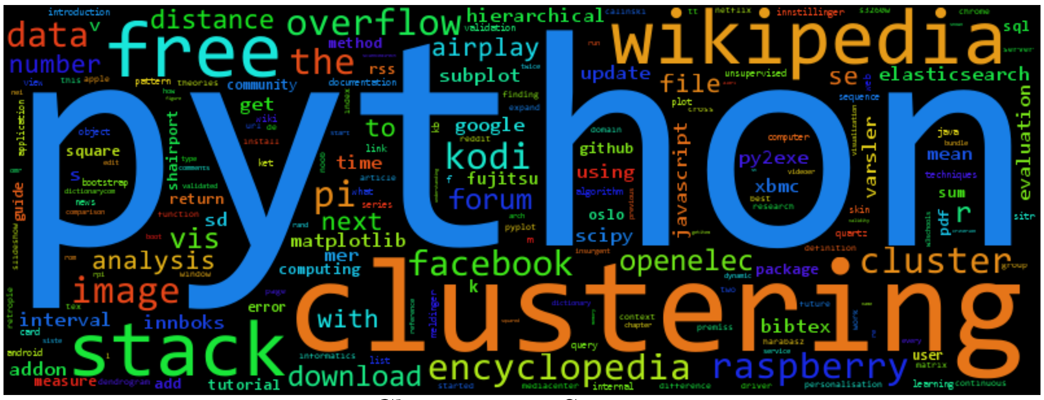
Cluster 4: 76 Sessions

Result 13: 5 clusters

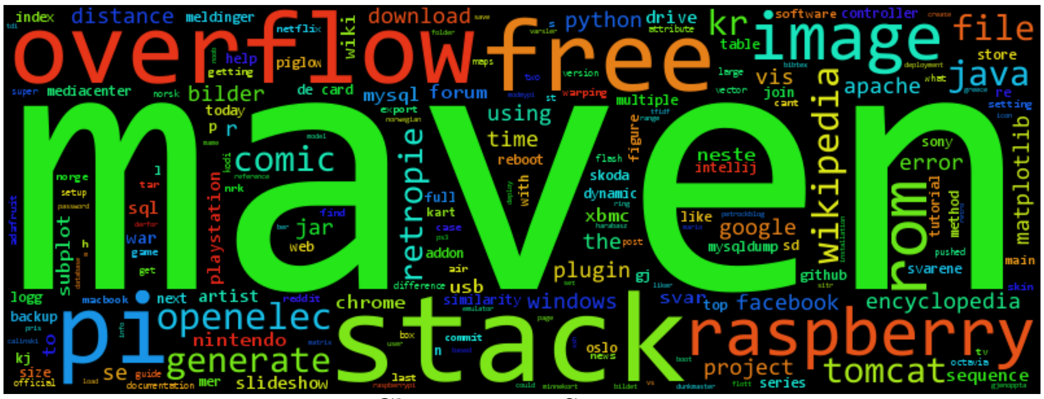

Cluster 1: 35 Sessions


Cluster 2: 32 Sessions


Cluster 3: 54 Sessions

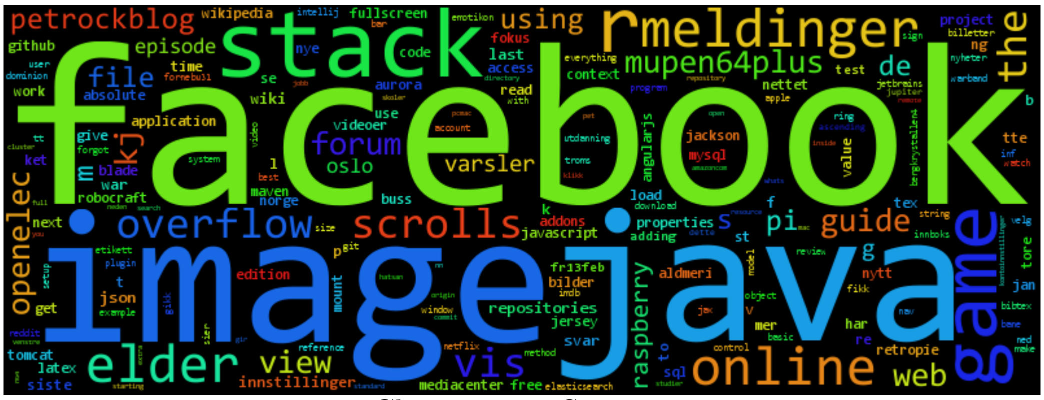


Cluster 4: 40 Sessions

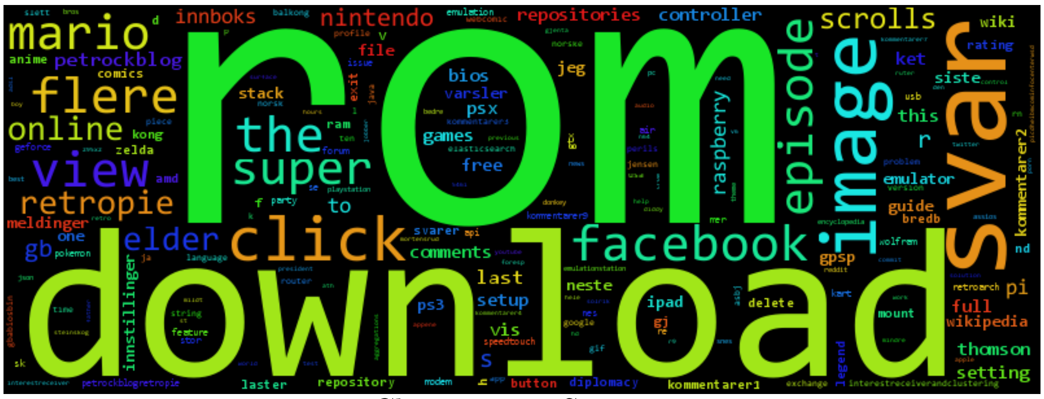


Cluster 5: 35 Sessions

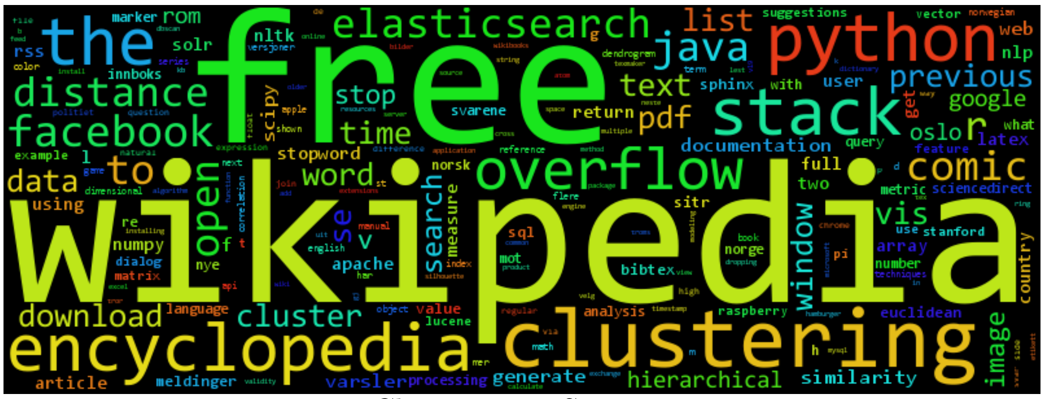
Result 14: 6 clusters



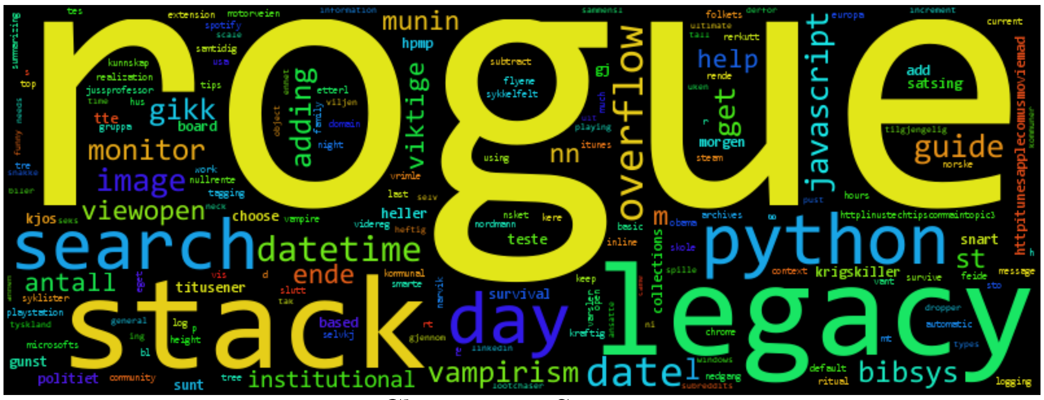
Cluster 1: 35 Sessions



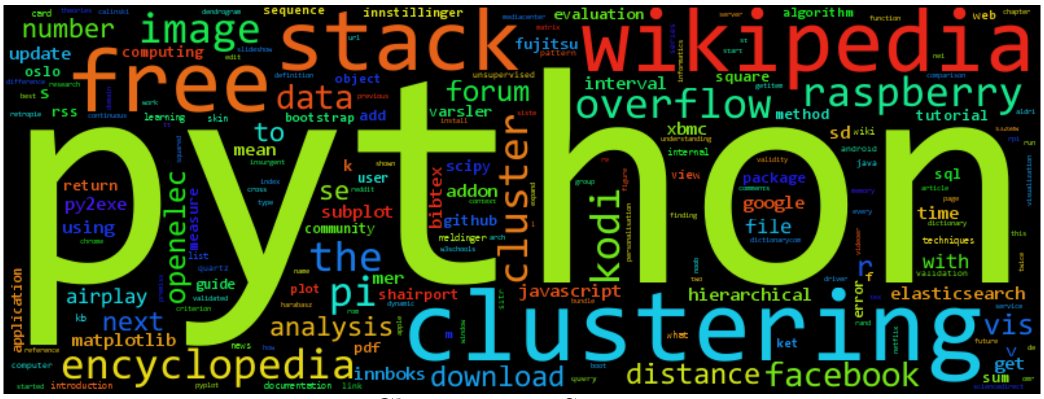
Cluster 2: 32 Sessions



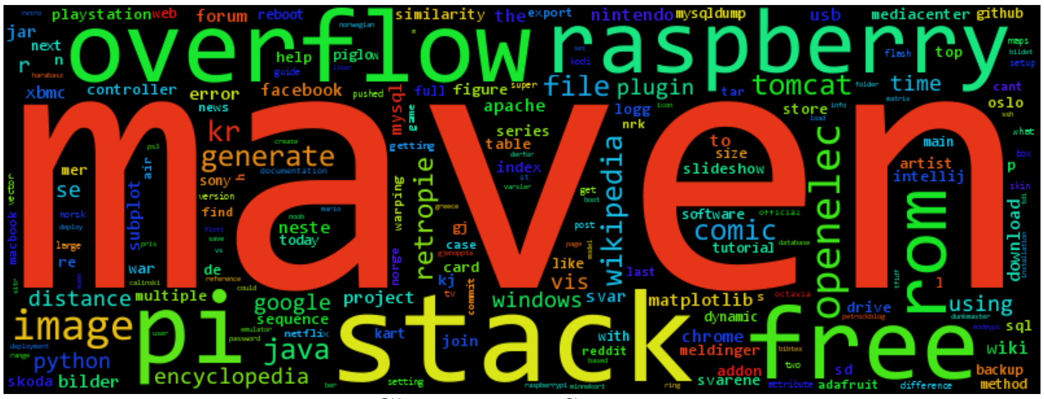
Cluster 3: 49 Sessions



Cluster 4: 5 Sessions



Cluster 5: 40 Sessions



Cluster 6: 35 Sessions

Result	**Silhouette score**
Original: 2 clusters	0.430
Result 11: 3 clusters	0.407
Result 12: 4 clusters	0.305
Result 13: 5 clusters	0.220
Result 14: 6 clusters	0.219

Table 6.8: Silhouette score for different number of clusters

6.5.2 Evaluation

The number of clusters in the results, especially on the user data, is low. The contexts they represent in the user data is very broad, with only work and spare time interests identified to a degree. Subsection 6.5 cover the results of increasing the number of clusters desired from the natural elbow point. This will tell how detailed the results can be, and if the system can produce more fine grained contexts by increasing the number of clusters. Result 11 to 14 cover the results from increasing the number of clusters from 2 in the original result with all user data, to 6 clusters.

Three clusters

In result 11 we can observe from the word clouds of the three clusters that the cluster representing the spare time interests is unchanged, with the same amount of session and the same terms. The other two clusters are the work context split in two. Both these clusters share most of their prominent terms such as "stack", "wikipedia", "free", "python", "clustering" and "overflow". The difference is that cluster 3 contain several raspberry pi related terms, while cluster 2 does not. The result is then that the work context in cluster 2 is much more accurate then before, while cluster 3 is more of a mix between work and spare time interests, while still being more related to work. This is not necessarily an improvement, but I think it is better to have two more accurate clusters and a mixed one, then to have two less accurate clusters.

Four clusters

Increasing the number of clusters to 4 as shown in result 12, split the previous spare time interest cluster into two clusters, while the clusters that was covering the work context stay relatively the same. The split up give similar results as when the work context split when the clusters was increased from 2 to 3, the resulting clusters have one accurate and one mixed. Cluster 1 contain the mixture between the work context and the spare time context with some of the most popular terms being ”facebook”, ”image”, ”stack”, ”java”, ”game”. While cluster 2’s most popular terms are ”rom”, ”download”, ”svar”, ”facebook” and ”image”. The results from using 4 clusters are not great. Having one cluster for mixed context is acceptable to some degree, but having two clusters with mixed context is not a good result in my personal opinion. The result have increased in accuracy in two clusters, but created two clusters with very low accuracy at the same time. Increasing the number of clusters to 4 have also not provided any new contexts.

Five clusters

Increasing the number of clusters to 5 as presented in result 13 practically split what was cluster 4 in the previous result in to two smaller clusters. This means the previous work context is now divided into three separate clusters. There is a lot of overlapping terms between the three clusters, such as ”stack”, ”overflow”, ”wikipedia”, ”free” and ”encyclopedia”. These terms are all generic in the sense that they come from Wikipedia and Stack overflow, which are sites that contain a broad spectre of data, and can be related to a lot of topics. Ignoring these terms we can see from the word clouds for cluster 3 and 4 that they are pretty similar, with some of the most popular terms being ”clustering”, ”python”, ”distance” and ”facebook”. The cluster that distinguish itself is cluster 5 with the most popular terms being ”maven” which is hardly present in cluster 3 or 4. Both cluster 4 and 5 contain terms related to the raspberry pi, so this topic has not been more centralized than before, rather the opposite. So splitting the work context now into three clusters have resulted in some terms being only present in one or two of the clusters, but overall they share too many terms to be considered more fine grained results.

Six clusters and more

When the number of clusters are increased to six, as shown in result 14, the previous cluster 3 is split. The result is a new smaller cluster containing the terms "rogue" and "legacy" as the most popular. Rogue Legacy is a game I played in a period while the user data was collected. This cluster also contain a lot of terms that are totally unrelated to this game, such as "stack", "search", "python" and "overflow", so it is not entirely successful in identifying a context, but it is the closest yet.

Increasing the number of cluster further does in some cases create more specific clusters, but it also seemingly equally often split topics between more clusters. Increasing the number of clusters further therefore does not make the results any better in my opinion.

Silhouette score

The silhouette score decrease with the addition of more clusters in the results. As sees in table 6.8, the score decrease slightly when the number of clusters is increased to 3, but fall rapidly after that, only to flatten at the end. This indicate the results are best with the original number of clusters, and getting worse by increasing the count.

6.6 Summary

The results from using the user data indicate two very broad contexts being discovered; work and spare time interests. These contexts are represented by clusters that to a large degree contain terms related to the other context as well as the context it represent. The precision and quality of the clusters in the results is therefore not very good, and the results is at par with my bare minimum of expectations.

Evaluating the system with generated data result in the system managing to successfully identify both work and a separate interest, and work with two separate interests in the same time frame. When the system is tested with four interests with different digital locations in the same time frame the results are close to successful. The system identifies the contexts, but two of the four contexts are split into two clusters. When the system is tested with what can be described as random data the results are as expected, random.

Using different linkage in the clustering algorithm on the user data did not provide any better results. When the number of clusters was increased the results was mixed. Increasing the number of clusters from 2 to 3 gave a result that might be considered better, but overall increasing the number of clusters gave a worse result.

Chapter 7

Future work

In this chapter, some possible future work will be discussed.

The solution in this thesis collect user data in sessions, and these sessions can be up to 4.5 hours long. This is problematic in the sense that those 4.5 hours can contain a lot of different activities and contexts. The results achieved in chapter 6 are very broad and not very accurate. Sessions that include a lot of data and possibly contain different contexts can be a large contributor to this. I have learned throughout the project by being aware of my actions that context change much quicker than I initially thought. It would be very interesting to perform a similar clustering on user data collected without the use of sessions. With my approach this would limit the attributes to relative physical location, digital location and time.

The terms that are recorded is collected from the links that users click to navigate the web. The idea behind this is that the link will contain text that describe something of interest to the user, and therefore the user will click it. In reality, words present in links are often just a small part of describing the content. An example of this is web comics like xkcd[1] where links with the words ”prev” and ”next” are used to navigate the comics. It is implied that this link will provide a new xkcd comic, but all the system get is the term ”prev” not describing anything. Based on this, collecting the terms describing interests more intelligently by for example using NLP on the entire website to collect a description of the content should be explored further.

[1]`https://xkcd.com/`

Determining the weighting of the session attributes for the distance measure is done on a trial and error basis with a subjective evaluation, or in other words by manually evaluating the results by changing the weighting values and determine if the results improved or not by looking at the word clouds. This might be one of the weakest links in this project as the weighting has significant impact on the results. It would be interesting to explore the weighting further and in a more objective manner to see how much better the results could be. This could be done by having an advanced and comprehensive dataset with a known correct state and by looking at the precision and recall for the clusters automatically while changing the weighting. My trial and error approach is also performed on the user data collected and the user data might not be the best data to calibrate the weighting on, since it might favour setting that would give bad results on other data.

The recommender system has played a much less important role then initially expected, since it was replaced by word clouds to represent the user interests in the evaluation of the system. Despite this, it would be interesting to see the result of using a web crawler or similar techniques to gather content for the recommender system, instead of web feeds. This could expand the content recommended to anything available online, instead of just news articles made available through a select number of feed sources. Recommending more content than just news articles will potentially provide a much better representation of the users' interests.

The recommender system recommend articles without considering their age. This result in recommendations containing old news that are no longer relevant. A major improvement would be to make fresher articles more prioritized by the system. This would make the recommender system useful, and not only a proof of concept to see what results is most relevant. This would also reduce the amount of articles covering a single term, as the most popular terms tend to dominate in the recommendations based on the results.

Further improvement of the recommendation system should include automatically selecting the context of the user to get recommendations based on the current contextual information available. This would make the system fully automatic and smarter. It seems counter intuitive to make the system detect the users' contexts and not identify the context of the user when accessing the recommender system.

With the somewhat disappointing results from using hierarchical clustering as the clustering algorithm in the system, even though there is no indication that the algorithm is at fault, it would be interesting to explore the results gained from using the DBSCAN or k-means algorithm instead. It is quite possible that one or both of the algorithms would perform better than the hierarchical algorithm on the dataset available.

An interesting approach to identify the contexts in the user profiles would be to utilize the terms registered as an attribute as well. In this thesis it has been a conscious choice to only utilize the contextual data available, and not to use the keywords describing the interests to perform the identification of contexts. The terms describing users' interests could be utilized to reduce the number of clusters terms are present in, and to concentrate the occurrence of terms to specific clusters. This should contribute to clusters reflecting specific interests and in turn specific contexts to a larger degree than in the results gained in this thesis.

This thesis has not considered the performance aspect in the design of this solution, as it is a proof of concept. It would be very interesting to explore the performance requirements for a large scale system with clustering to dynamically detect contexts in the user profile data. It would also be interesting to look at the design of the solution in regard to performance, since I am sure there would be a lot done differently.

Chapter 8

Conclusion

In this thesis a system that collect and analyse user data to create user profiles, that reflect the contexts of the users, is developed and tested. The system use these user profiles to recommend content to users on a per-context basis.

The evaluation of the system is performed by collecting user data through a time period of 77 days, and then evaluating the clusters of recorded keywords created by the system. The user data collected is from a single user, the author of the thesis. The system is also evaluated with generated fictional user data to compensate for the lack of diversity in the user data. Further, the system is tested in regard to the linkage method chosen in the clustering algorithm. Finally the system is evaluated in regards to the level of details in the contexts identified.

The evaluation show that the system can to some extent detect the user's contexts from the data collected. The contexts detected are very broad and not very accurate. Despite this, the system demonstrate that it is possible to detect contexts with clustering algorithms on contextual data. The contexts detected are based on my evaluation "work" and "spare time interests". These two categories define the broadest contexts I have been operating under while collecting the user data.

The generated user data provide an overview of the capabilities of the system. It show in the evaluation that the system does not handle similar contexts well. This support the findings with the collected user data.

The evaluation of the system further show that based on testing with the collected user data the best linkage for the clustering algorithm is average linkage as originally used. It also show that increasing the number of clusters to detect more specific and detailed contexts do not work, at least not with the available test data. The results of this show that the system is not very accurate, and only handle broad contexts, as indicated by the previous results.

The goal of this thesis was to *design, implement and test a system that generate user profiles based on user interests and available contextual information. The user profiles shall reflect the users' contexts or situations and provide the users' interests based on context.* This goal is achieved, but not in a satisfactory degree. There is a lot of variables that have to be taken into account in the design of this system. Because of this there is a lot to try differently to improve the results. Future work have the potential to significantly improve the results.

Bibliography

[1] Sofiane Abbar, Mokrane Bouzeghoub, and Stéphane Lopez. Context-aware recommender systems: A service-oriented approach. In *VLDB PersDB workshop*, pages 1–6, 2009.

[2] Gregory D Abowd, Anind K Dey, Peter J Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In *Handheld and ubiquitous computing*, pages 304–307. Springer, 1999.

[3] Gediminas Adomavicius and Alexander Tuzhilin. Context-aware recommender systems. In *Recommender systems handbook*, pages 217–253. Springer, 2011.

[4] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data. *Data Mining and Knowledge Discovery*, 11(1):5–33, 2005.

[5] Davide Albanese, Roberto Visintainer, Stefano Merler, Samantha Riccadonna, Giuseppe Jurman, and Cesare Furlanello. mlpy: Machine learning python, 2012.

[6] Olatz Arbelaitz, Ibai Gurrutxaga, Javier Muguerza, Jesús M Pérez, and Iñigo Perona. An extensive comparative study of cluster validity indices. *Pattern Recognition*, 46(1):243–256, 2013.

[7] Donald J Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, 1994.

[8] Petter Brandtzag. *Big Data – for better or worse*, 2013 (Accessed March 5, 2015). URL: `http://www.sintef.no/home/corporate-news/Big-Data--for-better-or-worse/`.

[9] Sung-Hyuk Cha. Comprehensive survey on distance/similarity measures between probability density functions. *City*, 1(2):1, 2007.

[10] Annie Chen. Context-aware collaborative filtering system: Predicting the user's preference in the ubiquitous computing environment. In *Location-and Context-Awareness*, pages 244–253. Springer, 2005.

[11] Hsinchun Chen, Yi-Ming Chung, Marshall C Ramsey, and Christopher C Yang. A smart itsy bitsy spider for the web. *Journal of the American Society for Information Science, Special Issue on AI Techniques for Emerging Information Systems Applications*, 1998.

[12] Zheng Chen, Fan Lin, Huan Liu, Yin Liu, Wei-Ying Ma, and Liu Wenyin. User intention modeling in web applications using data mining. *World Wide Web*, 5(3):181–191, 2002.

[13] Wikimedia Commons. *Hierarchical clustering simple diagram*, 2009 (Accessed February 28, 2015). URL: `https://en.wikipedia.org/wiki/File:Hierarchical_clustering_simple_diagram.svg`.

[14] David Dearman and Jeffery S Pierce. It's on my other computer!: computing with multiple devices. In *Proceedings of the SIGCHI Conference on Human factors in Computing Systems*, pages 767–776. ACM, 2008.

[15] Hui Ding, Goce Trajcevski, Peter Scheuermann, Xiaoyue Wang, and Eamonn Keogh. Querying and mining of time series data: experimental comparison of representations and distance measures. *Proceedings of the VLDB Endowment*, 1(2):1542–1552, 2008.

[16] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[17] The Apache Software Foundation. *Comparison between k-means and DBSCAN clustering*, Edited, (Accessed March 3, 2015). URL: `https://commons.apache.org/proper/commons-math/images/userguide/cluster_comparison.png`.

[18] E Frias-Martinez, G Magoulas, S Chen, and R Macredie. Automated user modeling for personalized digital libraries. *International Journal of Information Management*, 26(3):234–248, 2006.

[19] Min Gao, Kecheng Liu, and Zhongfu Wu. Personalisation in web computing and informatics: Theories, techniques, applications, and future research. *Information Systems Frontiers*, 12(5):607–629, 2010.

[20] Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. On clustering validation techniques. *Journal of Intelligent Information Systems*, 17(2-3):107–145, 2001.

[21] Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. Cluster validity methods: part i. *ACM Sigmod Record*, 31(2):40–45, 2002.

[22] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[23] Eric Jones, Travis Oliphant, Pearu Peterson, et al. *SciPy: Open source scientific tools for Python*, 2001– (Accessed 2015-03-24). URL: `http://www.scipy.org/`.

[24] Amy K Karlson, Brian R Meyers, Andy Jacobs, Paul Johns, and Shaun K Kane. Working overtime: Patterns of smartphone and pc usage in the day of an information worker. In *Pervasive computing*, pages 398–405. Springer, 2009.

[25] Leonard Kaufman and Peter J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis.* Wiley-Interscience, 9th edition, March 1990.

[26] David J Ketchen and Christopher L Shook. The application of cluster analysis in strategic management research: an analysis and critique. *Strategic management journal*, 17(6):441–458, 1996.

[27] Hung-Jen Lai, Ting-Peng Liang, and Yi-Cheng Ku. Customized internet news services based on customer profiles. In *Proceedings of the 5th international conference on Electronic commerce*, pages 225–229. ACM, 2003.

[28] Ting-Peng Liang, Yung-Fang Yang, Deng-Neng Chen, and Yi-Cheng Ku. A semantic-expansion approach to personalized knowledge recommendation. *Decision Support Systems*, 45(3):401–412, 2008.

[29] Fan Lin, Liu Wenyin, Zheng Chen, Hongjiang Zhang, and Tang Long. User modeling for efficient use of multimedia files. In *Advances in Multimedia Information Processing—PCM 2001*, pages 182–189. Springer, 2001.

[30] Christopher D Manning and Hinrich Schütze. *Foundations of statistical natural language processing.* MIT press, 1999.

[31] Meinard Müller. Dynamic time warping. *Information retrieval for music and motion*, pages 69–84, 2007.

[32] You-Jin Park and Kun-Nyeong Chang. Individual and group behavior-based customer profile model for personalized product recommendation. *Expert Systems with Applications*, 36(2):1932–1939, 2009.

[33] François Petitjean, Alain Ketterlin, and Pierre Gançarski. A global averaging method for dynamic time warping, with applications to clustering. *Pattern Recognition*, 44(3):678–693, 2011.

[34] Eréndira Rendón, Itzel Abundez, Alejandra Arizmendi, and Elvia M Quiroz. Internal versus external cluster validation indexes. *International Journal of computers and communications*, 5(1):27–34, 2011.

[35] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[36] Hidekazu Sakagami and Tomonari Kamba. Learning personal preferences on online newspaper articles from user behaviors. *Computer Networks and ISDN Systems*, 29(8):1447–1455, 1997.

[37] Albrecht Schmidt, Michael Beigl, and Hans-W Gellersen. There is more to context than location. *Computers & Graphics*, 23(6):893–901, 1999.

[38] Michael Steinbach, Levent Ertöz, and Vipin Kumar. The challenges of clustering high dimensional data. In *New Directions in Statistical Physics*, pages 273–309. Springer, 2004.

[39] Pang Ning Tan, Kumar Steinbach, and Vipin Kumar. Data mining cluster analysis: Basic concepts and algorithms, 2006.

[40] Jie Tang, Limin Yao, Duo Zhang, and Jing Zhang. A combination approach to web user profiling. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 5(1):2, 2010.

[41] Elena Tsiporkova. *Dynamic Time Warping Algorithm for Gene Expression Time Series*, (Accessed March 15, 2015). URL: `http://www.psb.ugent.be/cbd/papers/gentxwarper/DTWAlgorithm.ppt`.

[42] Mark Van Setten, Stanislav Pokraev, and Johan Koolwaaij. Context-aware recommendations in the mobile tourist application compass. In *Adaptive hypermedia and adaptive web-based systems*, pages 235–244. Springer, 2004.