

UNIVERSITY OF SOUTHERN DENMARK

SOFTWARE ENGINEERING 6. SEMESTER

---

## Datamining and its use

---

*Author:*

Lasse Bjørn HANSEN  
Simon FLENSTED

*Supervisor:*

Jan Corfixen Sørensen SØRENSEN



UNIVERSITY OF SOUTHERN DENMARK

*A report submitted in fulfillment of the  
requirements  
of Software Engineering 6. semester  
at*

University of Southern Denmark  
TEK

May 7, 2017

*“Some quote”*

- Gruppe 3

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation . . . . .	1
1.1.1	Data mining . . . . .	1
1.1.2	Product recommendation . . . . .	1
<b>2</b>	<b>Problem statement</b>	<b>2</b>
2.1	Problem description . . . . .	2
2.2	Problem statement . . . . .	2
<b>3</b>	<b>Implementation</b>	<b>3</b>
3.1	Solution overview . . . . .	3
3.2	Initial data dump . . . . .	3
3.3	Data Transformation . . . . .	4
<b>A</b>	<b>API commands</b>	<b>5</b>
	<b>Bibliography</b>	<b>7</b>



# Chapter 1

## Introduction

### 1.1 Motivation

The amount of data being processed around the Internet and within big systems is continuously increasing. This data should be structured and modelled in a way that makes it easily accessible and easy to work with. Handling large amounts of data the right way can prove to be very useful, not only to the company who possess the data, but also to the end users of a product. To achieve this, the art of data mining is very useful. The company Struct A/S have provided a typical software engineering task where data mining will create the foundation. This report will address theoretical aspects about data mining, how it is done in practice and how the final results of the processed data can be put to use. [1]

#### 1.1.1 Data mining

Data mining has become a big part of modern software engineering. Lots of companies tends to store large amount of data without structure and order within the data. This results in a lot of useless data which is both ineffective and a waste of resources. With prober data mining, it is possible to make this useless data useful to the company and its end users. In this case the data contain valuable information about users visiting the websites created and hosted by Struct A/S. By processing the data properly, it can be used for product recommendation, among other things.(Find kilde på dette)

#### 1.1.2 Product recommendation

If an e-commerce company wants to increase its profit, there is no doubt that product recommendation is one of the better ways of increasing your profit. This was first made popular by the retail giant Amazon. If you can predict what sorts of products your costumer may find useful, additional sales becomes more frequent. Big data sets, like the one provided by Struct A/S, can make it possible to predict customer needs, if the data is processed properly.(Find kilde på dette)

## Chapter 2

# Problem statement

### 2.1 Problem description

The initial problem/challenge is given to us by the company Struct A/S and is described as follows:

When launching sites, whether it being regular websites or web shops, a lot of user activity is logged. We therefore have a large amount of data associated with each of our sites but do not currently use it.

In the future we would like to be able to use logged data to generate an insight into the user activity on our site and actively use this data to create a personalized experience for the users.

This project will handle the initial normalization of the data, storing it in a scalable way and utilizing the data to create features which add value for the company and the users. In order to achieve this, theory has to become implementation. Research is required in terms of data storing, data mining and recommendation algorithms. This research is implemented in the end system creating an API allowing Struct A/S to get useful information from the data such as the recommended products for a certain user. This API will be the final product and will utilize different technologies and algorithms.

### 2.2 Problem statement

The data we have been given is in a de-normalized format and the problem therefore comes with two challenges - normalizing the data and utilizing the data to create a personalized experience for the users.

This leads to the following research questions:

- How can you effectively normalize large amounts of data?
- How can you optimally store and access data in a scalable way?
- How can you utilize the data to generate useful features for the company and the end user?

## Chapter 3

# Implementation

### 3.1 Solution overview

The final solution consists of a *RESTful* API build with ASP.NET Core and a MongoDB *No-SQL* database. Both are hosted through Amazon Webservices in an EC2 container using Docker. The API consists of the commands seen in appendix A.

This section covers the process from the initial data delivery to the final solution in a chronological order.

### 3.2 Initial data dump

In the beginning of the project we were supplied with data from one of *Struct A/S* customers. This data was in the form of many SQL tables and most of the data was not relevant for creating product recommendations. The main tables used were the following:

- Visitor: A collection of every unique visitor who visited the customer's website, each visitor gets a unique identifier called UID. Contains 3,073,665 visitors.
- Profile: A collection of every users signed up at the website. Contains 3037 profiles.
- BehaviorData: A collection of every unique action performed by visitors on the website, for example when a visitor views a product a new row is made with the visitor's UID, the product UID and the timestamp. Contains 3,326,736 visitor actions.
- Order: Contains each profile's orders. Contains 5520 orders.

A snapshot of the visitor and behaviorData table can be seen in figure 3.1 and 3.2.




	Uid	UserAgent	BrowserName	BrowserVersion	IPaddress
1	6820EDD0-E6A6-4105-A078-0000127E7AE1	AdsBot-Google (+http://www.google.com/adsbot.html)	Unknown	0.0	66.249. 
2	4D9FC023-9034-4B93-96D8-000013AB1940	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.3...	Chrome	36.0	212.71. 
3	CC43266B-5E13-473E-A4F2-000019D2B065	Mozilla/5.0 (compatible; Googlebot/2.1; +http://ww...	Mozilla	0.0	66.249. 

FIGURE 3.1: Visitor table from the original data

	Type	Id	Userid	Timestamp	ActivityData
1	ProductView	31070	794ebe68-49c5-451c-ba80-226cdc0508f4	2016-09-30 20:04:01.5376346 +02:00	NULL
2	ProductView	31071	055bf3eb-a556-4672-9f65-5f64932e3973	2017-01-05 12:18:05.5065939 +01:00	NULL
3	ProductView	31071	06d3f1b4-8347-499d-9edd-71c058024cc1	2017-01-27 10:42:00.8031017 +01:00	NULL

FIGURE 3.2: Behavior data table from the original data

These tables contain all the pertinent information for creating product recommendations and can be utilized after a cleaning and structuring process. This process is described in the following sections.

### 3.3 Data Transformation

To begin the initial data transformation a data storing technology has to be selected. The technology chosen was *No-SQL*, specifically *MongoDB* the most popular No-SQL framework [4].

*No-SQL* is chosen because of the good fit for this project. The data demands are not clearly specified in the beginning and with No-SQL it is easy to add or remove data or even change the data types on the fly. No-SQL's denormalized format also allows for faster retrieval of a single item without having to do joins or complex SQL queries. Finally No-SQL is easier to scale across multiple servers and many engines have built in scaling functionalities [3] which can come in handy when multiple clients begin using the service.

A brief overview of the different terminology for SQL and No-SQL is given in table 3.1.

TABLE 3.1: SQL vs No-SQL terminology

SQL	No-SQL	Comment
Table	Collection	
Row	Document	A No-SQL document can contain more complex datatype compared to a row in SQL e.g arrays or other documents

Python was used to accomplish the early migration from SQL tables to MongoDB.





# Appendix A

## API commands

Function	URI	Example	Description
GET	recommendation/visitorUID/ numberOfRecommendations/ database	recommendation/AAF995AE-1DD0-41C6-898B-9CBEE884E553/5/Pandashop	Returns a JSON array of size numberOfRecommendations containing productUIDs which are the product recommendations for the specific visitor
PUT	visitor/visitorUID/database	visitor/AAF995AE-1DD0-41C6-898B-9CBEE884E553/Pandashop	Registers a new visitor with the database
PUT	product/productUID/description/productGroup/database	product/5352/Agreatproduct/5/Pandashop	Registers a new product along with its description and product group with the database
PUT	behavior/visitorUID/behaviorType/ItemID/database	behavior/AAF995AE-1DD0-41C6-898B-9CBEE884E553/ProductView/5352/Pandashop	Registers a new behavior for the specific visitor with the database
GET	Update/database/password	Update/pandashop/supersecretpassword	Builds the collaborative filter for the database
GET	Updatevisitorproducts/database/password	Updatevisitorproducts/pandashop/supersecretpassword	Updates the top products for all visitors
GET	calculateTop20/database/password	calculateTop20/pandashop/supersecretpassword	calculates the top 20 products in the last 30 days

# Bibliography

- [1] A. Trivedi. (Feb. 2014). Mapping relational databases and sql to mongodb, [Online]. Available: <https://code.tutsplus.com/articles/mapping-relational-databases-and-sql-to-mongodb--net-35650> (visited on 02/23/2017).
- [2] M. D. C. Bowen. (Jan. 2017). What is normalized vs. denormalized data?, [Online]. Available: <https://www.quora.com/What-is-normalized-vs-denormalized-data> (visited on 02/23/2017).
- [3] C. Buckler. (Sep. 2015). Sql vs nosql: The differences, [Online]. Available: <https://www.sitepoint.com/sql-vs-nosql-differences/> (visited on 02/23/2017).
- [4] S. IT. (Jul. 2017). Db-engines ranking, [Online]. Available: <https://db-engines.com/en/ranking> (visited on 07/05/2017).