DM565: Formal Languages and Data Processing

Innovation Project

https://github.com/frederikgram/innovation

by

CARL SEBASTIAN BØGEHOLM FOLKMANN
cafol19
FREDERIK GRAM KORTEGAARD
frkor19
LASSE USBECK ANDERSEN
lasan19

1 Introduction

We have created a product with the goal of providing regional data to companies, with its primary intent being a helpful tool for marketing. Our product is built as a web-service, consisting of various graphical elements such as a map that - currently - gives a birds-eye view of Denmark, with each Danish municipality clearly separated from one and other. The purpose of this map, is to serve as a heat-map, giving clear and visual feedback to the user of our service, which regions fit their given criteria the best, criteria in this context meaning information about the population of a region.

Conventionally, a business looking to expand into a new , whether it be as a physical or an online presence, would have to do a lot of manual labor, analyzing each region manually. Realistically, they would have to either limit themselves to only look at larger areas simply as reasoned by the large quantity of data that must be collected, compared, and weighted against each other, to determine where the ideal market for the business is

This is obviously not ideal, and that is where our service comes into play, by not only automating the regional analysis part of marketing, and at the same time drastically reducing the time-spent by employees, freeing them up to increase their business value by other means.

It is important to note that this project is not intended to be limited to municipalities, as it could easily be expanded to work on say, a city-by-city basis. However, we found that this was outside of the scope of a product prototype.

2 Business Model Canvas Analysis

2.1 Value Propositions

Using our product, we intend to deliver an easy-to-use and highly customize-able platform which allows customers to narrow down a list of geographical regions where their product might have the most success. The task of comparing and analysing marketing regions can quickly become unmanageable or rather infeasible for companies which do not have large well-funded marketing teams inside of their organisation.

The hardships of this problem is - in part - due the the statistical challenge that comes with analysing and comparing a large amount of regions and for each of them, an extensive quantity of different metrics. However, it can also be hard to evaluate and assess the quality of not only the data used, but also the resulting statistics. Our product proposes to solve these hardships for customers, and thus allow them to benefit from advanced market analysis tools without having the in-house knowledge or risks associated.

As market analysis is by no means a new concept, our value proposition does not rely on the *newness* of our solution, but rather that we allow our customers to *get the job done*, allowing them more free time to further increase value for their respective organisations.

2.2 Customers

The description of the customer segment we see as being our primary market, can be described as a niche market, small and medium companies that want to improve marketing capabilities. Here we especially foresee companies such as web-shops, as their marketing needs tend to be more specialized, and typically have a higher need for targeted-marketing. Examples of which could be web-shops selling carbon-neutral clothing, luxury goods or in general any product whose target audience is easier to define by a combination specific characteristics such as income, political orientation etc.

Our customer relationship is most accurately described by the *self-service* paradigm. We provide the tools necessary for companies looking to perform in-depth market analysis, trying to find a suitable environment to expand into, based on various statistics that they can then use with their own ideas to form a plan for expansion. This view of the customers fits with our idea of a automated web-service.

The value our product serves, does not in any foreseeable way result in a diversified customer-segment, however, we do believe that a limited and free version could create a lot of intrigue from data-curious people.

1. Awareness
We propose that the ideal channel from which we should market our product would, in the beginning, be direct marketing towards companies we deem fitting, this could either be in the form of cold-calling, or targeted advertisements. On a larger timescale, we believe that collaboration with other platforms, targeting the same customer-segment could be profitable, as customers typically like all-in-one solutions.

2. Evaluation
Our service can be used to make the challenging task of collecting a large quantity of recent and validated regional data possible, and the computational work to compare and evaluate this data, possible. There are a lot of individual components to consider for each of these areas, and without having examined each one, it would be very hard to come to any reasonable conclusions. Whereas our web-service allows a customer to perform this type of analysis automatically in very little time.

3. Purchase
A more detailed explanation of the financial details of our product can be found in the *finance* section, however, the general idea is that we use the concept of a freemium model, where the quantity, type, and quality of the data is based on your subscription plan.

4. Delivery
As previously mentioned, our product will be served as a web-service, allowing product-delivery cross-platform, requiring no installation nor any technical aptitude.

5. After sales
We want to provide direct customer support, having a live-chat function, in conjunction with a user-guide and a technical FAQ.

2.3 Infrastructure

The infrastructure consists of the key *partners*, *activities* and *resources*. Looking at our key partners, we've recognised the companies and organisations that provide the data as the most important one - without the data we wouldn't have a product. Examples of these key partners would be sites such as *statistikbanken.dk* and *noegletal.dk*.

Our *key activities* mainly require us to maintain the web page and ensure that our data is up-to-date, representing the *current* information regarding each search-able region, this requires not only frequent updates but also verification of some sort, to ensure the sanity of the data, as correctness is an important part of a value proposition. With the latter as a basis, it is clear that our *key resource* is our data.

2.4 Finances

The product will be relatively cheap to build and maintain. Most statistical data - for Denmark at least - are provided for free, by data-banks such as *statistikbanken* et. al. resulting in us having no costs from data acquisition.

As the products customer-base increases, the computational cost (in the form of cloud-computing costs) used for comparing and searching our data will also grow. Other than this, the primary costs that we will we will incur are those related to web-hosting, salaries and what might be spent on marketing the product.

On the *revenue* side of things, our intentions are to implement a freemium model, allowing free users only a few search-able criteria, and on a much smaller set of data. The premium aspect will consist of additional data sets, increasing the value of each search query performed, along with a higher priority regarding customer support.

We've discussed the possibilities of using the premium model as either a type of subscription that runs a fixed time period (usually monthly) or as a payment per-search. The general idea is that the customer should be able to perform the searches they need - without being worried about user-error such as selecting the wrong criteria. We recognise that this product is generally used by a company either a single time or rarely, as the need for company expansion probably is not one that comes up too often. At our current stage, we find that a pay-per-search model would be the ideal model, no matter if a subscription model might be more profitable, as we consider the subscription model as a deterrent to potential customers.

3 Prototype

3.1 Data Gathering & Pre-processing

Seeing as our product is highly *data-centric*, we spent the majority of the development-cycle on gathering and formatting data. We primarily sourced our data by downloading it from various data-banks as previously mentioned, however this could also have been done using their official APIs.

Having fetched the data, we now had to sanitize it (specifically, fixing different encodings) and merge it into a single data set. The latter of these two problems we solved by writing a pre-processor, whose logic is mainly generic, however a few of the data sets we gathered required some separate logic.

With this pre-processor, we were able to quickly change, modify and add new search-able criteria to our application, as it required no more than a single line of code and a properly formatted *.csv* file, as collected from our data sources.

Typically, the next step would be to setup a database, schedule automatic database migrations, possibly setting up an MVC (Model-View-Controller) based architecture for our system, but seeing as this was a prototype, we decided on a much lower level solution, a single monolithic JSON file. While none of us are under the impression that this is an efficient nor a reliable way to interface with data, it is a good way to quickly setup prototypes without spending too much time needlessly optimizing a product that might not end up being viable anyways.

Having now gathered and merged our data into a singular data set, we now wanted to increase its value by pre-computing various metadata, this is expanded upon during the *Implementation* section, but can be described as a way of back-loading the computational requirements of our web-service, resulting in better response-times for end-users.

3.2 Implementation

3.2.1 Language & Frameworks

Remembering the emphasis on the *prototyping* aspect of this project, we decided to greatly value the speed-of-development, as we tried to utilize the *RAD* (Rapid Application Development) methodology that we learnt in DM571: Software Engineering.

In addition to this, we also wanted to use a technology which we had previously used during our studies. The culmination of this was the decision to implement the web-service using Python, as it abstracts away a lot of the necessary ground work by way of its many included libraries, and had previously been introduced to us in DM561 and further expanded in DM571.

From here we had to chose between the many web-development frameworks that exists for Python, we decided on using Flask, as it boasts the ability to easily create small applications with a very low barrier-to-entry, allowing developers to forego most all configuration, and instead focus on the implementation of the product itself.

3.2.2 Sorting the Regions

The core of our product - the value we give customers - is the ability to sort regions based on data. Because of this, it is clear to see that the computations which are used to compare regions are vital to our application. There are two primary types of data in our data sets:

- **Nested-data**
  Data such as parliamentary votes are nested inside not only the county it's relevant to, but also segmented into a dictionary for each of the parliamentary parties represented in the election from which our data inherits. As this data, and data such as the distribution of genders, highest-achieved level of education et. al. had its distribution as a percentage appended to it, during our pre-processing stage.
  Because of these percentage values, no further computation is required to normalize the data.

- **Absolute data**
  Currently, a few criteria in our data set are formatted as absolute values, this can be seen in the *disponibelindkomst* (disposable income) field, which is the literal value of the county's average. While this is directly comparable, when combined with other criteria, it must be normalized. If this is not done, values such as - say 200.000 - would heavily skew the total difference calculated for each county, as a other values might have other ranges.
  To solve this, we stepped through our data set, keeping track of the global minimum and maximum value of this field, allowing us to normalize absolute values, using the following logic.

  ```
  normalized = (( value - minimum) * 100) / (maximum - minimum)
  ```

  This could be done as a part of the pre-processing, but we decided to do it on-the-fly.

Having now normalized all of the data in our data set, comparing each county was a simple matter of comparison the sum of all the differences calculated for each region with one another. Hereafter, a simple iteration over the sorted list of regions allows us to know which regions to highlight and show to the end-user.

3.2.3 Graphical User Interface

Criteria Selection

With the idea of having an ever-expanding library of search-able criteria, we wanted to create a generic method for serving a menu to the user, allowing them to enable, modify and invert any number of criteria that they want. This was solved using the *Jinja2* rendering engine used by Flask, allowing us to create loops and other logic inside of the HTML files being served by the application.

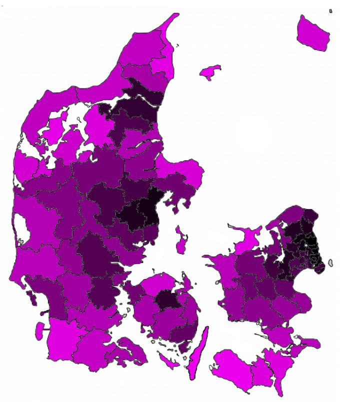
Performing this process starts by walking through the data set and collecting the key and path for each search-able criteria and storing them for future use. It should be noted that for our prototype specifically, we achieved a generic method for this, but had to extend it by using some hard-coded values, as this allowed us to skip a ton of logic without compromising the integrity of the prototype.

This would be an obvious place to start refactoring if the prototype was found good enough to warrant a production-ready version.

Visualizations

We implemented our graphical user interface as a one-page statically rendered website, displaying a heat-map over the supported regions, and a list of search-able criteria from which users can any number of, and chose to either change their influence on the search-results by using a slider, or invert the meaning of the criteria using a check-box. As to give users a less visual but copy-paste able output of the search-results, we also included a ordered list of the results on the web-page

The heat-map fills each region based on their index in the search-results, using a flood fill algorithm, taking basis in a coordinate given to each of the counties in the data set.
The color filled in becomes progressively darker the higher the region ranked in the search. RGB values for each region was calculated in order to always ensure a unique mapping no matter the number of regions represented.


** An example of the heat-map, when searching for the youngest average demographic of left-wing voters with a high disposable income*

While the design of the UI is simple, we feel that it is easy to understand and intuitive to use. This makes the navigation of the service much smoother as a user, and is in any case, enough to properly showcase the application.

4 Conclusion

As our final remarks, we conclude that we successfully developed a prototype application, allowing users to search and rank Danish municipalities based on various criteria which users are allowed to select and modify.

We feel that our product is very expandable, with many proposals for future optimization and refinement from within the group. As a result of our discussions both internally and with external parties, we believe that this product would be able to create real value for customers, and by way of our business model canvas and further analysis, we believe that we're equipped to take on that task, should it arise.

Lines of code:
- 270 for the web-service
- 107 for the pre-processor

in total: 377 lines of code.