PERSONALISING THE EDITORIAL MIX FOR A DIGITAL NEWSPAPER USING CONSTRAINT PROGRAMMING

MICHAEL LUNØE

Department of Informatics and Mathematical Modelling
Technical University of Denmark

August 2012

SUPERVISORS:
Michael Kai Petersen
Carsten Witt

[August 2012 – Technical University of Denmark]

ABSTRACT

This paper proposes a Constraint Programming (CP) approach to personalise a composition of articles that follows rules of the editorial mix in a digital newspaper. Inspiration from conventional newspapers will be used to express the problem as a Constraint Optimisation Problem and solved using local search for CP taking advantage of both probabilistic and logical approaches. Furthermore, this paper proposes a keyword based solution, using WordNet enrichment of articles in combination with a comparison of entities, to determine the relevance of an article to a user defined topic and similarity between articles. As a by-product of the implementation a library for solving personalisation problems using CP was developed.

RESUMÉ

Denne afhandling præsenterer Constraint Programming (CP) anvendt til at personalisere sammensætningen af artikler i en digital avis, der følger redaktionelle regler. Med inspiration hentet fra konventionelle aviser bliver de redaktionelle regler udtrykt som et Constraint Optimisation Problem og løst vha. local search teknikker for CP. Dette gør at både probalistiske og logiske fordele bliver udnyttet. Udover dette bliver en keyword-baseret løsning præsenteret til at udregne relevans af de enkelte artikler ift. brugerdefinerede emner, men også artiklerne imellem. Løsningen gør brug af WordNet til at berige artiklerne og en sammenligning af entiteter til den semantiske analyse. Som et bi-produkt af implementeringen blev der udviklet et bibliotek til at løse personaliseringsproblemer vha. CP.

ACKNOWLEDGEMENTS

I wish first of all to thank my family and friends for great support and for review of the paper. Without their support I would not have come as far as I hoped. I would also like to thank my counsellors, Michael Kai Petersen and Carsten Witt from the Department of Informatics and Mathematical Modelling at the Technical University of Denmark, for giving me room for my own definition of the project. They have both been a great aid in the process with guidance and review of the paper.

Finally, a special thanks to Jens for odd discussions and help in the process, Bo and Niklas for assistance with their knowledge, Ditte for the help and review and my brother and father for all the support.

PREFACE

Many digital newspapers present their content and navigation based on the scanning behaviour readers have on the web, but they do not consider the more in-depth reading tablet computers attract from their users.

This paper is the product of a personal need for a digital newspaper that is composed of articles that complement each other and provides a reading flow that matches the need for tablet computers.

The paper is produced under the Department of Informatics and Mathematical Modelling at the Technical University of Denmark and it will presume some knowledge of Constraint Programming as it will include an application that makes use of techniques from this field. The preconditions for the paper are the courses 02817 Personalization and Metadata Models and 02156 Formal Logical Systems taught at the Technical University of Denmark.

Kgs. Lyngby, August 2012

Michael Lunøe

CONTENTS

ACRONYMS

CP Constraint Programming: Constraint programming is a programming paradigm wherein relations between variables are stated in the form of constraints.

CSP Constraint Satisfaction Problem: Mathematical problems defined as a set of objects whose state must satisfy a number of constraints or limitations.

COP Constraint Optimisation Problem: Can be defined as a regular constraint satisfaction problem in which constraints are weighted and the goal is to find a solution maximizing the weight of satisfied constraints.

CPL Constraint Personalisation Library: A library that uses Constraint Programming to solve personalisation problems.

Part I

DEFINING THE PROBLEM

Discussing solutions to the problem of automatically generating the editorial mix and identifying rules that constitute the editorial mix.

1 INTRODUCTION

This chapter introduces the problem of creating a composition of articles that follow the principal of the editorial mix in a digital newspaper that supports the in-depth reading behaviour users have on tablet computers ([Tab]). Because conventional newspapers support this behaviour inspiration is sought from this field. Afterwards the paper introduces personalisation to features from the editorial mix and discusses the contribution of Constraint Programming as a technology in this context. The chapter concludes with a problem definition and a description of what this paper goes through in order to solve the problem.

Before it is possible to solve the problem of the editorial mix it needs to be defined. This is done in the following section.

1.1 THE EDITORIAL MIX PROBLEM

In the conventional newspapers the editors job is to compose an intriguing front page that offers the contents of the sections that might interest the individual user. His challenge is to accommodate the needs of the newspapers segment of readers, divide the articles into sections, with a nice reading flow and attractive illustrations, and hand-pick articles to go on the front page. In 1965 [Haskins] defines the editorial mix problem as finding the least number of items to obtain maximum audience coverage, but with adaptive web sites it is possible to obtain a single user's preferences and accommodate them. Therefore it is possible to redefine the problem of finding the personal editorial mix to:

> *Finding the composition of articles that provides the best satisfaction of the individual user preferences.*

An important part of the editorial mix is also that each piece of the mix needs to be interesting in it self, as stated by [Tidwell] in her definition of the editorial mix pattern. The personal editorial mix is from here on referred to as the editorial mix.

If we look at the personalisation part of the problem, [Perkowitz and Etzioni] decomposes the problem of synthesising an adapted page into several subproblems[1] :

[1] *In the subproblems stated here, "hyperlink" has been replaced by "item" to generalise them.*

- What is the content (that is, set of items) of the index page?
- Does it have a coherent topic? What should its title be?
- How are the items on the page ordered?
- How are the items labelled?
- Is the page consistent with the site's overall graphical style?
- Is it appropriate to add the page to the site? If so, where?

Some efforts have been made to digitally calculate similarities between articles and based on a current article suggest similar reading material or suggest articles based on other users' reading behaviour. Some papers propose a composition of articles from user picked RSS-feeds, which can, e.g. in the case of [Google Reader], be divided into sections. This comes close to conventional newspapers, but there is no ordering of the flow of articles. The ordering, flow and choice of relevant articles based on its content, is from here on referred to as using *relational* features for creating the editorial mix.

The solution for some digital newspapers is still to have an editor to create their coherent composed digital newspaper, like the New York Times or [Wired Magazine]. [Flipboard], on the other hand, compose their editorial mix of articles from feeds and divide their pages into three (or more rarely four or five) articles[2] with excerpts and images, much like conventional newspapers front pages. How they choose their composition is kept a business secret, but it does seem to vary a lot, see Figure 1.

[2] *Flipboard includes specialised layouts with more articles per page for Twitter content.*

It is hard to say if there is a control behind the placement of content other than the choice of featured and non-featured articles,

Figure 1: A screenshot of composition of three articles in Flipboard, with different subjects, i.e. world crime, world finance and technology news.

but this is actually an example of a computationally composed newspaper.

The placement and amount of room given for an article is from here on referred to as using *spatial* features for creating the editorial mix.

Finally, subjects of articles have more relevance at some points in time than others, and editors choose the amount of time stories should be available in, where RSS-readers just displays the newest articles first, which are not always the most relevant.

This selection of articles within a chosen time frame is from here on referred to as using *temporal* features for creating the editorial mix.

One thing that is vastly different from newspapers to RSS-readers is the ability to deliver personalised content, in that a user can choose which RSS-feeds to follow, whereas readers of newspapers need to navigate it in order to find interesting articles. Also, where newspapers have quality assurance of its content, RSS-readers have a seemingly unlimited amount of articles.

1.2 PERSONALISING A DIGITAL NEWSPAPER

[Bush] describes a collective memory library machine that can be indexed, called the Memex. Items in the library are linked together forming personal association trails. This is the early conception of the hypertext media that would later become the World Wide Web and later again personalised web applications. In many respects that is what this project tries to achieve; i.e. link information in the form of articles together and present them in personalised trails defined by the user. As opposed to [Bush] proposed manual linking, it is now possible to, e.g. classify and compute similarity automatically, which greatly aids the process.

User preferences are very diverse, and it is therefore hard to accommodate every individual in a single solution. A digital solution must be bound to a specific domain, but must also be open for novel use.

> *"Web personalization is defined as any action that adapts the information or services provided by a Web site to the needs of a particular user or a set of users, taking advantage of the knowledge gained from the users' navigational behaviour and individual interests, in combination with the content and the structure of the Web site."*
>
> – [Eirinaki and Vazirgiannis]

The three categories of the editorial mix, relational, spatial and temporal, can be described in the sense of personalisation as well. Accommodating individual user preferences based on *spatial personalisation* is achieved by a placement of articles, *temporal personalisation* by selecting articles of higher news value based on their relevance time frame and, finally, *relational personalisation* by selecting articles that provides more value based on their respective and collaborative topics. Temporal personalisation is also obtained by letting user preferences have a life time and decrease the preference influence on which articles to select as time passes. This is referred to as personalising using a "temporal user model", whereas the selection of articles based on their relevance time frame specifically is referred to as personalising "temporal user preferences".

All three categories are related, as they each provide some value to the editorial mix; a different spatial placement of a specific article can, e.g. provide a different composition of the editorial mix and therefore a different relational value to the user, which also means that the user will discover articles at different times and therefore also provide different temporal value.

This paper seeks a more general approach to solving personalisation problems, and tries to establish the contributions of Constraint Programming to this field.

1.3 CONTRIBUTIONS OF CONSTRAINT PROGRAMMING

As a declarative programming language, Constraint Programming (CP) offers means for describing the problem to be solved using constraints and a general purpose constraint solver. Once the general purpose solver is set up, the constraints can be defined to model the problem to be solved, but does not necessarily make it easy. However, the problem definition can easily be extended and modified afterwards.

The editorial mix, and personalisation problems in general, consists of a series of requirements on what should be shown to the user. These requirements changes according to the individual user, often just by an adjustment of general requirements. Because CP is a language for modelling requirements, in that it functions on a set of constraints, it can be a great contribution to this field. If requirements are modelled as logic constraints, the changes to the individual user could be done by adjusting variables and parameters of the given values to fit the individual user.

The editorial mix is a combinatorial optimisation problem, which is a special case of an optimisation problem – the difference being that combinatorial optimisation problems works on finite sets, whereas optimisation problems can work on infinite sets as well ([Schrijver]). Combinatorial optimisation problems enforce further constraints to the problem, because it should be possible to find the solution in the real world. In a personalised newspaper this would be that a combination of attributes for an

article should reflect that of a real article. It could be solved just by finding the combination of attributes that best fits the problem definition, but the solution could be a combination that is not possible to find in the real world. Because CP is good for combinatorial problems, this would also be a contribution.

1.4 PROBLEM DESCRIPTION

This paper attempts to create a system with the ability to automatically arrange personalised articles that complement each other in an editorial mix to attract more in-depth reading for tablet computers. Therefore the main hypothesis can be expressed as the following:

MAIN HYPOTHESIS

Is it possible to personalise the editorial mix of a digital newspaper?

The paper will be divided into two main areas; i.e. an application of a personal digital newspaper, where Constraint Programming (CP) is used to personalise the content and the composition of articles and an assessment of CP in the context of personalisation. Therefore this paper proposes CP as a technology for modelling this problem and solving it using a general purpose solver. With this technology a more elaborate control of the composition of articles in digital newspapers is introduced. Rules of the editorial mix and personalisation are utilised and applied. Many techniques for personalising digital solutions already exists, but the role of CP within this domain has not been determined. This paper seeks to explore CP as a tool for making personalised digital solutions.

1.4.1 *Personalisation Challenges*

In an attempt to introduce a personal editorial mix in the digital newspaper, this paper analyses the preferences of users with respect to the layout, navigation and information structure of the

application. It will seek inspiration from conventional newspapers to determine rules of composition and describe the search for articles to fit the user needs and the editorial mix as a Constraint Optimisation Problem (COP) and solve it.

1.4.2 *Algorithmic Challenges*

Because the problem has a fixed budget for finding a solution, algorithmic solutions will be discussed and a solution will be chosen and implemented. The findings will be concluded in an evaluation of the applicability of CP to personalisation problems.

1.5 THE STRUCTURE OF THE PAPER

In the following chapter (chapter 2) related work will be discussed and inspirations from these will be gathered. After this an analysis of the user needs and possible proposed use cases will be presented to derive requirements as features for the system in chapter 3. The chapter will analyse the editorial mix and conclude in a list of rules for the composition of articles. Chapters 4 through 6 will discuss and present the design and implementation of the solution. An evaluation of the solution is thereafter presented in chapter 7 and uses of it are discussed in chapter 8. Finally the paper will conclude if the main hypothesis can be verified and the value of CP for producing personalised solutions.

2 RELATED WORK

This chapter will look at the explored literature to find inspiration for the solution to be implemented. It discusses relevant potential solutions and concludes with a choice of approach.

Web personalisation is by [Mobasher] divided into phases of data collection and preprocessing, pattern discovery and evaluation, and applying the discovered knowledge in real-time to mediate between the user and the Web. There have been many suggestions on how to tackle these different processes of creating the interactive personalised digital newspaper. [de Buenaga Rodríguez *et al.*] proposes a strictly probabilistic approach to dynamic personalisation obtained by characterisation of content and user's interests. Both implicit and explicit relevance feedback[3] is used to refine the user models. Probabilistic approaches have the advantage of being effective, but often solves a very specific problem. Also, these approaches tend to get very complex in order to deliver promising results. Some cope with this by introducing logic to the problem like it is done in [Nilsson] with a spatial approach. In this project it is possible to benefit from the structure of the logic approach of CP and the effectiveness of a probabilistic approach by introducing preference constraints with an objective function.

[3] *Implicit is when the (unaware) user's behaviour is recorded to determine relevance, and explicit is when the user is aware of the action of giving the feedback.*

Many use the approach of computing the TF-IDF similarity. TF-IDF is weighting of words in a document represented by a Vector Space Model (VSM) as described in [Salton *et al.*]. When documents have been represented by VSM, their similarity can be determined using a cosine angle between them to compute a fast result. The method has been used in [Díaz and Gervs] to apply relational personalisation, where a set of keywords has been extracted from the news items to produce promising results, based on training with relevant documents. The use of TF-IDF constitutes the initial approach for computing similarity in this project. Later on a keyword based approach was implemented, which constitutes the final solution.

Classification techniques can also be applied in order to ease the task of selecting relevant articles and determine their mutual relationships. [de Buenaga Rodríguez *et al.*] uses a library of documents to train a categorisation algorithm, and the users are then asked to select categories of which they have interest. [Abuzir and Vandamme], on the other hand uses a thesaurus of hierarchically, and to the task specifically, structured terms to index news articles. Results of the indexing are thereafter mapped with user profiles to select the relevant articles. In the paper, [Abuzir and Vandamme] argues that semantic knowledge is more substantial than keywords. However, instead of using predefined root terms as the basis for a classification, [WordNet] can be used to obtain semantic knowledge for a document. WordNet is a large lexical database of English words and their relationships in the form of different graphs. [Bouras and Tsogkas] presents an algorithm for enriching articles using WordNet's hypernym-graphs. WordNet also contains similarity functions between words. These functions could be used to solve the problem proposed in this paper.

[Díaz and Gervs] does, however, present the means of combining the use of categories and keywords, but this approach demands predefined categories, which must be kept updated in order to follow semantic changes to the field. The time limitations and prioritisation of this project did not allow for a thesaurus to be obtained to aid the classification and will therefore not be introduced to the solution. One, could also argue that semantic assumptions are made, when categories are predefined, which could lead to some imprecise classification. [Bouras and Tsogkas]'s algorithm, on the other hand, is based only on words from the article and the general semantic (and more neutral) structure that constitutes the basis for WordNet.

To represent the users' interests, different approaches have been explored. The most promising results are generated by a short- and long-term representation of the user model as presented by [Díaz and Gervs] and [Mobasher]. [Díaz and Gervs] also propose a global user profile, to get the process of generating the user model started. It seems that this would be a viable approach.

[Díaz and Gervs] propose the use of collaborative filtering[4] to handle the problem of converging, which is what will happen if no non-personalised articles are introduced. Collaborative filtering, however, still only concerns articles that are within the area of the user's interest. If e.g. a user has not shown interest in politics, the news of Barack Obama becoming the President of USA will never be included in the newspaper. Instead a ratio between personalised and general articles will solve this issue, and since it is not within everyones interest to receive general news, this ratio should be adjustable.

[4] *Making automatic predictions (filtering) about the interests of a user by collecting preferences or taste information from many users (collaborating).*

There exist many different examples of preference modelling using CP, and [Abidi and Chong] describe a factual information system to find personal information, e.g. about healthcare. They present two constraints: (1) select only information-objects that correspond to the user-model and; (2) the content of the retained information-items do not contradict each other. This is an example of an editorial mix in that it incorporates the relational features, i.e. both between user and articles, and articles in between. However, they do not take into account the spatial part of their editorial mix, nor do they take into account any temporal features of the information needed. It is of cause notable that the user needs for a strictly factual information system are different than from a newspaper.

Another application of preference modelling using CP is [Vossen]. He proposes a CP approach to automatic playlist generation, which very much relates to what this project attempts to achieve. A playlist can, in this context, be perceived as a personal mix of songs. He presents constraints to exclude songs with certain attributes and constraints to model that certain songs should be similar to each other or a user preference. He also presents constraints to describe preference about the number of songs from a specific artist and finally, the well-known `all-diff` constraint. These can be directly translated to the editorial mix of a newspaper, where the songs are articles and an artist could be a specific author or content provider.

The presented relevant features can be summed up in the following list:

- fast computation using TF-IDF
- extension of similarity computation using a thesaurus
- WordNet enrichment of articles and extraction of keywords
- both a long-term and short-term representation of the user model
- collaborative filtering
- personal and general news in combination
- CP has been introduced to solve similar personalisation problems

This paper proposes a Constraint Programming (CP) approach to personalise the editorial mix. The problem will be expressed as a Constraint Optimisation Problem and solved using local search for CP taking advantage of both probabilistic and logical approaches. Furthermore, this paper proposes a keyword based solution, using WordNet enrichment of articles in combination with a comparison of entities, to determine the relevance of an article to a user defined topic and similarity between articles. A representation of the user model will not be chosen, as the focus lies with the composition of the newspaper. Instead, the representation of user needs will be defined manually to base the application on and make it ready for the implementation of the user model.

3 ANALYSIS

This section analyses the user preferences and identifies which features should be implemented in the application. It will also analyse and identify rules of the editorial mix of a conventional newspaper to be modelled as constraints in the application.

The focus of automatically generating the editorial mix introduces temporal, spatial and relational circumstances about the composition. But before these can be identified as constraints it is necessary to look at the user needs of the application.

It is also necessary to state that this paper will target the iPad 1 and 2 as its primary devices. The hand-held device also introduces mobility, which is also of great preference to the potential users and moreover, it attracts more in-depth reading of the articles. Furthermore, the research done by [Ihlström *et al.*] and [Ovesson and Wikström] determines the preferable size of the digital newspaper to be 14.732×20.828cm $\sim$ size A5, which reflects the size of the iPad.

3.1 USER NEEDS

This section will define the user needs for the application. A full description of personas, scenarios and business case is found in appendix A on page 93.

Because there are so many that reads an online newspaper of some kind, the user group is very large. An example of this is that a whole 17% of individuals, with no or low formal education, use the Internet for reading and downloading online newspapers and news magazines in 2011 ([Eurostat]). And that is the group with the lowest online reading activity. Even if only users of iPads are considered, it does not reduce the problem much as more and more users of tablet computers emerges.

This means that the application must accommodate many user needs, and it is therefore rewarding to focus on few, but very general needs to outline the objectives. This is done in the following.

From the scenarios it is clear that the users needs an overview of the content and that it should be presented in a digestible layout. This can be done through the front page, where the most interesting stories should be found. In addition only headlines, images and excerpts could be displayed. In order to make the users familiarise the application with conventional newspapers the application could build on a paged interface, from the front page (page 0) through sections until it reaches the back page. Page numbers could, furthermore, be used to keep the overview. From the scenarios it seems that both general and personal content is needed in order to satisfy users needs for information. However, the users may not necessary look for it. The amount of general versus personalised news is hard to define, but the user can be provided with functionality to adjust it. Also, reviews and opinionated articles could be of interest and maybe even puzzles and cartoons. In all cases it seems that the articles in the newspaper should be fresh and if the user makes changes in the personal settings, the newspaper should instantly update.

From the scenarios it is seen that users want to be social with their personal newspaper. This could be implemented both through a community in the application with comments on articles, but also by the possibility of sharing through common social networks. Notifications can be added to provide the user with the functionality of getting information on when there is a comment on articles. Articles that the user have already shown interest in or new articles on a subject the user follows.

Finally, it seems to make sense that the users provide their preferences about sections in keywords, which could also be gathered using relevance feedback. These models of the users can later be used to sell user behaviour patterns and very targeted ads.

Based on the personas, scenarios and business case the following general user needs have been derived:

- Get an easy overview of the content of the newspaper
- Easily navigate between articles with few touch-friendly interactions
- Read articles presented in a digestible layout
- Read relevant articles based on user defined topics

Furthermore, the editorial mix might not emerge as a need that users are aware of, but the rules of the editorial mix are established to better accommodate user needs in a composition of articles on tablet computers.

- Read articles in a composition based on the editorial mix

In order to work with it in the application this is added to the user needs and the rules of the editorial mix are derived in the final section of this chapter.

These user needs can be interpreted and fulfilled in many ways, e.g. to establish whether an article is relevant to a user is very individual and the way to apply these preferences on articles needs to be addressed. Therefore no design choices are made before the problem have been analysed on a deeper level.

3.2 USE CASES

This section presents use cases based on the user needs from the previous section.

Use case #1	Get an easy overview of the content of the newspaper
Description	It is possible to get an overview of the articles in the newspaper
Actors	User, web server
Scenario	1. The user opens the application 2. The application sends a request to the web server to fetch articles 3. From the articles the personal newspaper is composed 4. The user is presented an overview of articles contained in the newspaper
Extension	3a. There are not enough personal articles The newspaper is composed of less strict preferences or the newspaper is composed only from available articles

Table 1: Use case 1

Use case #2	Easily navigate between articles with few touch-friendly interactions
Description	It is possible to browse and navigate the articles in the application using few touch and conventional interactions
Actors	User
Scenario	1. The user gets an overview of the content of the newspaper as suggested in use case 1 2. The user navigates between articles using accessible menus of topic sections, headlines of articles and excerpts from articles 3. The user finds an article to read
Extension	3a. The user does not find an article to read The user searches for an article using the search bar

Table 2: Use case 2

Use case #3	Read articles presented in a nice and digestible layout
Description	It is possible to get articles displayed in a layout that serves the purpose of reading
Actors	User
Scenario	1. The user navigates the articles as suggested in use case 2 2. The user chooses an article to read 3. The chosen article is presented in a layout that easy to read

Table 3: Use case 3

Use case #4	Read relevant articles based on user defined topics
Description	It is possible for the user to read relevant articles of topics based on data gathered about user interests
Actors	User, web server
Scenario	1. The user uses the application regularly as suggested in use cases 1 to 3 2. The system gathers data about user interests 3. The system composes a newspaper of articles from the web server based on user interests
Extension	3a. The system does not have enough data on the user The system composes a newspaper of articles from a general model of a good newspaper

Table 4: Use case 4

Use case #5	Read articles in a composition based on the editorial mix
Description	It is possible for the user to read articles in a composition based on rules of the editorial mix
Actors	User
Scenario	1. The user uses the application as suggested in use case 1 to 4 2. Every composition of articles provided by the system follows rules of the editorial mix

Table 5: Use case 5

Notice that no specific assumption about the design was made in these use cases, but instead merely that some components, like menus, headlines and a representation of the user interests are present.

There are many ways to achieve the presented use cases, but the next section will draw from the presented use cases and the explored literature to derive requirements.

3.3 REQUIREMENTS

In the explored literature, scenarios and presented use cases expresses some non-functional requirements. These are described in this section.

User needs states the requirement of having a clear overview of the content and as stated in [Ihlström *et al.*], this includes a clear marking of the beginning and end of the articles and sections. This is obtained by both having a summary of the most interesting articles on the front page and by having a list of headlines in each section.

From the user needs it is also required that the system should be easily navigated and as stated by [Ovesson and Wikström], this should be through clickable sections, headlines and through pagination, or as the users from [Ihlström *et al.*] describes it; "open, turn pages, chose article, read and return". [Ihlström *et al.*] also states that the newspaper indexing is the most effective "navigational" tool in newspapers and headlines are the main entry points to text, which means that these should be very central in the application.

The layout, typography and design should be familiar to what is found in conventional newspapers, as stated by [Ihlström *et al.*] and [Åkesson *et al.*]. This is achieved by choosing a structure that resembles that of a newspaper and displaying content in balanced columns. In appendix A.4.2.1 on page 101 is found calculations on how many columns should be used on the iPad and on desktop computers based on the column sizes from conventional newspapers. The result of the calculations is that there should be 2 columns in portrait mode and 3 columns

in landscape and on desktop computers of 1200px in width. However, it only requires a screen of 1320px in width before 4 columns would be optimal. But as this project targets the iPad 1 and 2 screen sizes only 2 and 3 columns will be considered from here on.

The contents of the newspaper should consist of both personal and general news according to the scenarios. Furthermore, the typography of the application should also resemble that of a conventional newspaper. It should contain a good ratio of both graphical and textual content and should, when possible, supply multimedia content. The conclusions made from the empirical data in [Ihlström *et al.*], about exploring which features to bring from conventional to digital newspapers, was that valuation and position of the news was important. More importantly that the reader should be guided through the digital newspaper. It is, however, crucial to consider that their empirical basis is not very substantial. They have a qualitative selection of respondents from newspapers that have, recent to its execution, become dedicated to the project. Moreover, they have chosen 16 open questions for the respondents to answer, which should provide some sort of basis for their conclusions. In this project it is chosen to use them as guidelines, but the choices made on this basis must be verified[5]

[5] *Earlier statements from the paper have been backed by additional sources.*

That the reader should be guided through the newspaper using valuation of the items does, however, fall in line with the editorial mix, which also have been used in conventional newspapers for a long time. This suggests control of the temporal, spatial and relational values of individual articles and between them. Also, using the Gestalt principles [Tidwell] suggests providing a visual hierarchy so the user can see the relative importance of the page elements and the relationship among them.

Some technical requirements have also been gathered from the explored literature.

[Díaz and Gervs] suggests to use article excerpts in addition to the navigation using clearly marked sections and article headlines, and that these should be personalised. The results from [Ovesson and Wikström] suggests that the menu with clickable sections should be placed on the left side of the screen, but they could have been biased as it was already placed there in the

tested prototype. In, addition, they found this as a good choice as they recognised it from the web. A menu in the top of the page would therefore also be in line with their findings, as it is a general pattern of the web [Tidwell]. In addition, it would take up less space in the view, leaving more room for the general purpose of the application, namely reading. The menu items will work well as the user defines the content of them. Furthermore, it would aid the user to relate more to these divisions if it is possible for him to name them himself.

Furthermore, it should be possible for the user to get an overview of the headlines contained in a section. This could be done by just having a list of the headlines, or by using the overview plus detail pattern presented by [Tidwell].

Many articles discuss different ways of representing the user's interests. It seems, however, that both [Díaz and Gervs] and [Billsus and Pazzani] generate good results with a dynamic short-term user model in combination with a static long-term user profile.

Finally, the implementation of a community in the application should be done with the possibility of sharing the story on different social networks, but could also include comments on articles, as suggested in the scenarios.

These requirements can be summed up in the following list:

- Summary of few most relevant articles on the front page
- Overview of article headlines
- Clear section and article headlines and personalised article excerpts to ease navigation
- Paginated navigation of articles
- Personal and general news
- Layout, typography and design should be resemble that of a newspaper
- Visual hierarchy
- Balanced columns
- The reader should be guided through the newspaper using valuation of the items in terms of categories of the editorial mix
- Menu of sections should resemble those from the web
- Combination of long-term and short term interest model of the user
- Incorporate community and social networking

3.4 THE EDITORIAL MIX

The task at hand is to decompose the spatial, temporal and relational features of the composition of articles that provides the best satisfaction of the individual user preferences into constraints. This section analyses the existing literature on reading behaviour of conventional and digital newspaper and derives constraints that supports in-depth reading to compose the editorial mix of.

Users reading behaviour when reading conventional newspapers differs from when reading digital newspapers. In the experiments done in [Holmqvist *et al.*] it is concluded that the net paper[6] readers read stories thematically close to their own specific profession or interests. So it is important to provide this setting for the reader. The readers also used the front page as a

[6] *Newspapers on the Internet.*

provider of main entry points. And finally, the readers "claim to scan more in order to find the two or three stories they will read in the net paper" which is explained by the poorer chances of links catching reader interest. However, it could be possible to attract the reading behaviour from conventional newspapers onto digital platforms – it is certainly interesting to see an equal analysis of the reading behaviour of tablet computers, which calls for more in-depth reading ([Tab]). If digital tablet platforms are to attract more in-depth reading it requires some flow in the presentation of the articles, i.e. it requires an editorial mix, so the readers do not feel like they have left the main trail, as explained by the users from [Holmqvist *et al.*].

To attract reading behaviour of conventional newspapers it is worthwhile understanding readers expectations of these. [Holsanova *et al.*] confirms a summary of reading behaviour assumptions from [Kress and Van Leeuwen] on conventional newspapers using eye-tracking measurements:

- Readers prefer the most general information at the top and the most specific information at the bottom of the semiotic space
- Readers look for the most important information in the centre of the page and less important information on the periphery
- Readers look for paratexts[7]

[7] *[Genette] defines paratext as those productions accompanying a text, such as an author's name, a title, a preface, or illustrations.*

And, two are not confirmed, but not declined either:

- Readers look for graphically salient elements; however, it is important to bear in mind that 'what is made salient is culturally determined' [Kress and Van Leeuwen]
- Readers follow elements connected to each other by framing devices such as lines and arrows

That the most important information should be in the centre of the page will be hard to attract on digital platforms because of the limited space. Because of the screen size only one or two, and in some cases three, articles are shown at a time and the user will have to scroll to see the next items.

However, the relation between adjacent articles can still be controlled, so a featured[8] article should be adjacent to some smaller, but still very relevant articles. This will hopefully attract the same behaviour, but of course needs to be confirmed.

[8] *A featured article means providing it with more space than others, a central position and it is often accompanied with graphically salient elements.*

Also, a central position is hard to obtain, as many articles will be listed below each other, so a central position is here deemed to be higher than its relevant non-featured articles.

Based on the user study and the explored literature on reading behaviour the editorial mix problem can be divided in two; (1) the front page and (2) the sections.

1 The purpose of the front page is to draw attention and provide an intriguing overview of the whole newspaper. This is done by using many images and providing headlines and excerpts of the most relevant articles of the newspaper. The most relevant article should be featured in the centre with a selection of a little less, but still very relevant articles adjacent to it. A visual hierarchy should be provided so the user can see the relative importance of the page elements and the relationship among them. The front page should, if available, provide interesting articles from all sections as main entry points.

2 The purpose of each section is to provide a flow of articles relevant to a, by the user provided, topic that keeps the user interested and invites for in-depth reading. More general articles should be placed in the top of the screen and more specific at the bottom, with a featured main article in the centre. Framing and lines should guide the user to what is related.

These descriptions can be decomposed into the following constraints.

GENERAL CONSTRAINTS

- A featured article should be allowed to take up more space
- A featured article should be accompanied by an image
- A featured article should have a central position
- A non-featured article should take up less space
- A featured article should be adjacent to non-featured articles
- All articles should be different

FRONT PAGE CONSTRAINTS

- Every article should have a very high level of relevance to at least one of the section topics
- Most or every non-featured article should be accompanied by an image

SECTION CONSTRAINTS

- Every article should have a high level of relevance to its containing section topic
- A section should contain an article if the front page contains the article and its relevance to this section is highest
- Articles should be grouped into subjects
- The section should contain a balanced weight between graphical and textual content
- Images should be spread evenly in the section

The following chapter will discuss and present the choices in design of the application.

Part II

THE PROPOSED SOLUTION

Discussing choices and presenting the solution.

4 DESIGN

This section discusses different choices in the interface for a digital newspaper and how the application can provide the user with a personal editorial mix.

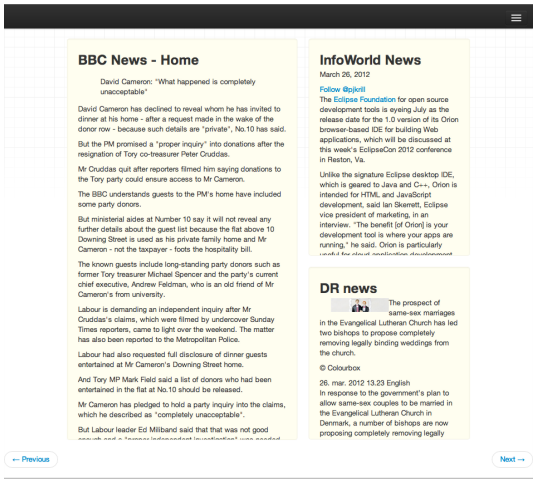
Before it is possible express the editorial mix problem formally the interface of the application must be described, because the constraints will have to make some assumptions about the application structure. The two following sections will describe the layout and typography of the application followed by the interaction with it.

4.1 LAYOUT AND TYPOGRAPHY

This section describes iterations of the design in terms of layout and typography and the preliminary work to base design choices on.

After the definition of the initial requirements was done the first prototype was developed. Its main features followed the requirements on turning pages, choosing an article, read it and returning to the overview of articles, see Figure 2a. And after a small preliminary survey it became clear, that a column-based layout showing full articles would be more attractive, and would provide a better opportunity to explore the editorial mix. With a column-based layout the digital newspaper would have more resemblance to conventional newspapers and therefore it was possible to apply some of the same principles of the editorial mix. This choice removed the paginated layout and instead introduced a scrollable layout to have enough space for the articles to fit in.

Figure 2 shows three iterations of the prototype design, which were based on the derived requirements. The third iteration



(a) Initial prototype layout with adjustable ratios between articles and a paginated interface of each section.

(b) Second iteration of the prototype with an scrollable layout. Sections are placed beneath each other.

(c) Third iteration of the prototype with a column-based and scrollable layout. Sections are placed beneath each other.

(d) Third iteration of the prototype with a column-based and scrollable layout. Sections are placed beneath each other.

Figure 2: The figure shows three iterations of the prototype layout.

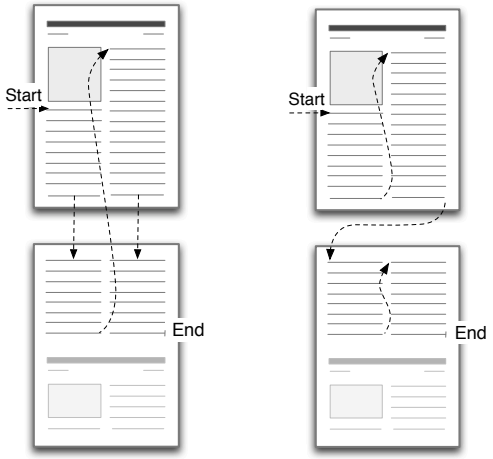
of the prototype (Figure 2c and 2d) was used as the foundation for user tests. The prototype consisted of the basic navigation between topic categories, i.e. sections, and articles. Navigational choices was made in order to present the general idea of the framework, but where more crucial choices on its uses have not yet been made. This was also to encourage the test subjects to talk about what uses they would have of the presented framework. However, they were also asked about the navigational structure and indeed some changes had to be done. A specification of the test can be found in Table 6.

Table 6: Test Specification

Test subjects	The test was conducted on a total of 7 test subjects of ages between 21-29, and of different sex and occupation.
Participants	Each test was done with 1 test conductor and 1 test subject.
Materials	An iPad with the application running and a computer to write notes on the test subject's statements and propositions.
Description	The test subject was presented with the prototype layout seen in Figure 2c and 2d. The test was conducted as an informal qualitative talk with a basis in the test subject's interests in such a product. Transcripts from each test can be found at `http://lestrade.imm.dtu.dk/~s062596/data/test-transcripts.zip` and a summary of the results in section A.5 on page 103.

The main points from the user test was that the newspaper should provide an overview of its contents and that it should be easy to navigate relevant new and archived articles. The users also wanted to provide relevance feedback on articles and wanted an indication of the relevance of the article. Moreover, that the newspaper should provide a good balance between images and textual content and that images should be as large as possible. It was also noted that white space in between articles was not a problem and that an article should be read screen by screen, even if it means dividing text into a new set of columns. Some of the test subjects pointed out that problems may arise if there is not room enough in the top menu, e.g. when in portrait mode and, as the test subjects specifically requested, this could be solved by introducing a carousel-like arrow buttons to scroll menu items, or even just using touch interactions. Moreover, the user should be able to read the newspaper screen

by screen, meaning that trailing text should be put into a new set of columns whenever it exceeds the screen, see Figure 3.



(a) Reading pattern where the user has to scroll in order to see the full length of the column.

(b) Reading pattern where the user can finish reading a whole page before scrolling to read the next.

Figure 3: The figure shows reading patterns of full length columns and columns divided into screen sized chunks.

Many of the test subjects played around with the text and expressed that it was good that the iPad could read the text out loud for them. One user in particular expressed that the application with some polish would provide a readable layout, easy navigation and a good overview of its content. She thought that this was the problems with `http://nyhederne.tv2.dk/`[9] . No user expressed the need for any general news as presumed in the scenarios, only personalised news was of preference to the test subjects. They argued that if they wanted news of some kind, they would just create a section for it. Even so [Díaz and Gervs] argues that common sense dictates some "breaking" articles to be universally interesting and that the problem of some user may not receive them[10] could be solved by collaborative filtering. Collaborative filtering is a good way to apply wisdom of the crowd to the application, which might make it stronger as the number users grows. Finally according to the user tests, it should be chosen from which period the articles should come from, as opposed to what was extracted from the scenarios and

[9] *The website of a Danish news channel.*

[10] *This is also known as the black sheep problem.*

Figure 4: The figure shows mockups of the layout in landscape and portrait mode, respectively.

respondents from [Ihlström *et al.*] which suggests that the paper should be continuously updated.

Based on the user feedback a new design was developed. It is seen in Figure 4.

The top menu from the prototypes is kept, but arrows are added to solve the problem of overflow if the items gets too numerous. The menu bar is given a dark colour to provide some visual contrast from content to functionality. In the new layout articles are shown in full and images are maximised. Rules of readability determines, as opposed to that on print, that small point size text work better with a sans-serif font [Tidwell]. The neutral Helvetica has therefore been chosen as the body text, whereas the article headlines are the most important thing on the page, and therefore are supplied with the largest typeface. To provide them with further focus a serif font has been used. The section headers are assigned with a medium size, but wide, font because the user needs to navigate using this headline. However, the user already knows the name of the section (he has probably given it himself) and does therefore not necessarily need to read it – just recognise it. On this basis the section headline is placed in a bar and supplied with the same light colour as the background. This makes them more neutral, but still easy to navigate using the bar. To relate the paratext with

the section they are supplied with the same colour as the section bar. This should let the user associate the article with the topic of the section. Furthermore, the line that was on top of the articles (see Figure 2d) is moved to be beside it. This visually indicates when an article starts and ends. This will also aid to understand that the article continues if the article columns should be divided into screen sizes. Again the same colour as the section bar is used to set both the visual and contextual frame.

In the prototype the menu consisted of all the settings in a modal panel, but the test subjects wanted a division between visual tools, e.g. changing size of the font or colour scheme, and the content settings, i.e. the control of what each section should contain. The former is moved into a side menu with a button to access the latter, which is kept in a modal panel. This way the handy visual tools are only one interaction away, whilst the more complex settings are hidden away with two interactions. And the overview of article headlines in the section could be done as in Sublime Text 2[11] , see Figure 5

[11] *Sublime Text 2 is a text editor for coding.*

Figure 5: The figure shows the overview plus detail function in Sublime Text 2.

This overview could be placed in the side menu and show a larger (and readable) scale of headlines and the user should be able to see the images further down in the newspaper.

4.2 INTERACTIONS

In Figure 6 is the navigational structure of the application outlined.

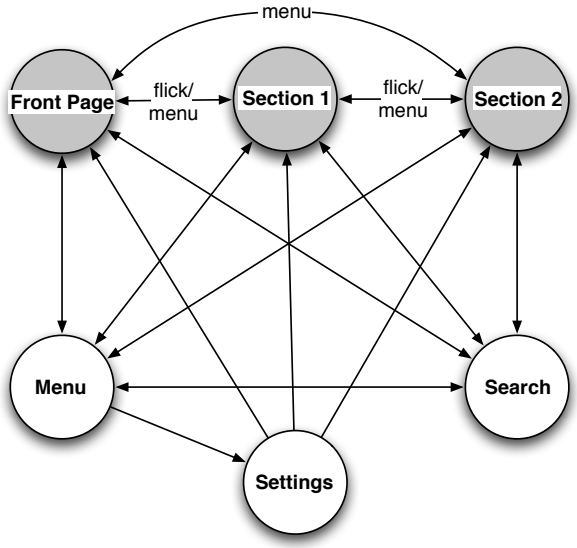


Figure 6: The figure shows the navigational structure of the application. Arcs are navigational interactions and nodes are pages in the application.

The three sections (grey nodes) holds the contents of the newspaper, and can be extended with additional sections through the settings menu. That the top menu presents the items along side each other supports the fact that sections lay along side each other in the navigational space, i.e. it is possible to navigate to sections beside the current through the flick gesture (see Figure 7).

Flick

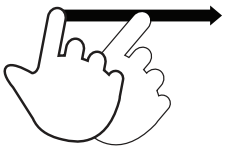


Figure 7: Touch gesture: Quickly brush surface with fingertip.

An animated transition could visually supports this structure by sliding the section out and the next in (same direction as the gesture). It is, however, still possible to directly reach any of the section through the top menu. Because the top menu is visible at all times except when in the settings menu, it is possible to go directly to any of the sections and do a search anywhere from the application, except of course from the settings menu. This means that the application is very interconnected and since the settings menu is meant to be used rarely the extra navigational step does not matter. The settings menu should only be used the first time the user opens the application to adjust the basic settings and then afterwards only to correct if the application does not comply with the user preferences, or when the super

user wants to adjust the settings. In a perfect world the user would never have to open the settings menu to adjust anything, only observe while the application learns the user interests and delivers what is expected.

When the user is at first presented with the application he should have as a direct path as possible leading to actually reading articles, which is of main user needs. He is presented with a form to make choices about the contents of the newspaper. The application provides the possibility for choosing whether the front page should be visible or not. This functionality is given to the user that would rather just have his sections and no front page. After this the user can choose the topic for the first section from a list of predefined topics. If the topic he is looking for is not in the list he can choose to fill out some keywords to cover his interests. After this he can provide it with a name for the section and choose to add another section or save his user profile. It is also possible for him to choose how many articles he would like in each section, including the front page. Figure 8 shows a mockup of the settings menu.

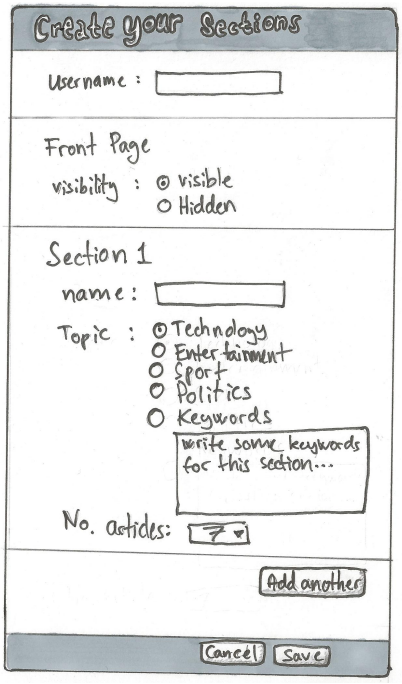


Figure 8: The figure shows mockup of the form that constitutes the settings menu.

Finally, it is important to state that users tend to play more with the screen when reading on tablet computers. Therefore the application should support selection of text and as from

the text it should be possible to get it read out loud. This way articles can also become audio books.

To be able to solve the editorial mix problem using CP it must be introduced and the presented constraints must be translated to logical constraints. This is done in the following chapter.

5 CONSTRAINT PROGRAMMING

The purpose of this chapter is to introduce the reader shortly to CP and afterwards define the problem as a COP. Different possibilities for implementation exists and this chapter will discuss these choices and conclude with a choice of algorithm.

5.1 CONSTRAINT PROGRAMMING AND PERSONALISATION

To be able to use CP to solve the editorial mix as a personalisation problem, it is necessary to define which problems CP works with. Constraint Satisfaction Problems (CSPs) and Constraint Optimisation Problems (COPs) are the two types of problems CP can be used to solve. The following descriptions of CSPs and COPs have been modified to fit personalisation problems from the original definitions provided by [Russell and Norvig, 2010] and [Apt].

A Constraint Satisfaction Problem is defined by the 4-tuple $(\mathcal{V}, \mathcal{X}, \mathcal{D}, \mathcal{C})$, where $\mathcal{V}$ is the set of values, $\mathcal{X}$ is the set of variables, $\mathcal{D}$ is the corresponding set of domains and $\mathcal{C}$ is the set of constraints on the variables.

Each variable has a corresponding domain and each domain has sub-domains corresponding to the attribute of each variable. A value also has a set of attributes, but they may extend the variables set of attributes. However, if a variable is assigned, the variable's attributes should reflect that of its assigned value[12]. Unlike [Russell and Norvig, 2010] and [Apt] a constraint is here defined as a function on specific variables returning a boolean value.

[12] *Others do not consider representation of values because they in their case only consist of simple integer, real or boolean values.*

Therefore the tuple of u values with w attributes, n variables with m attributes and p constraints, where the ith constraint is defined on s_i variables, can be expanded to:

$$\begin{pmatrix} \mathcal{V}: & \left\{ v_1 : \begin{pmatrix} v_1.a_1 \\ \vdots \\ v_1.a_w \end{pmatrix}, \ldots, v_u : \begin{pmatrix} v_u.a_1 \\ \vdots \\ v_u.a_w \end{pmatrix} \right\}, \\ \mathcal{X}: & \left\{ x_1 : \begin{pmatrix} x_1.a_1 \\ \vdots \\ x_1.a_m \end{pmatrix}, \ldots, x_n : \begin{pmatrix} x_n.a_1 \\ \vdots \\ x_n.a_m \end{pmatrix} \right\}, \\ \mathcal{D}: & \left\{ d_1 : \begin{pmatrix} d_1.a_1 \\ \vdots \\ d_1.a_m \end{pmatrix}, \ldots, d_n : \begin{pmatrix} d_n.a_1 \\ \vdots \\ d_n.a_m \end{pmatrix} \right\}, \\ \mathcal{C}: & \left\{ \begin{matrix} c_1 : func(x_{(1,1)}, \ldots, x_{(1,s_1)}) \to \mathbb{B}, \\ \vdots \\ c_p : func(x_{(p,1)}, \ldots, x_{(p,s_p)}) \to \mathbb{B} \end{matrix} \right\} \end{pmatrix} \tag{1}$$

Where $\mathbb{B}$ is either `true` or `false`.

A CSP is a special case of a Constraint Optimisation Problem (COP) and a COP is defined by the 5-tuple $(\mathcal{V}, \mathcal{X}, \mathcal{D}, \mathcal{C}, \mathcal{O})$, where the three first elements are defined as in a CSP and $\mathcal{O}$ is a set of objective (or cost) functions on variables, that determines the quality of a current state. The set of objective functions can be described with the same structure as constraints in CSPs and can be expanded as follows with q constraints, where the ith function is defined on t_i variables.

$$\mathcal{O}: \left\{ \begin{matrix} o_1 : func(x_{(1,1)}, \ldots, x_{(1,t_1)}) \to \mathbb{R}, \\ \vdots \\ o_q : func(x_{(q,1)}, \ldots, x_{(q,t_q)}) \to \mathbb{R} \end{matrix} \right\} \tag{2}$$

Where $\mathbb{R}$ is the set of real numbers.

Satisfaction (or regular) constraints are also called hard constraints and objective functions are called soft constraints because a solution can be found if all hard constraints are satisfied, whereas an optimal assignment is enough to satisfy objective functions.

Finally, a constraint on a single variable is called an unary constraint, a constraint on two variables is called a binary constraint and a constraint on three or more variables is called a global constraint.

5.2 PROBLEM REPRESENTATION

The division of rules between the front page and the section can be kept in the problem representation, which will, as we will see later, be the source for the possibility of incorporating lazy loading of each section. This section will therefore present a general problem specification for a section, which can be used in every section and on the front page, with varying constraints. Also, the front page will through this section and the following be referred to as a section, and specifically as section 0.

The set of values, $\mathcal{V}$, from equation 1 in the problem is represented by a library of currently available articles and the set of variables, $\mathcal{X}$, is represented by the available positions in the section. Each variable can then be assigned a value in the form of a specific article and when a complete assignment satisfies all constraints, a solution has been found. An article consists of a set of attributes, e.g. a date, the number of words in the article and a number indicating a relevance. Constraints are defined on variables and through them bound to their specific places in the section.

In the following the constraints presented in section 9 will be formulated as logical constraints divided into general constraints for all sections and specific constraints for the front page and other sections, respectively. Constraints defined on variables with letters, e.g. x_a and x_b, means that the constraint is defined for every combination of variables from the problem. Hard constraints (equation 1) returns whether it is satisfied and preference constraints (equation 2) returns a violation, where 0 means not violated.

GENERAL UNARY CONSTRAINTS

$$\left\{\begin{array}{lcl} \texttt{time-frame}(x_a) & \rightarrow & x_a.date >= today - 7, \\ \texttt{featured-space}(x_a) & \rightarrow & x_a.featured = \texttt{false} \vee \\ & & x_a.columns = 2 \vee \\ & & x_a.columns = 3, \\ \texttt{nonfeatured-space}(x_a) & \rightarrow & x_a.featured = \texttt{true} \vee \\ & & x_a.columns = 1 \vee \\ & & x_a.columns = 2, \\ \texttt{featured-image}(x_a) & \rightarrow & x_a.featured = \texttt{false} \vee \\ & & x_a.has\_image = \texttt{false} \end{array}\right\} \quad (3)$$

Where n is the number of positions and therefore variables, in the section and *today* is variable that holds the current date. The `time-frame` constraint is a part of the temporal aspect of the problem. It is set to include articles from a week ago, but can of course be adjusted. As the layout is defined in 2 and 3 columns the `featured-space` constraint is satisfied only when the article fills out 2 or three 3 columns. Likewise with the `nonfeatured-space` which is only satisfied with articles that fills 1 or 2 columns. These two constraints are apart of the spatial aspect of the problem. `featured-image` is a constraint, to control that a featured article should have an image. These final three constraints could be used as they are, but could also function as assignments along with a calculation of the relevance to base the rest of the constriants on, i.e. to determine that a featured article is an article with many words, has an image and has a lot of relevance; otherwise it is not.

GENERAL BINARY CONSTRAINTS

$$
\left\{
\begin{array}{rcl}
\texttt{featured-adj}(x_a, x_b) & \rightarrow & \texttt{not adjacent}(x_a, x_b) \vee \\
 & & (x_a.featured = \texttt{false} \wedge \\
 & & x_b.featured = \texttt{false}) \vee \\
 & & x_a.featured = \texttt{true} \wedge \\
 & & x_b.featured = \texttt{true}, \\
\texttt{featured-pos}(x_a, x_b) & \rightarrow & \texttt{not adjacent}(x_a, x_b) \vee \\
 & & (x_a.featured = \texttt{false} \wedge \\
 & & x_b.featured = \texttt{false}) \vee \\
 & & (x_a.featured = \texttt{true} \wedge \\
 & & x_a.position > x_b.position) \vee \\
 & & (x_b.featured = \texttt{true} \wedge \\
 & & x_b.position > x_a.position), \\
\texttt{adj-subj}(x_a, x_b) & \rightarrow & \texttt{if}(\texttt{not adjacent}(x_a, x_b)) \\
 & & \texttt{return max}(0, \\
 & & 64 \cdot \texttt{similarity}(x_a, x_b)^2 \\
 & & -64 \cdot \texttt{similarity}(x_a, x_b) \\
 & & +15.36) \\
 & & \texttt{else return } 0
\end{array}
\right\} \tag{4}
$$

Where `max(number, number)` is a function that returns the maximum of the given numbers and `similarity`(*variable, variable*) is a function that returns the mutual similarity between the values of two given variables. These general binary constraints control most of the editorial mix, because they express that two featured articles should not be placed adjacent to each other, that featured articles should be placed higher than its adjacent non-featured articles and that adjacent articles should have the same subject. The latter is a preference constraint because some uncertainty is introduced when controlling that two articles should have the same subject. That articles have the same subject can approximately be determined by how similar they are to each other, i.e. if the articles are much too similar they could be articles on the same story, and if they differ too much, they may be on a different subject. However, in between should roughly determine that the articles are on the same or a similar subject, which is what is needed. In the `adj-subj` constraint this is done by returning a violation based on the parabolic function seen in Figure 9.

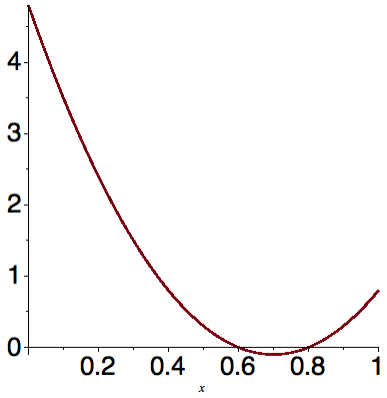


Figure 9: Plot of $10x^2 - 14x + 4.8$.

The function was chosen to express the needed violation if articles did not match the desired similarity. If the similarity of the two articles are below 0.6 or above 0.8, then its distance to this point will determine its violation by the parabolic function. These numbers are of course adjustable and will vary according to the selected similarity function, but are given to show a more expressive example. These constraints are part of the relational aspect of the problem.

GENERAL GLOBAL CONSTRAINTS

$$\left\{ \texttt{all-diff}(x_1, \ldots, x_n) \;\rightarrow \bigwedge_{i=1,\ldots,n} \bigwedge_{j=1,\ldots,n} \texttt{similarity}(x_i, x_j) < 0.9 \right\} \tag{5}$$

The final of the general functions determines that every article must be different. In the constraint this is expressed by checking if the similarity between the articles is too high, it could also be done by just checking the internal application ID of the article, but this might introduce more of the same story, but from different content providers. This is also a part of the relational aspect of the problem.

FRONT PAGE CONSTRAINTS

$$\left\{ \begin{array}{rcl} \texttt{main-stories}(x_a) & \rightarrow & \bigvee_{i=1,\ldots,g} \texttt{relevance}(x_a, i) >= 0.75, \\ \texttt{nonfeatured-image}(x_a) & \rightarrow & \texttt{if}(x_a.\mathit{has\_image})\ \texttt{return}\ 0 \\ & & \qquad \texttt{else return}\ 1 \end{array} \right\} \tag{6}$$

Where g is number of sections and `relevance`$(variable,\ section\ number)$ returns the relevance of the given variable in the given section number. The former constraint expresses the need for only articles of high relevance on the front page. This is therefore also a part of the relational aspect of the problem. The latter expresses that non-featured articles preferably also should have images on the front page.

SECTION CONSTRAINTS

$$\left\{\begin{array}{rcl} \texttt{topic}(x_a) & \rightarrow & \texttt{relevance}(x_a, k) >= 0.65, \\ \texttt{fp-article}(x_1, \ldots, x_n) & \rightarrow & \bigvee_{i=1,\ldots,n} \bigwedge_{j=1,\ldots,m} x_i.id = a_j.id, \\ \texttt{image}(x_t, x_{t+1}, x_{t+2}, x_{t+3}) & \rightarrow & \texttt{if}(\bigvee_{i=t,\ldots,t+3} x_i.has\_image) \\ & & \texttt{return } 0 \texttt{ else return } 1 \end{array}\right\} \tag{7}$$

Where k is the current section number, $a_1, \ldots, a_m$ is a list of articles from the front page that should be contained in this section and x_t are variables where t fulfils the equation $(t-1)\%3 = 0$. In other words, it is defined for every fourth variable. The first section constraints controls that articles should be relevant to the topic, which is therefore a relational part of the problem. The two latter controls that this section contains all necessary articles from the front page and that at least every fourth article should hold images.

It is worth noting that there is no constraint defining combinations of attributes to make up articles. In a classical sense the CP could solve this problem, but would probably end up with a combination of attributes for each article, that would solve the problem perfectly, but is impossible to find in the library. The classical sense of CP is therefore in some cases more hypothetical, meaning that it would solve the problem in terms of numbers and boolean values, but does not reflect the real world. This approach works well for school examples like a planning problem or the well-known n-Queens problem[13] and if one were to follow the classical sense of CP one would have to define constraints expressing legal combinations of attributes. This will, however, not be an optimised solution as the library of articles potentially could be very big in this problem and

[13] *A classic CP problem: Place n queens on a $n \times n$ chessboard, so no queen is attacked.*

would require a constraint for each, which would increase the problem size and therefore the time to solve it significantly. A small example of this is an automatic playlist generator written in Prolog, that defines a whole music library and generates a playlist based on some constraints[14] . Running this program one realises that this is not very efficient for a library of about 5000 songs. Therefore a novel approach is proposed. Instead of defining constraints based on the library of articles values are assigned based on look-ups in the library, which thereby introduces implicit constraints of attribute constraints, i.e. it is never possible for the system to find a combination of attributes, which is not found in the library, because they are taken from the library. This will also introduce some randomness to the compositions, which is often sought in real world problems.

[14] *The library can be downloaded here:* `http://lestrade.imm.dtu.dk/~s062596/data/PrologAPG.zip`.

Following the classical sense of CP the variables could be viewed as "mega-variables" and the combination of attributes could be defined as a sub-problem. The combination of attributes found in the library could be expressed as a big logical OR constraint. Whenever a value for a variable in the super-problem is needed the sub-problem could be solved. This structure of the sub-problem will incrementally look through the library returning the first value combination of attributes that satisfies the big logical OR constraint. This is done every time the super-problem needs a value, which is not an optimal solution.

The presented constraints are more or less simply translated from the presented textual constraints into logical functions and luckily only few contains multiple or every variable in the problem. It is in many cases profitable to choose a different representation of the problem if constraints builds on multiple variables. This can be done e.g by a tree decomposition or a reduction of the constraints to binary constraints. To compose the tree decomposition, first the constraint graph[15] needs to be considered. In this problem a complete graph of all variables will emerge, as it contains constraints that holds every variable. The tree decomposition is done by dividing the problem into sub-problems and again viewing them as "mega-variables" with their set of solutions as their respective domains, so the outermost problem becomes a tree. [Russell and Norvig, 2010] states that this works well if no sub-problem is too large, but for a complete graph of n variables the tree decomposition will be-

[15] *A constraint graph is a graph where vertices are variables of the problem and their links are the constraints that binds them.*

come a problem of $n-1$ sub-problems each with $n-1$ variables, which is not very efficient. As each sub-problem can be solved independently the decomposition could be done continuously until small enough problems emerge. However, the time is of course dependent on the branching factor of the tree, so this is not efficient either. Reduction of constraints to binary constraints can be done by introducing new variables, with constraints that defines the relation to this and old variables in the new problem. [Russell and Norvig, 2010] however argues that it is possible to design special-purpose inference algorithms that only handles global constraints and in practice this can be done by counting the number of variables the constraint is defined for and then deciding how to handle it.

5.3 CHOICE OF ALGORITHM

Because the problem is a fixed budget computation problem, in that the user should not have to wait too long for a solution, it is worthwhile exploring the algorithm choices of the solution.

Several algorithms exists to solve CSPs and many of them can be converted to work on COPs. A depth-first search using BACKTRACKING can e.g. be used. The search continues with a descendant spanning the whole tree, which is independently viewed as a new CSP. The search backtracks to the parent node, whenever an empty domain is encountered. The algorithm stops with the first assignment that satisfies the CSP if a single solution or inconsistency is sought. Because no information, other than that given in the problem formulation, is used to solve the problem it is a uniformed search and therefore not expected to perform very well [Russell and Norvig, 2010, p. 81].

With the use of heuristics the search can be improved. An example of this is the BRANCH AND BOUND search which work on COPs, but can handle CSPs as well. The branch and bound heuristic bounds the search by an objective function, that returns the current best assignment. States after a worse assignment is encountered are therefore not considered. Another example of a heuristic function is the *most-constrained-variable* (MCV) heuristic. This function always selects the variable that appears in the largest number of constraints, to be assigned

next. When used with backtracking, the performance can be greatly improved [Apt, pp. 337].

Constraint Propagation is another type of heuristic. Where the branch and bound heuristic is concerned with which variable to select, constraint propagation is concerned with the implications of the assignment of a variable. This means that if the assignment of a variable removes the possibility of some values to others in the solution, the values are removed from these domains. An example of this is *arc consistency*. If every constraint is represented by an arc, like in the constraint graph, but directed, arc consistency is when there for every value exists some value that it is consistent with. Arc consistency can be applied multiple times to obtain *path consistency* and until no inconsistency is left, which is the property of the MAC (Maintaining Arc Consistency) algorithm.

FORWARD CHECKING uses propagation whenever a variable is assigned to a value to detect inconsistency. It does, however, not detect for new inconsistencies after the removal of values.

Finally there is the MIN-CONFLICTS algorithm. It is the result of the application of *local search* to CSPs. It uses the min-conflicts heuristic, which chooses the value that results in a minimum number of conflicts. The algorithm is shown in Figure 10.

Local search are algorithms that do not care about which path they take to a solution, just that they find one. The MIN-CONFLICTS algorithm operates by moving to neighbours from a current state using the *min-conflicts* heuristic, i.e. selecting the value that results in a minimum number of conflicts with other variables. It can be applied to both CSPs and COPs. In the latter the techniques of *hill climbing* and *simulated annealing* can be used to improve the search [Vossen]. Local search has proved very effective in solving many CSPs and COPs. For the MIN-CONFLICTS algorithm to work an initial complete assignment is needed, so it can operate on a current state. The efficiency is depending on this initial state [Russell and Norvig, 2010].

In [Russell and Norvig, 2003, p. 143] a thorough survey on commonly used algorithms efficiency on commonly known CSPs is conducted and the results from the n-Queens problem is listed in table 7.

```
MIN-CONFLICTS(csp, max_steps) returns a solution or failure
    inputs:
        csp, a constraint satisfaction problem
        max_steps, the number of steps allowed before giving up
    current ← an initial complete assignment for csp
    for i = 1 to max_steps do
        if current is not a solution for csp
            return current
        var ← a randomly chosen,
              conflicted variable from VARIABLES[csp]
        value ← the value v,
                that minimises CONFLICTS(var, v, current, csp)
        set var = value in current
    return failure
```

Figure 10: The MIN-CONFLICTS algorithm for solving CSPs by local search. The initial state may be chosen randomly or by a greedy assignment process that choses a minimal conflict value for each variable in turn. The CONFLICTS function counts the number of constraints violated by a particular value, given the rest of the current assignment [Russell and Norvig, 2010, p. 221].

Table 7: Comparison of various CSP algorithms on the n-Queens problem. The algorithms from left to right, are simple backtracking, backtracking with most constrained variable (MCV) heuristic, forward checking, forward checking with MCV, and MIN-CONFLICTS local search. Listed in each cell is the median number of consistency checks (over five runs), required to solve all n-Queens problems for n from 2 to 50; note that all entries are in thousands (K). Numbers in parentheses mean that no answer was found in allotted number of checks [Russell and Norvig, 2003, p. 143].

Problem	Backtracking	BT+MCV	Forward Checking
n-Queens	(> 40,000K)	13,500K	(>40,000K)

Problem	FC+MCV	Min-Conflicts
n-Queens	817K	4K

Table 7 shows that the MIN-CONFLICTS local search is by far the most efficient algorithm for solving the n-Queens problem. The initial assignment of the editorial mix problem can be randomly chosen or by a greedy algorithm and the neighbour states can be generated by selecting a new value for a variable or swapping the values of two. The CONFLICTS function could return the number of hard constraints violated plus the sum of violation of the preference constraints ([Apt, pp. 372] and [Russell and Norvig, 2010, p. 150]). It is afterwards up to the solution check, to check if all hard constraints are satisfied and if an optimal solution has been found for the preference constraints. In choosing a variable, MCV can be used to guide the search more than the proposed random conflicted choice.

Because the local search techniques can be applied to the MIN-CONFLICTS algorithm and because of its iterative nature this algorithm is chosen to solve the problem. The property that the algorithm starts by an initial complete assignment and refines it, makes it attractive for the purpose of composing the newspaper and solving personalisation problems in general. Either it returns an optimal solution or it returns the current best assignment if there is no iterations left.

5.4 APPLICATION STRUCTURE

After the description of the interface, the problem and choice in algorithm it is possible to discuss how the client side should handle user requests and the web server should handle the requests from the client side. In Figure 11 is shown a use case diagram of how the client side and web server handles use cases.

In the diagram an extra use case has been added to show how the system should handle the initial topic selection by the user. The user is able to select relevant topic categories to get an easy start with the application and his choices are thereafter saved to the user profile for later use. The user can thereafter get an overview of the content from the front page or read articles from a selected topic category, both in a nice readable layout and in a composition based on the editorial mix. Moreover, it is

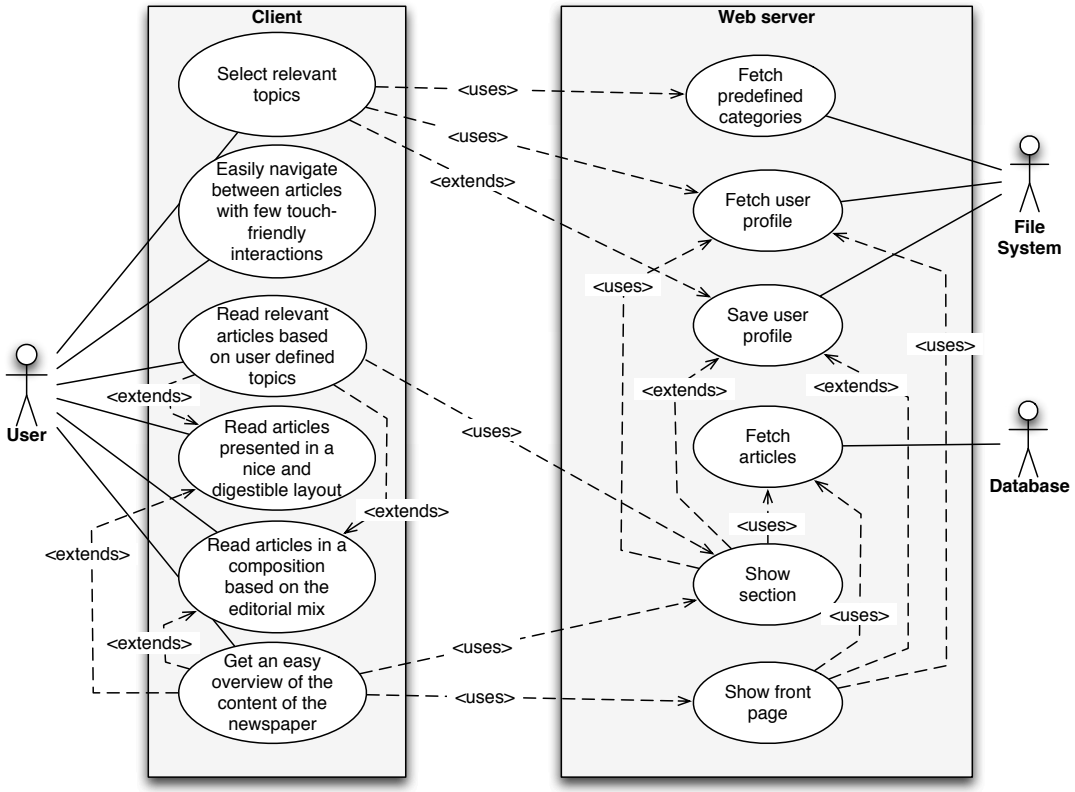


Figure 11: The figure shows the use cases of the system and how the client side handles them and how the web server acts to accomplish requests from the client side.

possible to get an overview of the articles within a section from a list of headlines from articles contained within it.

From the constraints presented in this section it is possible to derive server tasks. As noted earlier the latter three constraints in equation 3 could be used as functions to determine if the article is featured or not. It makes sense to do this on the server as a pre-computation along with the similarity and relevance of articles. For this to be possible some metadata on the articles is needed. In order to determine how many columns an article should fill a word count or character count is needed for each article. In addition, it could also be interesting to look at the image sizes, which potentially could hold much information, and in some cases even constitute the body of the article.

The next chapter will present the implementation of the client and server, which constitutes the product of this project.

6 IMPLEMENTATION

This section describes the implementation of the design choices made in the two previous chapters in terms of client side and server side tasks.

The proposed design presented in the previous chapter has accounted for the presented requirements and user needs, but there are some requirements that will not be implemented due to prioritisation.

Chapter 4 and 5 presented an application that accounted for a user model, this chapter will only present how to collect the necessary user data and use it in the application – and not implement it. It will, however, describe which metadata is needed and how to acquire it.

The social aspects is an important part of the system are also useful channels for awareness. [Tidwell] even states her editorial mix pattern as a social media pattern. Nonetheless, these will not be implemented in the presented application as it does not contribute with new knowledge to the field. The gathered news articles to be used in this project contains both images and videos, but only images will be considered here. It is however trivial to implement support for videos as the same space allocation principles applies, but it was not prioritised.

Also, the personalised summaries have already been very well explored in [Díaz and Gervs] and this project will not try to compete with this solution, so only the first few sentences will constitute the excerpts from articles to be used on the front page. Because the full articles are shown in the sections no excerpts will be used in these. This should, however, be supported in the further development of the application.

The sections are based categories. These are by no means complete and they have not been verified. However, they are some of the most recurring in popular news sites and are used in or-

der to proof the concept is possible. Thus, their definitions are not comprehensive either.

Finally, only a subset of the editorial mix constraints, presented in the Constraint Programming chapter (chapter 5), will be implemented. Furthermore, before the user test was done the implementation had already started. This led to the implementation of a fairly complex layout constraint to minimise white space. It turned out that some white space was actually a user preference, but the implementation of the constraint will be presented nonetheless, as it shows a good example of what is possible with the system.

6.1 SIMILARITY AND RELEVANCE COMPUTATION

The [Dublin Core Metadata Element Set, Version 1.1] proposes 15 metadata elements for documents:

- Title
- Creator
- Subject
- Description
- Publisher
- Contributor
- Date
- Type
- Format
- Identifier
- Source
- Language
- Relation
- Coverage
- Rights

The articles in this project have been acquired using the [Readability API] to scrape articles given by links from RSS-feeds. This way it is possible to get the full article. Under normal circumstances, these would probably be supplied by a content provider, say through agreements with newspaper companies or social networks. The Readability API provides several metadata elements for the parsed articles; a domain, a title, an article URL, a lead image, the author, an article excerpt, a word count and a date of publication. This satisfies many of our needs, but there are still some very crucial calculations to be done, i.e. the article relevance according to user topics and similarity between articles.

The analysis of article relevance and similarity to other articles will use the same analysis, namely a keyword analysis using the WordNet and entity comparison using [Open Calais API]. These will be presented in the following.

WordNet is a large lexical database of English words and their relationships in the form of different graphs. WordNet is based on *synsets*, which is a set of synonyms that describes different meanings of the same word. WordNet has a hyperonymy and hyponymy graph for noun synsets, which is based on the ISA relation between words. A *hypernym* relation is a generalisation, e.g. a hypernym for a bed is a piece of furniture; and a *hyponym* relation is a specification, e.g. a hyponym for a bed is a bunkbed. For nouns there also exists the meronymy graph, which is a graph describing the part-whole relation; a chair e.g. has a back, a seat and legs. Also, parts are inherited by superordinates, e.g. if a chair has legs, then an armchair has legs as well. Furthermore, WordNet has a graph describing elaboration, i.e. *troponyms*, of verbs synsets, adjectives organised in terms of antonymy and adverbs which can be be described in terms of adjectives. The synsets, the hypernym and hyponym graph and the troponym graph of WordNet are the most interesting, because they describe different meaning of a given word, whereas the others describes relation to other words.

The initial approach involved computing the TF-IDF similarity using the Python libraries for this [Bird *et al.*]. This approach works on a bow (bag-of-words) with keywords and weights representing a single item. The weight is computed by the number of occurrences in the provided text and a cosine of

the angle between them determines the similarity. However, Python also provides an interface for working with WordNet. This allows for a more in-depth analysis of the obtained news articles. [Bouras and Tsogkas] presents an algorithm for enriching articles using WordNet's hypernym graph, which proves to yield precise results in the context of enhancing labelling using K-means clustering. However, only the WordNet enriching of articles will be used to aid the keyword based approach in this implementation. In the presented approach a subgraph of WordNets hypernym graph is generated by the top 20% frequent keywords of an article and weighted by equation 8

$$W(d, f) = 2 \cdot \frac{1}{1 + e^{-0.125(d^3 \frac{f}{TW})}} - 0.5 \tag{8}$$

Where d stands for the node's depth in the graph (starting from root and moving downwards), f is the frequency of appearance of the node to the multiple graph paths and TW is the total number of words used to generate the hypernym graph. An example of such a hypernym graph is seen in Figure 12.

To be able to work with hypernyms, words from articles must be converted to synsets. For each word there exists a synset for each use of the word, with the most frequently used first. Every synset is included in the analysis of this implementation, but in a later stage this could be further focused by only using the top n uses of the word. In the algorithm given by [Bouras and Tsogkas] only the most general use is included (see Figure 12), which is a rough assumption. By this it is assumed that the meaning of every word extracted from an article is of the most general use. A better solution would be to use the Wu-Palmer similarity to find the best match of words ([WordNet Interface Documents]). Wu-Palmer calculates a similarity based on the lowest common ancestor in the hypernym graph, which is available through the Python implementation. This way a set of keywords is gathered based on their most frequently common uses, as opposed to their independently common uses.

WordNet distinguishes among Types (common nouns) and Instances (proper nouns)[16] , so it is possible to extract these from the articles to base the analysis on. Both nouns and adjectives are extracted from articles, but adjectives could also be derived from adverbs to add meaning. A hypernym graph is produced

[16] *Common nouns are general words for any people, places and things while proper nouns are specific names of individual people, places, things, or a title.*

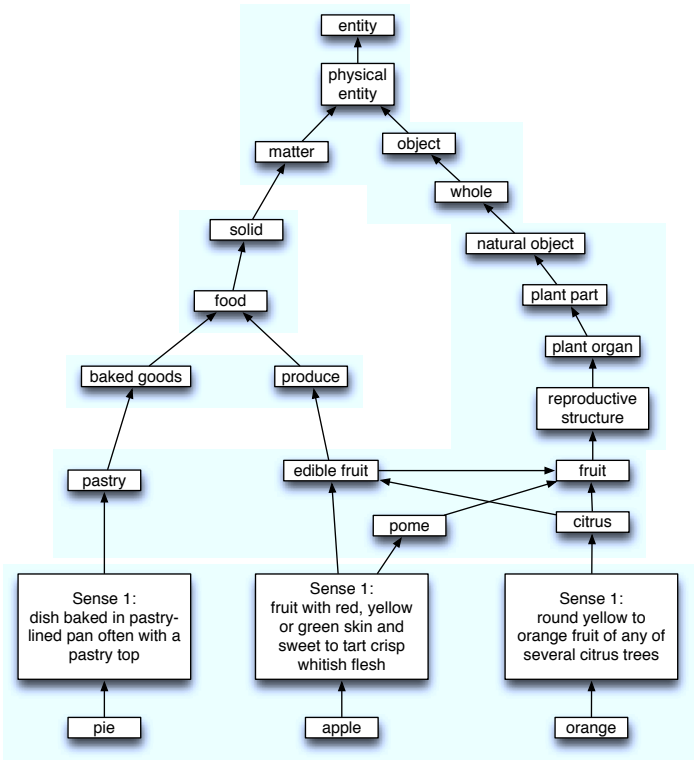


Figure 12: The figure shows an example of a hypernym graph that could be generated by the WordNet-Enrich algorithm.

of a maximum of 9 levels up in the graph and each of these synsets are weighted and the top 20% extracted hypernymns along with the top 20% of the original given keywords constitutes the final set of keywords, see Figure 13.

WordNet-Enrich(a) **returns** enriched list of keywords
inputs:
a, an article
$total\_hypernym\_graph \leftarrow$ a new graph
$kws \leftarrow$ fetch 20% most frequent kws for a
for each keyword kw in kws **do**
$hgraph \leftarrow$ WordNet-HypernymGraph(kw)
for each hypernym h in $hgraph$ **do**
add 1 to frequency of h in $total\_hypernym\_graph$
for each hypernym h in $total\_hypernym\_graph$ **do**
$d \leftarrow$ calculate depth of h in WordNet hypernym graph
$f \leftarrow$ frequency of h
$weight \leftarrow 2 \cdot \frac{1}{1+e^{-0.125(d^3 \frac{f}{\text{SIZE}(kws)})}} - 0.5$
SORT-WEIGHTS($total\_hypernym\_graph$)
$important\_hypernyms \leftarrow$ top $\frac{\text{SIZE}(kws)}{5}$ of $total\_hypen\_graph$
return $kws \leftarrow kws + important\_hypernyms$

Figure 13: The WordNet-Enrich algorithm for enriching articles using hypernym graphs from WordNet, inspired by [Bouras and Tsogkas].

After this extraction the similarity is computed by a path similarity, which is based on the shortest path between the words, on the best matches between words. The Wu-Palmer similarity, could again, be used instead, but to avoid more computation time the naïve solution was chosen.

Because it is possible to distinguish between proper nouns and common nouns in WordNet as well, this was used to extract instances from articles. However, WordNet often seemed to have problems with this particular issue and would, e.g. match an article about an exhibition of Claude Monet's flower garden to a topic containing the company Apple inc.

An example of where Wordnet does not perform well is New York. When the words are removed from each other, they provide another meaning, which is what the implementation supply WordNet with. But Open Calais can handle the full doc-

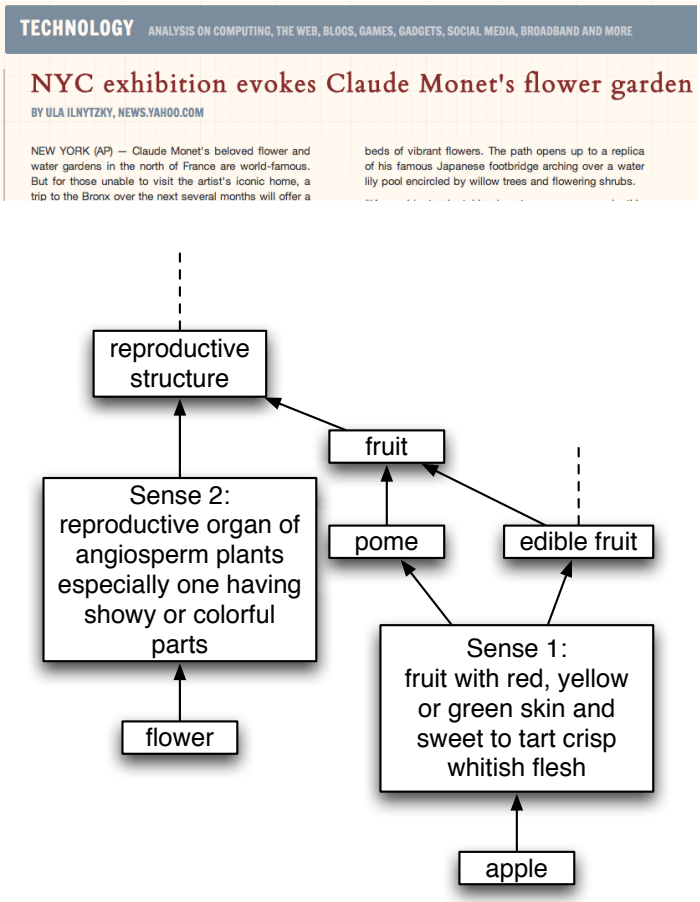


Figure 14: The figure shows an example a close realtion in the hypernym graph of the words "apple" and "flower".

ument and can detect where to keep the words together in order to find meaning.

To account for this a similarity using the Open Calais API was implemented.

The [Open Calais API] provides analysis of documents and extraction of several kinds of metadata. This implementation will only use the extraction of entities, but it could however, be interesting to see how well the document categorisation performs, which includes many general news topics. After the extraction of entities the similarity is calculated by the sum of entity matches divided by the average length of the two given entity sets.

The final similarity is found by the average of the WordNet similarity and the entity similarity. This seemed to form a solid structure for computing the similarities between articles, but the relevance according to the user defined topics needs to be calculated as well. Fortunately, the decision of using keyword based user models lets us use the same functions to compute similarity between articles and topics. In Figure 15 is a plot of similarities between articles and 9 different topics. The similarities are ordered by an average between them.

These similarities between articles, worked well but was somewhat strict. The highest similarity given was 0.396 in a range of $[0;1]$. 86% of them was below 0.1. Because of this and that even fairly low similarities yielded promising results it seemed appropriate to standardise it. This is also plotted in Figure 15. The standardising of the data is done by taking the logarithmic function of their percentages to flatten it more and dividing it by the largest similarity to spread it evenly over the $[0;1]$ scale.

The articles are stored in a file on the server for easy retrieval by the client side and the metadata is stored in a database. The articles could have been stored in a database along with the metadata for a more sustainable solution, but this was just an easy solution.

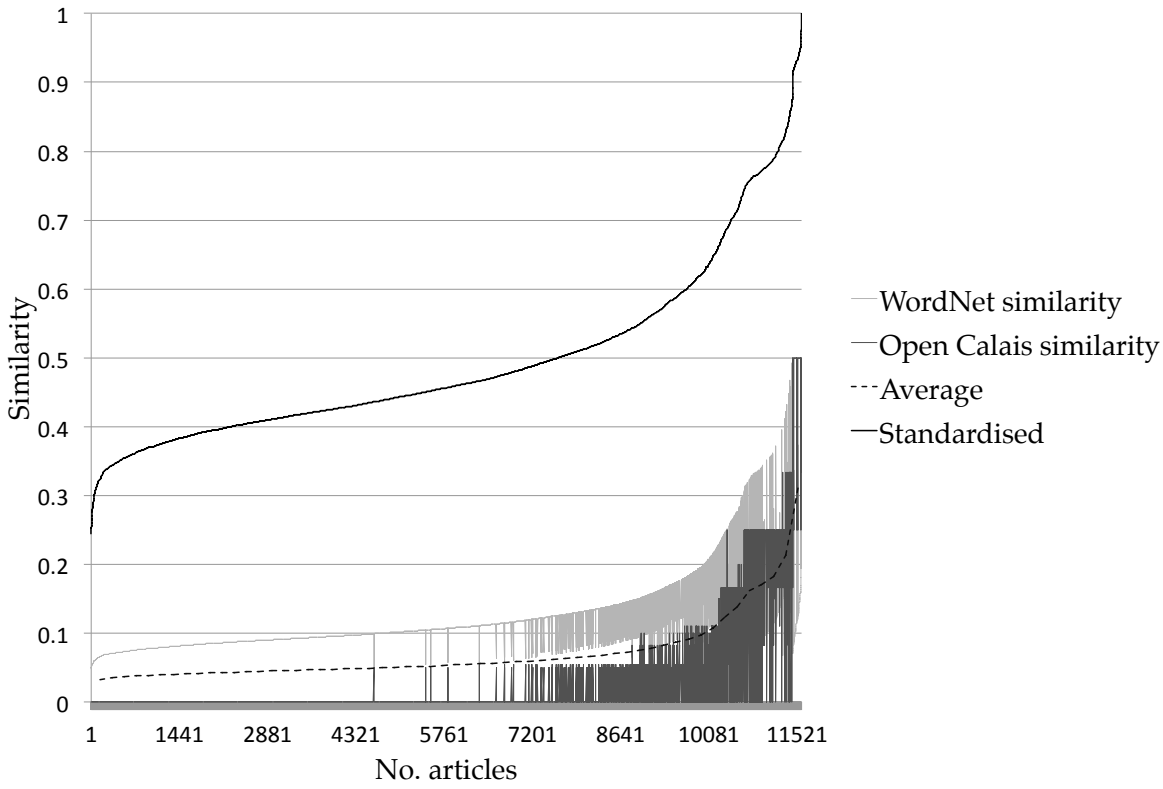


Figure 15: The figure shows a plot of articles with WordNet and Open Calais similarities, their average and a plot of the standardised average.

6.2 INTERFACE

In order to fulfil the requirements of a column based interface it seemed appropriate to use a grid based layout. [Twitter Bootstrap] provides an excellent framework for this which is both touch friendly and responsive[17] . A layout was developed using conditional styling so an article can be assigned a class according to its size in columns so the layout can adapt to display the article differently according to size. An article that uses 3 columns in landscape mode should necessarily only uses the available 2 columns in portrait mode. This also promotes the possibility of having articles that displays in 1 column in portrait and 2 columns in landscape and finally articles that display 1 column in both.

[17] *Responsive Web Design means that the layout is adaptable to the viewing devices screen size.*

The calculation of how many columns an article should use was done as a preprocessing, first by the largest image found along with the article and next the number of characters in the article. Users from the test pointed out that images should be as large as possible, and this should therefore dictate the space allocated for the article.

There are different ways of displaying text in columns on the web, but one of the newcomers is [CSS3 Multi Columns], which introduces easy styling of text in balanced columns. It was chosen to use this technology to see what was possible with it and because it is a technology that is under development. This means that it is not possible at the moment to divide the columns or let, e.g. an image span a number of columns using styling. Either an object has to span all columns or be wrapped into one, but their respective functionality is however soon to be supported. One could manually implement the division of columns, but it was deemed unimportant for the project to investigate this further. This means that some white space beside images might occur and that columns continues with no division.

The main page of the application is created dynamically. User preferences are registered from the from submission and saved locally. If the view changes, the articles from the former section is removed and the new ones are inserted in their place. This

provides the possibility of introducing animations between section changes and lazy load of the pages.

In Figure 16 is shown a sequence diagram of what the system does in order to display the front page (or a section), when the user opens the application.

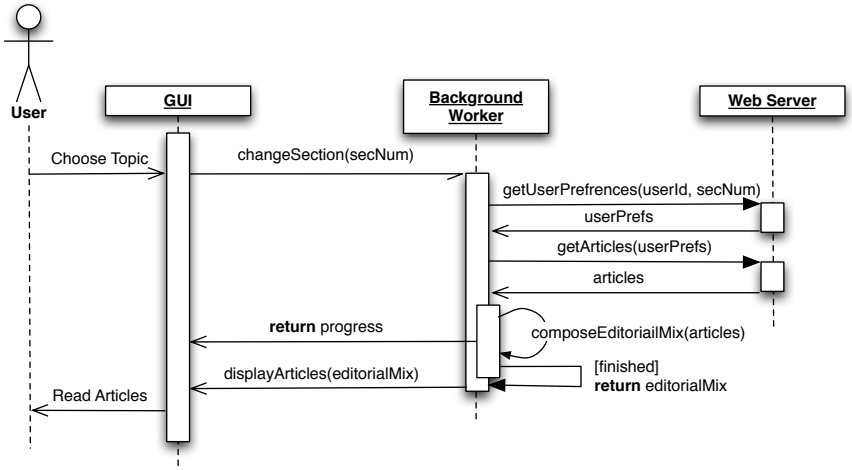


Figure 16: A sequence diagram from when the user chooses a topic until reading articles. *secNum* is a section number, (front page is section 0), *userId* the user id, *userPrefs* the user preferences on a given section and *articles* is a library of articles used to compose the editorial mix.

When the application is opened, or the application is changed to display a section, a background worker is initialised to compose a mix of articles. If the mix of articles in a section, or the front page, have already been computed, it should not have to recompute it. The background worker needs to get both the user preferences of the chosen topic and articles that potentially fit the user preferences. While the worker computes the editorial mix it sends messages to the user interface about the progress. This is used to provide feedback to the user. When it finishes the user interface is asked to display the articles.

The implementation does not log user behaviour and does not support relevance feedback to be stored in a user model. The user model have been manually written with manual definitions of keywords and their respective weights for sections. It was chosen not to focus on the logging of data to represent the user model, because it does not contribute with new knowledge to the field. The logging of data could, however, be done by detecting when an article enters the screen, e.g. using [jQuery Waypoints]. [jQuery Waypoints] provides events whenever an item enters the screen, then a time stamp could

be registered and stored with a duration, when a new article enters the screen. Problems might, though, emerge when more articles are shown on the screen at the same time. Which article does the user read? A solution to this could be to register the touch, as the user might interact more with the screen in some places based on the position of the article he reads. This project will leave the problem to its field of study.

The next section will describe how the background worker handles its assigned tasks.

6.3 BACKGROUND WORKER

The choice of using a background worker to do the computation is essentially based on minimising the work for the UI thread. [JavaScript Web Worker] introduces long-running JavaScript (js) scripts which are not interrupted by scripts that respond user interactions, and allows long tasks to be executed without yielding to keep the page responsive. With this supporting browsers assigns a thread to handle the background worker tasks, but it remains in the same process.

Background workers are not meant to be numerous, because they require a high start-up performance cost and per-instance memory cost. On this basis a background worker was build to handle the section constraints and the front page constraints, respectively. In the further development it would make more sense to merge the two files, as they share a lot of code. It did however give a nice division between their respective assignments. When the user has submitted the form, as described in section 4.1 on page 31, a background worker is asked to compute the editorial mix. It first creates the COP, with variables based on the user input, their respective domains and constraints to fit the section. Information is sent from the main script to the worker, so it can base the constraints on this, e.g. a list of articles from the front page is sent when a section should be composed. This way the knowledge is kept in the main script and shared amongst workers to be applied in specific problems. After the COP has been created it fetches potential articles and calls a library to do the constraint computation.

6.4 CONSTRAINT PERSONALISATION LIBRARY

A library was developed for solving personalisation problems using CP. It was implemented using the MIN-CONFLICTS algorithm along with the MCV heuristic to guide the selection of variables. The motivation for building the library comes from a need of a CP library that supports the use of a list of values consisting of real world items instead of having to define the items as a sub-problem. This section will describe how to use it and discusses the choices of implementation. The library is from here on referred to as *Constraint Personalisation Library* or CPL[18] .

[18] *The library can be downloaded here: http://lestrade.imm.dtu.dk/~s062596/js/cp.js.*

The library provides the possibility for creating a list of variables, with their respective domains and subdomains, see Table 8.

Table 8: Declaration of a domain, subdomain and variable

```
var d = new Domain();
d.addSubdomain(
  // descrete domain (min, max, gap)
  new Subdomain(0, 1, 1),
  // subdomain name
  'images',
  // function for retrieving value attr
  function(x) {
    return x.images.length >= 1? 1: 0;
  }
);
// variable at position 0
var variable = new Variable(0,d);
cop.variables.push(variable);
```

After variables and domains have been declared it is possible to declare constraints on them, see Table 9.

The constraints of the COP is stored in a list of lists of lists for two reasons. Firstly the nested list of nested lists is interpreted as a conjunction of disjunctions of conjunctions. The outermost conjunction is interpreted as the set of constraints,

Table 9: Declaration of a constraint

```
var c = new Constraint(
  // constraint name
  'image0',
  // violation function
  function(vars) {
    if (vars[0] > 0) { return 0; }
    return 1;
  },
  // name of subdomain to work on
  'images',
  // variable positions to get
  // in the violation function
  [0]
);
cop.constraints.push([[c]]);
```

where the inner disjunction of conjunctions is interpreted as a decomposition of the constraint. This is to make it possible for the developer to declare a list of alternate sub-constraints as a decomposition of a constraint, i.e. it is possible for the user to declare a set of sub-constraints where either one of them should be fulfilled in order to satisfy the rule. This means that $[[[c_1, c_2], [c_3]], [[c_4]]]$ would be interpreted as $((c_1 \wedge c_2) \vee c_3) \wedge c_4$, where c_1 to c_4 are constraints.

An example of where this can be used is in the earlier described `image` constraint (see equation 7 on page 47). This constraint is defined on 4 variables, where either one of them should have an image for the constraint to be fulfilled. Instead this could be described as a list of four lists with one constraint in each. Constraints that each looks like the one presented in the Table 9, on variables in position 1 to 4, 4 to 7, 7 to 10, etc. There might also emerge situations where it could be useful to have alternating conjunctions of constraints. An example of this is the earlier mentioned white space minimisation, which is expressed as a more complex constraint.

The minimisation of white space can be expressed in terms of columns as; the accumulated sum of the columns assigned to articles in the problem in the next step minus the accumulated

sum in the former step always be dividable with both 2 and 3. The following boolean expression expresses the same:

$$(sum_i - sum_{i-1} < 2 \vee sum_i - sum_{i-1} \texttt{ mod } 2 \neq 0) \wedge$$
$$(sum_i - sum_{i-1} < 3 \vee sum_i - sum_{i-1} \texttt{ mod } 3 \neq 0)$$

Where `mod` is the modulus fiunction.

In other words; any composition of articles should always fill out the screen in both portrait and landscape. Many possibilities for achieving this is available. For a list of three articles, this could be done by having three articles where each would have 2.5 columns assigned. An article that is assigned 2.5 columns means that is fills two columns in portrait and three in landscape, whereas an article assigned 1.5 would display in one column in portrait and two in landscape. A more complex function could also describe the possibilities for two articles in a row, or three, or even more. The constraints could therefore be described as a disjunction of conjunctions as the following:

$$\begin{aligned}&[[columns_0, columns_1, columns_2],\\ &\ [columns_{0-1}, columns_2],\\ &\ [columns_{0-2}]]\end{aligned} \tag{9}$$

Where the number denotes which variable positions in the problem the constraint works on. A declaration of a white space minimising constraint over two variables is seen in Table 10.

The second reason for this representation of constraints is that it is possible to declare constraints on fewer variable and that it is possible for the system to prune some calculations. The first constraint in a disjunction that evaluates to true, or in this case to violation 0, makes it possible to discard the rest of the constraints in that disjunction. If the constraints are then ordered by how many variables they work on, so the system tries to evaluate the constraints in a conjunction that is defined on fewest values first, then it may never have to evaluate constraints that are global. In worst case it would have to evaluate all of them. An example of this ordering is seen in equation 9, where the first nested list is only of unary constraints, the second a list of a binary and an unary and finally a single global constraint. However, it is also worth deciding how important it is that the application explores more complex solutions if simpler solutions works just as well. In any case it leaves the possibilities for the developer to optimise his definition of the problem.

Table 10: Declaration of a constraint for minimising white space

```
var c = new Constraint('columns0-1',
  function(vars) {
    var violation = 0;
    if (vars[0] == 1) {
      if (vars[1] != 1.5) {
        violation++;
      }
    } else if (vars[0] == 1.5) {
      if (vars[1] != 1) {
        violation++;
      }
    } else {
      violation++;
      if (vars[1] != 1 && vars[1] != 1.5) {
        violation++;
      }
    }
    return violation;
  }, 'columns', [0,1]);
cop.constraints.push(c);
```

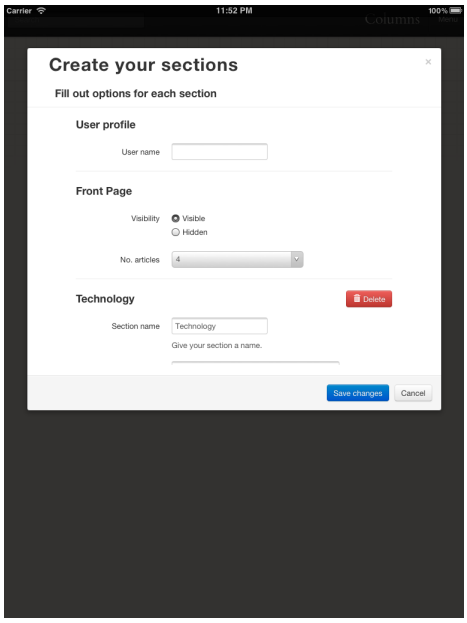
In further development it could be interesting to explore automatic reduction and organisation of constraints.

[Russell and Norvig] proposes constraint weighing to aid the organisation and furthermore maybe even lead the search to concentrate on variables that is bound by these constraints to solve them first. Constraints in this implementation are only represented as preference constraints, so every constraint returns a violation. This aids the search towards concentrating on changing variables that yields a lesser violation, but it also means that the returned assignment could violate constraints that are not supposed to be violated if the algorithm had trouble with finding a solution for the given problem. Using solely preference constraints eliminates the possibility of letting the search go on until a solution has been found. On the other hand one could argue, with a fixed budget computation problem, it is better to deliver the best assignment so far, than nothing at all.

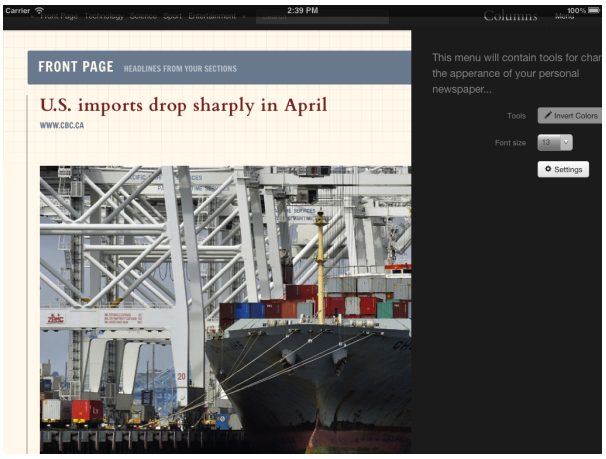
Constraints that have been implemented to compose the sections are `all-diff`, `fp-articles`, `topic`, `main-stories`, `adj-subj`, `images` and constraints to minimise white space. Also, the `featured-image` and `nonfeatured-image` have been implemented, but only as the same constraint that prefers images on every article, meaning that there is not a distinction between featured and non-featured articles.

Screenshots of the final implementation is seen in Figure 17 and can be accessed on `http://lestrade.imm.dtu.dk/~s062596`.

The next chapter will evaluate the solution in terms of quality and performance.



(a) Screenshot of the form.



(b) Screenshot from the sports section.

(c) Screenshot from the technology section (portrait).

(d) Screenshot from the technology section (landscape).

Figure 17: The figure shows four screenshots of the final implementation.

Part III

EVALUATION AND PERSPECTIVES

Evaluation of the Solution and Discussion of uses in New Contexts

7 EVALUATION

This section evaluates the implementation as a solution to the problem of personalising the editorial mix and Constraint Programming as a technology for creating personalised web applications.

7.1 TEST

The test described in Table 6 on page 33 was conducted on 7 test persons. This is not a solid ground to base decisions upon, but it has been helpful in finding some tentative problems. Main design choices should, in the further development of this application, be verified on a larger empirical basis.

The editorial mix has been conducted on rules inferred from analysis of conventional newspapers. It has not been verified that the composition of articles using these rules actually satisfies user needs better in a digital environment – even if users are unaware of them. This could be done by turning constraints on and off in the system and presenting them to users, so they can give their feedback on the best composition. In addition, new rules could be added to test new hypotheses. It could for example be interesting to explore users' preferences on the two rejected reading behaviours presented in [Holsanova *et al.*]:

- Readers Prefer new information and expect this to be on the right in the semiotic space
- Readers scan the semiotic space before taking a closer look at certain units

The next section will evaluate the interface of the application.

7.2 INTERFACE

In the implemented layout no visual difference between featured articles was supplied. Articles was just supplied with more space if the contents could fill it out. In the further development of the application a distinction between featured and non-featured content should be applied as described in the presented constraints in section 5.2 on page 43. This way content distinction can be applied visually.

The implemented columns constraints does not always provide good solutions, because they account only for horizontal features of the articles and not vertical. Therefore problems emerge when long-length articles are placed beside short-length articles leaving a lot of white space under the short. The implementation of this fairly complex constraint show that it is possible to implement constraints layout, but that it is hard to formalise and account for every detail. Therefore it worthwhile extracting key features about the layout that is needed to be handled and spend time on reducing constraints to a minimum, let other technologies handle these problems or explore a combination. A constraint that would co-operate with the positioning of articles using [jQuery Masonry] could be a solution for this. Here it is worth noting that the automatic placement could destroy the combinatorial work of the constraints. However, it seems that [jQuery Masonry] works to keep the chronological structure of elements, which minimises the harm it can do.

At an early stage, the pagination on a section was discarded, because a preliminary test (ask around) showed that users wanted to scroll down to see the full section. This was also necessary if full-length articles were to be shown, but the reading behaviour analysis suggests that articles should be divided into chunks of subjects, which may be better to visualise using pages. Following the reading behaviours more strictly, the featured article could be shown with large headlines, large images and a longer length excerpt, and stories on the same subject could surround it with only headlines, smaller images and short excerpts. If the user then wants to read one of the articles in full length, he can select it, and the article could be displayed in full, using the whole screen.

Furthermore, [Ovesson and Wikström] suggested ads in the application, but it was implemented. It is, however, possible specific to extract keywords and weights for a user on a topic basis. This could be done by direct relevance feedback from the user and by implicit relevance feedback by how long time a user spends on an article, as described in the former chapter. The new weights on keywords can be assigned based on the user behaviour in combination with the calculated weight from the article. This is actually in line with what [Google AdSense] does. AdSense is build on analysing the semantic structure of documents using WordNet, but might include other techniques ([Hane] and [Oin]).

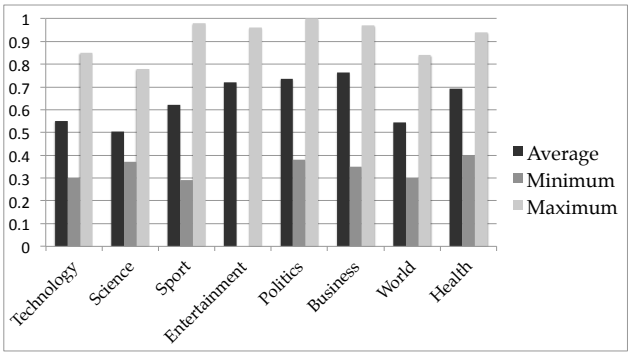
The application is implemented with a responsive design, with a layout that is familiar to that of a conventional newspaper. It has explored some possibilities for having full articles in balanced columns with a flat navigational structure of user defined topics.

7.3 CONTENT

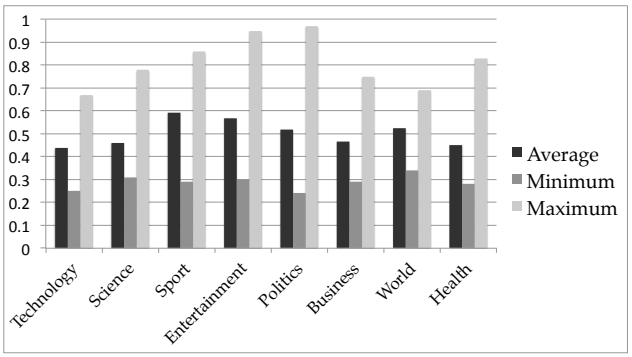
This section will analyse and discuss the solution with regards to content.

Similarities were computed with respect to manually defined topics[19] and with respect to articles in between. To test whether similarities actually make sense it seems logical to look at similarities of articles that match a topic and articles that do not match. The feeds of the gathered articles are often categorised under a certain topic. These topics were saved as tags on each of the articles, and it was then possible to analyse the similarities of articles that hold tags that match the defined topic and which do not. Figure 18 shows averages and minimum and maximum of similarities distributed on the defined topics on articles that hold a tag matching a topic and articles that do not hold a tag matching the topic, respectively. The overall average similarities for matching articles is 0.57 and for mismatching articles it is 0.5 in a range of $[0;1]$. This seems to be a low match similarity and very high mismatch similarity. The low match similarity could be explained by the definitions of the topics, which might not be the same as what the content providers deem as

[19] *The definitions can be seen in Table 12 and 13 on page 106.*



(a) Similarities of articles that matches the section.



(b) Similarities of articles that do not matches the section.

Figure 18: The figure shows respectively matches and mismatches of similarities distributed over sections.

relevant for these topics. Moreover, the definitions are by no means comprehensive and should more be seen as a subset of their general definitions. The high mismatch similarity average could be explained by the fact that some of the article are actually matches, but are not listed under these topics. An example of this is shown in Figure 19.

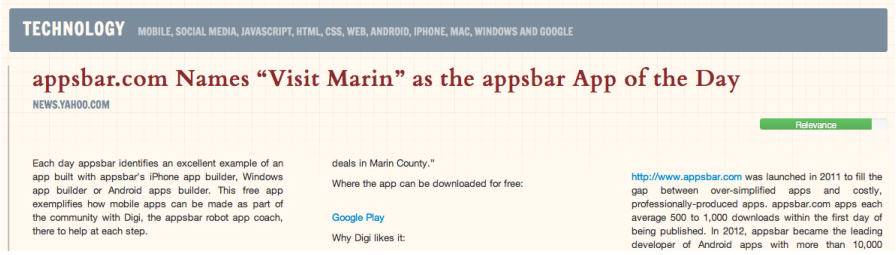


Figure 19: The figure shows an article which has been deemed by the system an article about technology, but is only listed under a business press releases tag by Yahoo.

The fact that most of the used content providers list their feeds under several topics and supplies the articles in all matching feeds could explain this. To handle this, tags were added to the article whenever a URL from the feed was already visited. This did, however, not solve all problems because many content providers listed articles under a different URL for each topic. Looking at the average similarities it is seen that the "World" section is near 0.5 in both matches and mismatches. This could be explained by its definition. It consists mostly proper nouns of parts of the world which are hard to match with proper nouns of countries or cities, that may more often be named in articles.

The similarity between articles is semantically analysed in same way and it is therefore expected to perform equally or better. Better, because the full set of information is available in the article and thereby the analysis has a better chance of extracting keywords, perform enriching and extract entities. It should be noted that [Open Calais API] sometimes have problems with extracting entities from a set of keywords instead of sentences, and it could therefore be interesting to explore the possibilities of representing topics based on sentences instead.

In any case, it seems that this analysis is not accurate and a deeper analysis of the similarities in terms of numbers must be done in order to draw any conclusions. However, a qualitative assessment of the results is that the application rarely provides irrelevant articles according to the topics, and that it especially

performs well with recognition of entities in articles. And that is even for low similarities. Also, the application seems to be good at following the rules of placing articles that relate near each other, and thereby supplying them in more relevant context than just under the topics, as it is done in conventional newspapers. Finally, some quality of the method can be seen in that the same semantic analysis is done by AdSense and that [Bouras and Tsogkas] has produced promising results with the same algorithm.

It could be interesting to evaluate the precision and recall[20] points as described in [Esteban *et al.*, 2000]. This could be done by letting a number of test subjects organise a subset of the gathered articles into the given topics and then do an equal analysis as shown in Figure 19. Precision and recall evaluation have been done in [Diaz *et al.*] with categories represented by keywords, it shows poor results for categories of with represented by a few keywords.

[20] *Precision is the fraction of retrieved instances that are relevant, while recall is the fraction of relevant instances that are retrieved.*

In addition, a more comprehensive list of categories and their definitions could also be used. This could either be retrieved by the list of Google News categories[21] and a WordNet enriched list of key words from Google News list of suggested keywords[22] or by the root terms presented in [Abuzir and Vandamme]. These can later on be refined by information retrieved by the user behaviour in the system. The user behaviour would also make the system stronger as their relevance feedback can be used to remove false positive and as stated earlier, collaborative filtering to suggest false negatives. The latter could be backed by providing a higher relevance for articles with a subject with many available articles in recent time.

[21] `http://support.google.com/webmasters/bin/answer.py?hl=en&answer=42993`.

[22] `http://support.google.com/news/publisher/bin/answer.py?hl=en&answer=116037`.

In the application a user could be presented with an article he has seen before. [Billsus and Pazzani] suggests the nearest neighbour (NN) algorithm approach, using a TF-IDF similarity, to determine whether the story is already known, i.e. if the similarity to the NN of articles, that are detected to be already seen, is above a given threshold the article is also determined to be seen. This could be solved by keeping a library of read items and then match new items against this list and down prioritise them if their similarity is too high. However, user might be interested in reading many articles about the same story.

[Diaz *et al.*] raise a precision problem in finding relevant news within a single day. This can hopefully be handled by having the user specify in which period of time he wants news and maybe notify the user that solutions might be inaccurate if a limited period of time is chosen, or just provide only articles that are available.

The system does at this point not supply articles of relevance based on the users location. If geo targeting are introduced, using e.g. the users IP address or [Geolocation API], the system would be able to provide higher relevance according to words that suggest a position on the map. This could also be used to define the "World" section even better.

Furthermore, weights on keywords could be adjusted by a strength (or an uncertainty) of prediction is proposed by [Claypool *et al.*]. The uncertainty could be calculated on the basis of how many uses there are of a word, as provided by WordNet.

Finally, some articles presented with the same content over and over again and even contained the main menu of the sites. This was due to some bad results from the scraping using the [Readability API]. The problem with this is that the menu of news sites often holds keywords that relate to most sections and will therefore be assigned a good relevance to most sections. Because of this, these articles will show up more regularly than others, which disturbs the results from the compositions. Moreover, it could also explain some of the causes to the high relevance mismatch mean derived from the similarity analysis earlier. The problem could be solved with a better sanitising of HTML and standardising of the content. This would remove odd looking elements.

7.4 FUNCTIONALITY

In order to come closer to understanding what conventional newspapers does in terms of editorial rules, it could be interesting to analyse their component structure, e.g. using the algorithm presented in [Liu *et al.*]. It could also be interesting to establish a co-operation with newspaper editors to extract rules.

The asymptotic running time of the algorithm is $O(m + n^2cv)$, where m is number of articles provided to solve the problem with, n is number of variables in the problem, c is the number of constraints and v is the number of variables a constraint is defined on. This means that the running time is very much dependent on the problem definition, but also of the number of articles given. The function is actually only provided with a subset of the full library based on relevance to spare computation time, but the upper bound remains because it potentially could provide the full library, whereas in practise only a certain percentage constitutes the articles used. More crucially, the running time is polynomially dependent on the number of articles need in the section multiplied by the constraints and the number of variables a constraint is defined on. It is a high running time, but many actions can be taken towards optimising the algorithm, e.g. the use of local search techniques, like hill climbing and simulated annealing. Also, the very effective constraint propagation has not been implement, which can aid the algorithm in removing values that are inconsistent with the problem. However, the algorithm is guided to solve the problem both by the MCV heuristic and by the violations of the constraints. Moreover, it is possible to decompose and organise the constraints to aid the algorithm in solving the problem.

Furthermore, the CPL only works on finite domains and propagating through many values causes many iterations. Instead it could be interesting to explore a combination of values and ranges in domains. A range could be defined by a start value, a step size and an end value, e.g. the list $[0, 2, 4, 6, 8, 10]$ would be represented by $[0; 1]$ with step size 0.2. This way a whole range could be discarded in a propagation step by looking at its maximum and minimum value. Moreover, if the range holds a potential valid value it can be divided into smaller

ranges and their minimum and maximum values may be examined. This divide-and-conquer technique may continue until the search reaches atomic values, which is determined by the step size of the range. If some atomic values and ranges seems to fulfil the constraints they should be kept in the domain.

Moreover, as an answer to the stated hypothesis in section 1.4 on page 8, it was possible to personalise the editorial mix and solve the problem with CP and new rules can easily be added or existing ones can be changed. A by-product of the implementation was even a general purpose solver for combinatorial personalisation problems. The library may even be used for other purposes, as long as the problems can be modelled by means of values, variables, domains and constraints. This means that CP was indeed applicable to personalisation problems. The library should, furthermore, be understandable for developers to use it, even if they do not have insight in CP. Finally, it is prepared for implementation of constraint propagation and an automatic ordering of constraints, to optimise the running time for the solver.

8 DISCUSSION

This section will discuss the uses of the application and of the CPL.

8.1 APPLICATION

It was possible to provide a composition of personal articles that follows rules of the editorial mix, which in turn should provide a better reading experience of personal content on the web. The main difference from other news sites and this application is that it is the user that decides what is interesting and what is not, i.e the user has become the editor.

The displayed content could be optimised by looking at the user behaviour and learning from it. Collaborative filtering or clustering could be used to generate user groups to suggest targeted articles; "users that like some of the articles as you do likes this article". These technologies can also be used to improve the existing sections for a given user.

The application is, however, not bound to its domain of a newspaper. Any page that needs to serve a personalised composition of information on the Internet, can use this application. Or, this application could be used to gather information around the Internet, much as Wavii[23] does, but serve it under user defined topics, in a nicely presented layout and with a relevant reading flow. Examples of this are aggregation of news from social sites, information about tourist attractions to compose a vacation, or information about bargains that fit the individual user.

[23] https://wavii.com/ *aggregates different kinds of information from the Internet and displays it to the user.*

The first example could be just for personal uses, but the keyword analysis could provide substantial insight about user behaviour assembled into groups of e.g. interests, geographic location or age. The system could then be used as a tool to

access vital knowledge about trending behaviours to use for e.g. targeting adverts or developing products that satisfies user needs better. This information has the potential to create value.

The second example could be used by a travel agency to provide personally composed vacations, or just by the users themselves, where the travel agency provides the necessary information. In the case of planning a vacation there might be restrictions of which attraction to see first, the distance between them and the time frame. If the travel agency formalises the problem of planning a vacation the restrictions could be exposed to the user, who can state their preferences, and in turn get a promising vacation.

In the third example different kinds of online shops could engage in a network where the user could order anything. The system could then suggest different combinations of complimentary items, e.g. if the user needs to shop for a dinner party, the system could suggest combinations of ingredients for the dish the user searched for and propose a wine that is specifically good with the ingredients. The difference between this and other systems, is where other systems might show a long list of the same items, this system can be modelled to show only one of each kind based on similarity measures.

The application in principle could compose even a print copy of the newspaper, with the personalised sections and content and specific ads to go on the printed edition. User could then select how often the newspaper should arrive at the door and when its content should be generated.

The revenue flow could come from online auctions of targeted ads, using [Google AdSense]. However, it seems that AdSense scrapes the cached page and uses WordNet extract semantic meaning, possibly along with other techniques. And it is therefore not possible to serve keywords from topics and subjects to base the ads on, even though it would make them more precise. It is, however, possible to control what is shown on the page, which, in the application, contains targeted content. Central places between articles could be sold to the highest bid.

To attract users to the system [Google AdWords] could be used. Here it is possible to serve the extracted keywords to get tar-

geted ads on other pages, which might aid the awareness of the system. The optical being that the cost of the keywords served to Google AdWords, should be less than the revenue gained from Google AdSense. Firstly, it could be done by using WordNet to find synonyms or words with similar meaning, but that are cheaper. Secondly, by controlling which content to display for users optimising the revenue from Google AdSense. However, it should be considered that this might not be the intended use of Google's products.

8.2 CONSTRAINT PERSONALISATION LIBRARY

The automatic composition of items with metadata that should follow some rules, can be found in many personalisation problems – and even other problems as well. The possibilities for this library seems to be manifold, and it could be interesting to explore the full extend of the possibilities by making it available for everyone, in the hope that someone takes it and uses it in a completely new context that it was never intended for.

9 CONCLUSION

This paper solved the problem of personalising the editorial mix using Constraint Programming (CP). Different patterns in the editorial mix might emerge as new knowledge to the field is applied. However, the Constraint Personalisation Library makes it easy to extend the implemented solution with additional rules or changes to the existing ones. The fact that the library is implemented with a general purpose solver seems to make it possible to use it in many different contexts. The logic structure of how to define the problem should make it possible for other developers, that may not have knowledge of CP, to use the library in new contexts. The running time of the library is acceptable, but not optimal. However, it does make use of both probabilistic and logical approaches. The library is prepared for additional actions to optimise it and it is possible for the developer to define the problem to optimise the running time of the solver.

The implemented semantic analysis of articles does in practice seem to perform as it should, but a deeper analysis of this is needed in order to draw conclusions.

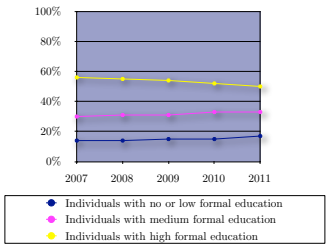
Part IV

APPENDIX

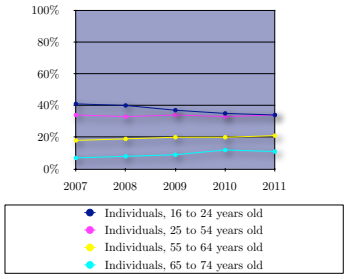
A USER NEEDS

This section will define the user needs for the application.

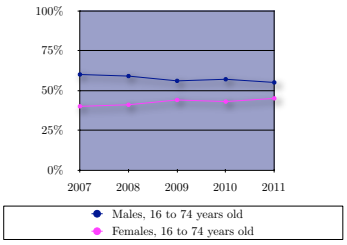
A.1 PERSONAS



(a) Individuals distributed by educational background.



(b) Individuals distributed by age.



(c) Individuals distributed by sex.

(d) Individuals distributed by occupation.

Figure 20: Eurostat: Individuals using the Internet for reading / downloading online newspapers / news magazines

In Figure 20 is shown the basis for the division into the following user groups:

The user groups provide the basis for the following personas.

Table 11: Eurostats distribution of individuals using the Internet for reading and downloading online newspapers and news magazines.

Description	**%**
student	35
employees, self-employed, family workers	31
unemployed	22
retired or other inactive	13
Description	**%**
high formal education	50
medium formal education	33
no or low formal education	17
Description	**%**
male	55
female	45
Description	**%**
16-24 of age	34
25-54 of age	34
55-64 of age	21
65-74 of age	11

A.1.1 *Thomas: student, medium formal education, male, age 21*

Thomas is 21 and a student at the Technical University of Denmark to be a bachelor of engineering in software. He is very interested in soccer and is therefore always updated on sports news. He reads about it online, newspapers and talks about it with friends. With big events he even likes to post it on Facebook. As a soon-to-be software engineer he has a natural thirst for news about technology, and he mainly reads these at home at the dormitory. wired.com, newz.dk, engadget.com, facebook.com computer, Samsung Galaxy Tab

A.1.2 *Laura: employed, high formal education, female, age 39*

Laura is 39 and is employed as a key account manager. She likes to be updated on strategies and economical status of rivalling companies. She is also very interested in politics and likes to discuss this subject with her friends. She reads economical news and likes to be updated on the run. b.dk, borsen.dk, twitter.com iPhone, iPad

A.1.3 *Marie: unemployed, no or low formal education, female, age 61*

Marie is 61 and a currently unemployed housekeeper. She spends her day looking for a job and taking care of her pet cat until her husband comes home. She mostly looks for the gossip sections or news about crime or big disasters. She also spends some time reading through the travelling guides as she dreams of going away with her husband. ekstrabladet.dk, bt.dk, nyhederne.tv2.dk computer, Lenovo IdeaPad A1

A.1.4 *Carl: retired or other inactive, high formal education, male, age 69*

Carl is a retired professor in psychology. He likes to discuss human behaviour and relation with his acquaintances and is very interested in cultural events. Therefore he often seeks the cultural sections and discussion fora to see what is going on. politiken.dk, aok.dk, dr.dk computer, iPad

A.2 SCENARIOS

A.2.1 *Thomas*

Thomas comes home after a day at the study, picks up his tablet computer and opens Editor from the desktop. Editor opens and shows him the front page where all the headlines stories are displayed. The main story is about a new version of the Android OS that has been released today and presses it to read more. The story opens in a full window display with quality images to match the articles. He reads the first section and feels satisfied with the amount of information, but wants to share the information on Facebook, so he clicks share button and writes a comment and posts it on his Facebook wall. He closes the article and returns to the front page. He sees a top story below the main story about Mr. Mærsk Mc-Kinney Møller who has died. It is not a story that falls into his key interests, but as the news is big he is satisfied that he got informed about it. Thomas feels like reading more about technology so he opens the menu and chooses the "Tech" section he has installed in the application. The section opens with a head line and a page number to let him know where in his paper he has navigated to and finds an article about a new multicore CPU technology. He has never been interested in CPU technology before, but finds this technology interesting after reading about it, so he opens the application settings and types in keywords about the technology under his "Tech" section to keep him updated about it. He also adjusts the ratio between general and personal news, to be less personal as he feels like he needs to broaden his horizon a bit with respect to news. He closes the settings menu

and Editor immediately starts updating the articles. Some new articles about CPU technology has been included amongst the articles in the "Tech" section after paging through the section and reading some of the most interesting articles he closes the application.

It could be nice if the key words of a story could be or is already highlighted, so he can click it and add it to his positive or negative list.

A.2.2 *Laura*

Laura is on the train on her way to a business meeting this morning and pulls out her tablet and sees she has one notification from Editor. She opens Editor to get updated on todays news. The front page is displayed and there are headlines from different top articles and a notification is shown in the corner. She presses the notification and the pages turns to show her the article, which opens in full screen. After reading it she wants to see todays headlines, so she presses the back button to return to the paper and presses the return to front page button and the paper turns pages to reach the front page. She scans the page to see if there is any big news about her rivalling companies. There is no breaking news, so she just turns the page to browse the content of todays paper. As she browses the "Politics" section of her paper she finds an article about the Prime Minister introducing a new bill about a toll ring around the capitol city. She chooses the article and it is shown in full screen. As she reaches the bottom of the article she sees the comments about it where her friends and most others are against it. She decides to join the discussion and posts a comment on the article wall. She also sees one of her friends has not commented on the article wall and decides to share the article with her as she thinks she would agree with her opinion. She presses the share button and chooses the Editor logo. A list of her friends is shown, some of them who has already read the article is greyed out, but the one she was looking for is not. So she chooses her and a notification is sent to her.

A.2.3 *Marie*

It is morning and Marie wants to check the news with her coffee in the couch, so she opens Editor from her tablet to get updated. The front page is displayed with a collection of stories as highlights of the content of the paper. It mainly contains stories about celebrities and a big disaster that has happened in japan, but there is also a story about a big political change, that she does not find interesting. So she goes to the settings menu and types in "politics" to add to her negative list. She also adjusts the personal/general news ratio to contain only personal news as she wants only news that is directed to her. She returns to the front page which is now free of political stories. Her newspaper contains many images and videos as she has set her graphical/textual content ratio more towards graphical content.

A.2.4 *Carl*

Sunday morning Carl wakes up, puts over the kettle to make a cup of coffee. While he waits for the water to boil he picks up his iPad and opens Editor to check the news. The front page opens with headlines from the different sections. There is a review article about a new show in the theatre. Carl presses the article and the system turns pages to the "Cultural" section of his newspaper and opens the article in full screen. Because the show gets good ratings he decides to order some tickets to him and his wife, which he does using the devices browser. After this he reopens Editor which opens in a display of the same article, as he left it. Carl pours his coffee and turn to the back page with the some crossword puzzles and some cartoons. He presses a puzzle that looks funny and it is displayed in a full page, where he can solve it. When he is done he returns to the page and chooses his regular cartoon to read.

A.3 BUSINESS CASE

A.3.1 *Need*

User value: personal quality and up-to-date stories enriched with quality images. This means that content providers should be chosen/verified. Same navigation as actual newspapers, but faster and with endless more content. Instantly up-to-date. Adaptive layout. Adjustable to individual user.

A.3.2 *Approach*

Personalised content in an editorial mix.

Constraint Programming: fast computation - good for optimal solutions, describes the generic solution in stead of how to solve or find it, easy to tailor the problem definition of the solution and adjust it and even let users make the adjustments - transparency.

Content providers can get to know their readers preferences better and improve the provided content.

A.3.3 *Benefit Per Cost*

Revenue flow: Content providers are paid. Income from advertisers (scattered [Ovesson and Wikström, p. 6-7]) and users. Income from selling user behaviour patterns and precise targeted commercials.

A.3.4 *Competition*

FlipBoard, Wired magazine, Zite and app with actual editors affiliated.

A.4 REQUIREMENTS

The above scenarios, user needs and business case led to the following requirements.

A.4.1 *Non-functional Requirements*

- "the clear overview of content, including a beginning and an end, the ease of use, typography and design" [Ihlström *et al.*, p. 7]
- both general and personal news (collaborate filtering solves that some news are not received, but are universally interesting [Díaz and Gervs])
- familiarity in design from printed paper [Ihlström *et al.*, p. 7]
- Design and layout from printed newspaper [Åkesson *et al.*]
- both images and videos - test
- a good ratio of graphical and textual - test
- front page should give a good overview of the content - test
- "news valuation, e.g. positioning of lead story" [Ihlström *et al.*, p. 7]
- mobility [Ihlström *et al.*, p. 7]
- continuous updates [Ihlström *et al.*, p. 7]
- "easy and intuitive navigation" [Ihlström *et al.*, p. 7]
- add video and sound [Ihlström *et al.*, p. 7]
- incorporate social community and social networks

A.4.2 *Functional Requirements*

A.4.2.1 *Calculations on number of columns*

iPad 1 and 2 screen size:

$197 \times 148mm$ or $1024 \times 768px$

International Herald Tribue (the global edition of the new york times): $\frac{398mm}{6} = 66.333333333mm$

Børsen (uses both 5 and 6 columns): $\frac{285mm}{5} = 57mm$ and $\frac{285mm}{6} = 47.5mm$

Information (5 columns 4 on the back): $\frac{285mm}{5} = 57mm$ (back $\frac{285mm}{4} = 71.25mm$)

Guardian (5 columns): $\frac{314mm}{5} = 62.8mm$

Politiken (6 columns): $\frac{392mm}{6} = 65.33mm$

Berlingske (4 columns): $\frac{285mm}{4} = 71.25mm$

Average on the most regular columns:

$\frac{66.3+57+57+62.8+65.3+71.3}{6} = 63.283333333mm$, i.e. 3.11 columns in landscape and 2.34 in portrait.

(Average on every column width:

$\frac{66.3+57+47.5+57+71.25+62.8+65.3+71.3}{8} = 62.30625mm$, i.e. 3.16 columns in landscape and 2.38 in portraint.)

$1200px$ screen:

$\frac{1200 \cdot \frac{197}{1024}}{63.28} = 3.65$ columns, where 197/1024 is px to mm ratio and 63.28 is the thinnest column width

column size in px:

$63.28 \cdot \frac{1024}{197} = 329px$

$62.31 \cdot \frac{1024}{197} = 324px$

average $= 326px$

- 2 columns in portrait and 3 in landscape
- "open, turn pages, chose article, read and return" [Ihlström *et al.*, p. 6]
- section headlines [Ovesson and Wikström, p. 6-7]
- article headlines
- article summaries / extracts [Díaz and Gervs]
- menu w. section headlines [Ovesson and Wikström, p. 8]
- page numbers [Ovesson and Wikström, p. 6-7]
- press "like" or key word based user profile (mark self or highlighted? right click to add): positive + negative list (keywords+categories [Abuzir and Vandamme], [Díaz and Gervs] and [de Buenaga Rodríguez *et al.*])
- full screen display of article
- organise into personalised sections
- opens in front page view (summery of newspaper 8 articles) [Ovesson and Wikström, p. 8]
- adjust variables
- share directly (grey out the ones who have read it)
- comment
- see friends comments
- "The presentation schema – headline, abstract, and text, together with a relevance value with respect to the user profile – rates the highest in terms of user satisfaction, and yet it is not the most frequent." [Diaz *et al.*]

- ability to search [Ihlström *et al.*, p. 7]
- Landscape + portrait [Ovesson and Wikström, p. 6-7]
- touch screen interaction [Ovesson and Wikström, p. 6-7]
- Functionality from online newspaper [Åkesson *et al.*]
- Name of columnist [de Buenaga Rodríguez *et al.*, p. 4]
- Transparency of implicit relevance feedback (see/modify current weights of categories) [de Buenaga Rodríguez *et al.*, p. 7]
- dynamic short-term + static long-term user profile [Abuzir and Vandamme], [Díaz and Gervs] and [de Buenaga Rodríguez *et al.*]
- relevance feedback [Abuzir and Vandamme], [Díaz and Gervs] and [de Buenaga Rodríguez *et al.*]

A.5 TEST RESULTS

This section sums up the test results in an unordered list.

- Touch friendly interaction
- Tools for changing the layout, like changing the font size and colour scheme
- The front page should give an overview
- View whole menu all the time
- Give suggestions to similar articles, i.e. more on this story, subject and topic
- List overview of headlines in top of sections
- Search within relevant articles. Search bar should be visible at all times.

- Indication of similarity on articles
- Archive possibility
- User feedback notated with “relevant” and “irrelevant”
- White space besides articles is not a problem
- General layout corrections
- Better visual division between articles
- More and larger images
- No need for general news, personalised news is enough
- Choose categories with topics to get started
- Ability to choose period show articles from
- Ability to choose when the newspaper should be generated, e.g. on Fridays to be read in the weekend
- Dividing columns into screens
- Social community implementation
- Visualisation of user behaviour
- Possibility to use it for research
- Get articles from magazine and newspaper subscriptions, e.g. by adding them to a specific section
- Play with the text and get it read out loud
- It has solved the problems that http://nyhederne.tv2.dk/[1] has, i.e. confusing layout, no overview and hard to navigate

[1] *The website of a Danish news channel.*

B NEWSPAPER TOPICS

Table 12: Topic specification

Name	Tags	Keywords (weights)
Technology	technology, science	technology (1.0), mobile (0.6), social_media (0.8), context-aware (0.3), Javascript (0.6), HTML_5 (1.0), CSS_3 (1.0), Web (0.9), Android (0.4), Mac_OSX (0.5), Windows (0.2), iPhone (0.5), Apple_Inc. (0.7), Google (0.5)
Science	science	science (1.0),software (0.6), software_technology (0.8), engineer (0.4)
Sport	sport	Sports (1.0), NFL (0.5), football (0.9), soccer (0.9), NBA (0.5), NCAA (0.5), NHL (0.5), baseball (0.5), golf (0.5), tennis (0.5)
Entertainment	entertainment	entertainment (1.0), art (0.5), celebrity (0.2), fashion (0.3), TV (0.2), Television (0.2), television_show (0.4), play (0.3), opera (0.2), cinema (0.7), theatre (0.3), read (0.7), hobby (0.5), fun (0.6), enjoy (0.5), laughter (0.5), social (0.2), cartoon (0.4), review (0.7), movie (0.7), book (0.5), music (0.9), picture (0.6)

Table 13: Topic specification (continued)

Name	Tags	Keywords (weights)
Politics	politics	politics (1.0), democrat (0.7), liberal (0.7), government (0.5), state (0.5), corporate (0.4), policy (1.0), authority (0.4), power (0.3), democracy (0.7)
Business	business, money	business (1.0), money (1.0), organization (0.7), trade (0.9), goods (0.5), service (0.4), stock (0.4), market (0.8), consumer (0.6), economy (1.0), profit (0.8), owner (0.4), administer (0.4), CEO (0.6), company (0.8), work (0.5), commercial (0.4), proprietorship (0.7), partnership (0.7), corporation (0.7), cooperative (0.4), debt (0.4), value (0.4), payment (0.5), bank (0.6), currency (0.5), account (0.3), savings (0.3)
World	world	world (1.0), international (1.0), Europe (0.6), Middle_East (0.6), Australia (0.6), New_Zealand (0.6), USA (0.4), Africa (0.6), Latin_America (0.6), Asia (0.6)
Health	health	health (1.0), wellness (1.0), fitness (1.0), diet (0.6), weight_loss (0.5), well-being (0.6), nutrition (0.7), parenting (0.5), health_care (0.6), pregnancy (0.5), cancer (0.3), allergies (0.6), asthma (0.6), family (0.7)

BIBLIOGRAPHY

[Abidi and Chong, 2004] Syed Sibte Raza Abidi and Yong Han Chong. Constraint satisfaction methods for information personalization. Technical report, Dalhousie University, Canada and Universiti Sains Malaysia, Malaysia, 2004.

[Abuzir and Vandamme, 2002] Yousef Abuzir and Fernand Vandamme. E-newspaper classification and distribution based on user profiles and thesaurus. Technical report, University of Ghent, 2002.

[Apt, 2006] Krzysztof R. Apt. *Principles of Constraint Programming*. Cambrige University Press, 2006.

[Billsus and Pazzani, 2000] Daniel Billsus and Michael J. Pazzani. User modeling for adaptive news access. *User modeling and user-adapted interaction*, pages 147–180, 2000.

[Bird *et al.*, 2009] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media, Inc., 2009.

[Bouras and Tsogkas, 2010] Christos Bouras and Vassilis Tsogkas. W-kmeans: Clustering news articles using wordnet. *Knowledge-Based and Intelligent Information and Engineering Systems*, pages 379–388, 2010.

[Bush, 1945] Vannevar Bush. As we may think. *The Atlantic Monthly*, 1945.

[Claypool *et al.*, 1999] Mark Claypool, Anuja Gokhale, Tim Miranda, Pavel Murnikov, Dmitry Netes, and Matthew Sartin. Combining content-based and collaborate filters in an online newspaper. Technical report, Worchester Polytechnic Institute, Massachusetts, 1999.

[CSS3 Multi Columns, 2012] Css3 multi columns. http://www.w3schools.com/css3/css3_multiple_columns.asp, 2012.

[de Buenaga Rodríguez *et al.*, 2004] Manuel de Buenaga Rodríguez, Manuel J. Maña López, Alberto Díaz Esteban, and Pablo Gervás Gómez-Navarro. A user model based on content analysis for the intelligent personalization of a news service. Technical report, Halmstad University, 2004.

[Diaz *et al.*, 2001] Alberto Diaz, Pablo Gervas, Antonio Garcia, and Inmaculada Chacon. Sections, categories and keywords as interest specification tools for personalised news services. *Online Information Review, OIR*, 2001.

[Dublin Core Metadata Element Set, Version 1.1, 2012] Dublin Core Metadata Initiative. *Dublin Core Metadata Element Set, Version 1.1*, 2012.

[Díaz and Gervs, 2005] Alberto Díaz and Pablo Gervs. Personalisation in news delivery systems: Item summarization and multi-tier item selection using relevance feedback. Technical report, Universidad Complutense Madrid, 2005.

[Eirinaki and Vazirgiannis, 2003] Magdalini Eirinaki and Michalis Vazirgiannis. Web mining for web personalization. Technical report, Athens University of Economics and Business, 2003.

[Esteban *et al.*, 2000] Alberto Esteban, Pablo Gómez-Navarro, and Antonio Jiménez. Evaluating a user-model based personalisation architecture for digital news services. In *Research and Advanced Technology for Digital Libraries*, pages 259–268. Springer, 2000.

[Eurostat, 2012] Eurostat. http://epp.eurostat.ec.europa.eu/tgm/table.do?tab=table&init=1&language=en&pcode=tin00097&plugin=0, 2012.

[Flipboard, 2012] Flipboard. http://flipboard.com/, 2012.

[Genette, 1997] G. Genette. *Paratexts: Thresholds of interpretation*. Cambridge Univ Pr, 1997.

[Geolocation API, 2012] Geolocation api. http://dev.w3.org/geo/api/spec-source.html, 2012.

[Google AdSense, 2012] Google adsense. www.google.com/adsense, 2012.

[Google AdWords, 2012] Google adwords. https://adwords.google.com, 2012.

[Google Reader, 2012] Google reader. http://www.google.com/reader, 2012.

[Hane, 1999] Paula J. Hane. Beyond keyword searching—oingo and simpli.com introduce meaning-based searching. *Information Today, Inc.*, 1999.

[Haskins, 1965] Jack B. Haskins. The editorial mix: One solution to a magazine editor's dilemma. Technical report, Journalism & Mass Communication Quarterly, 1965.

[Holmqvist *et al.*, 2003] K. Holmqvist, J. Holsanova, M. Barthelson, and D. Lundqvist. Reading or scanning? a study of newspaper and net paper reading. *Mind*, 2(3):4, 2003.

[Holsanova *et al.*, 2006] J. Holsanova, H. Rahm, and K. Holmqvist. Entry points and reading paths on newspaper spreads: comparing a semiotic analysis with eye-tracking measurements. *Visual communication*, pages 65–93, 2006.

[Ihlström *et al.*, 2004] Carina Ihlström, Maria Åkesson, and Stig Nordqvist. From print to web to e-paper - the challenge of designing the e-newspaper. Technical report, Halmstad University, 2004.

[JavaScript Web Worker, 2012] Javascript web worker. http://www.whatwg.org/specs/web-apps/current-work/multipage/workers.html, 2012.

[jQuery Masonry, 2012] jquery masonry. http://masonry.desandro.com, 2012.

[jQuery Waypoints, 2012] jquery waypoints. http://imakewebthings.com/jquery-waypoints, 2012.

[Kress and Van Leeuwen, 2006] G.R. Kress and T. Van Leeuwen. *Reading images: The grammar of visual design*. Psychology Press, 2006.

[Liu *et al.*, 2001] Fei Liu, Yupin Luo, and Masataka Yoshikawaf Dongcheng Hu. A new component based algorithm for newspaper layout analysis. Technical report, Tsinghua University, Beijing, 2001.

[Mobasher, 2007] Bamshad Mobasher. Data mining for web personalization. In *The Adaptive Web*, Lecture Notes in Computer Science, pages 90–135. Springer Berlin / Heidelberg, 2007.

[Nilsson, 1999] Jørgen Fischer Nilsson. A conceptual space logic. Technical report, Department of Information Technology Technical University of Denmark, 1999.

[Oin, 2000] Oingo inc. launches adsense; meaning-based application boosts ad click-through rates by four times over current methods. *Business Wire*, 2000.

[Open Calais API, 2012] Open calais api. http://www.opencalais.com/calaisAPI, 2012.

[Ovesson and Wikström, 2005] Fredrik Ovesson and Kristin Wikström. From visions to specification - using user designed mock-ups for envisioning user requirements for the future e-newspaper. Technical report, Halmstad University, 2005.

[Perkowitz and Etzioni, 2000] Mike Perkowitz and Oren Etzioni. Adaptive web sites. *Commun. ACM*, pages 152–158, 2000.

[Readability API, 2012] Readability api. http://www.readability.com/developers/api, 2012.

[Russell and Norvig, 2003] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Mordern Approah*. Prentice Hall, 2nd edition, 2003.

[Russell and Norvig, 2010] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Mordern Approah*. Prentice Hall, 3nd edition, 2010.

[Salton *et al.*, 1975] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. pages 613–620, 1975.

[Schrijver,] A. Schrijver. *Combinatorial optimization*. Springer.

[Tab, 2011] How people use tablets and what it means for the future of news. *PEW Project for Excellent Journalism*, 2011.

[Tidwell, 2010] J. Tidwell. *Designing Interfaces*. O'Reilly Media, Inc., second edition, 2010.

[Twitter Bootstrap, 2012] Twitter bootstrap v. 2.0. http://twitter.github.com/bootstrap/, 2012.

[Vossen, 2005] M.P.H. Vossen. Local search for automatic playlist generation. Master's thesis, Technische Universiteit Eindhoven, 2005.

[Wired Magazine, 2012] Wired magazine. http://itunes.apple.com/dk/app/wired-magazine/id373903654, 2012.

[WordNet Interface Documents, 2012] Wordnet interface documents. http://nltk.googlecode.com/svn/trunk/doc/howto/wordnet.html, 2012.

[WordNet, 2012] Wordnet. http://wordnet.princeton.edu, 2012.

[Åkesson *et al.*, 2005] Maria Åkesson, Carina Ihlström, and Jesper Svensson. How would you like your e-newspaper? – converging the best from two worlds. Technical report, Halmstad University, 2005.