Sommelier: A Recommender System

By Peter Chamberlin
May 1, 2013

BSc Information Systems and Management Project Report,
Birkbeck College, University of London.

Abstract

This is the paper's abstract ...

Contents

1 Introduction

1.1 Online Recommender Systems

Since their origin in the mid-1990s with systems such as Tapestry [**?**] and GroupLens [**?**], recommender systems have become ubiquitous on the World Wide Web, being employed by some of the worlds largest online businesses as core parts of their offering.

The growth of the Web has given companies the ability to gather unprecedented amounts of data about their users' preferences, both explicitly collected and inferred from their behaviour, while at the same time enabling them to reach users for less cost more often than ever before.

Amazon's system of product recommendations using item-to-item collaborative filtering is regarded as a "killer feature"[**?**], and is one of the defining features of the Amazon website. The importance of recommendations to Amazon is reflected in their stated mission, "to delight our customers by allowing them to serendipitously discover great products"[**?**].

Another company which, like Amazon, is synonymous with recommender systems is Netflix, an "Internet television network" [**?**]. In October 2006 Netflix launched "The Netflix Prize", a competition with a $1,000,000 Grand Prize on offer for any team which could beat their own Cinematch recommender system by "at least 10%" accuracy over a fixed set of data [**?**]. In 2009 the prize was awarded to the BellKor's Pragmatic Chaos team, who had improved on Netflix's own system by 10.06% [**?**].

As well as online stores using recommender systems to recommend products a user might be interested in, there are systems ...

1.2 Aims and Objectives

My interest in recommender systems is founded in their variety and ubiquity. It occurred to me that I encounter, and am the subject of, dozens of these systems in my everyday life. Whether it's Twitter or Facebook recommending interesting interesting people or content to me, Amazon recommending me a book or film, or even a supermarket targeting special offers to me, I interact with recommender systems all the time. I am fascinated both by how these systems work theoretically and by how they are implementated in practice. As such the core objective of my project is to implement a recommender system for a web or mobile application.

I have kindly been permitted to freely use data from Decanter.com's wine reviews database [**?**] for my project. The sphere of wine recommendations is particularly interesting; wine is at first glance a narrow subject, but it is a

nuanced one. Among oenophiles there is an emphasis on personal taste and a strong tradition of rating and grading.

In this project I aim to build a recommender system based on Decanter.com's wine database, identifying and overcoming the challenges associated with the implementation of a real-world recommender system.

The most important aspect of any recommender system may be considered to be the quality of its recommendations, and I intend to focus very strongly on recommendation quality. Nevertheless I do not want to overlook the practical implementation of the system. I aim to produce a system satisfactory in both its recommendations and its performance as a web service, with a robust and elegant implementation in code.

I will not build any kind of graphical interface for the system, but will instead provide a machine readable API, and where necessary batch scripts and command line tools for preparing and manipulating data.

I aim to satisfy several of the most common use cases for wine recommendation, including user-item, item-item and user-user recommendations.

2 Literature Review

2.1 Recommender Systems

Although the term *recommender system* was not coined until 1997 by Resnick and Varian [**?**], the Tapestry system of 1992 [**?**] is widely recognised as the first of the kind [**?**]. The creators of Tapestry coined the term *collaborative filtering* to describe their method of recommendation, which is based on the principle that if two users rate a number of the same items in a similar manner, then it can be assumed that they will rate other new items similarly [**?**].

Su and Khoshgoftaar point out that although collaborative filtering has been widely adopted as a general term to describe systems making recommendations, many such systems do not explicitly collaborate with users or exclusively filter items for recommendation [**?**]. In fact the term recommender system itself was coined by Resnick and Varian in response to the inadequacy of collaborative filtering for describing the plurality of techniques that were beginning to become associated with it. Recommender system is intentionally a broader term, describing any system that "assists and augments [the] natural social process" of recommendation [**?**].

In his 2002 survey of the state of the art in recommender systems, Robin Burke presents five categories of recommender, including collaborative filtering as one [**?**]. I have reproduced his table of recommendation techniques in Table **??**. Burke presents five main categories of filtering technique: collaborative, *content-based*, *demographic*, *utility-based* and *knowledge-based* [**?**].

These five kinds of system are classified using three properties: *background data*, *input data* and *process* [**?**]. Background data is that which exists before and independant of the recommendation, such as previous stated preferences of a group of users U for a set of items I. Input data is that which is considered by the system when making recommendations, such as the ratings of an individual u of items in I. Process is the method by which recommendations are arrived at by application of the input data and the background data [**?**]. These three aspects provide a good lens through which to compare the different approaches.

2.1.1 Collaborative Filtering

Collaborative filtering is "the technique of using peer opinions to predict the interest of others" [**?**], and uses the ratings of a set of users U over a set of items I as background data, and the ratings of each individual user u of items in I as input data. The process of recommendation is to identify

Table 1: Recommendation Techniques, reproduced from Burke, 2002 [?]

Technique	Backgroud	Input	Process
Collaborative	Ratings from U of items in I.	Ratings from u of items in I.	Identify users in U similar to u, and extrapolate from their ratings of i.
Content-based	Features of items in I.	u's ratings of items in I.	Generate a classifier that fits u's rating behaviour and use it on i.
Demographic	Demographic information about U and their ratings of items in I.	Demographic information about u.	Identify users that are demographically similar to u, and extrapolate from their ratings of i.
Utility-based	Features of items in I.	A utility function over items in I that describes u's preferences.	Apply the function to the items and determine i's rank.
Knowledge-based	Features of items in I. Knowledge of how these items meet a user's needs.	A description of u's needs or interests.	Infer a match between i and u's need.

similar users to u in U, and then to infer their preferences for items in I based on the preferences of those similar users [?].

In 2002 Burke described collaborative filtering as the most widely used and mature of these types [?], citing GroupLens [?] and Tapestry [?] as important examples of such systems, and from my observations of more recent literature that remains the case. Collaborative filtering is still certainly among the most widely used of these techniques, with Su and Khoshgoftaar describing a vast array of cutting-edge collaborative filtering-based systems in their 2009 survey [?].

Even in its most basic form there are many potential methods for measuring similarity between users in a collaborative filtering system. The most simple are such distance metrics such as Manhattan distance or Euclidean distance [?]. One of the most commonly used, even in very advanced systems,

is Pearson correlation coefficient [**?**].

There are a number of significant issues associated with the application of pure collaborative filtering can which make it difficult to apply successfully without an auxiliary strategy:

- Early rater problem, whereby an item entering the system with no ratings has no chance of being recommended [**?**].
- Sparsity problem. Where there is a high ratio of items to ratings it may be difficult to find items which have been rated by enough users to use as the basis for recommendation [**?**] [**?**].
- Grey sheep, which are users who neither conform nor disagree with any other group in a significant way, making it very difficult to recommend items for them [**?**] [**?**].
- Synonymy, whereby identical items have different names or entries. In this case the collaborative filtering systems are unable to detect that they are the same item [**?**].
- Vulnerability to shilling, where a user or organisation might submit a very large number of ratings to manipulate the recommendability of items in their own interest [**?**].

Much of the variation between collaborative filtering techniques described by Su and Khoshgoftaar [**?**] can be attributed to efforts by system developers to minimise the impact of one or more of these problems by introducing auxiliary methods, and it is testament to the power of collaborative filtering that these efforts are made.

2.1.2 Content-based

In content-based filtering systems the features of items in I form the background data, and the user u's ratings serve as the input data. The process of recommendation depends on building a classifier that can predict u's rating behaviour in respect of an item i based on u's previous ratings of items in I [**?**].

Typically...

2.1.3 Demographic

Demographic recommender systems use demographic information about users U and their ratings in I as background data, with demographic information

about u as the input data. The recommendation process depends on matching u with other demographically similar users in U [**?**].

2.1.4 Utility-based

Utility-based systems use features of items in I as their background data, and depend on a utility function representing u's preferences in order to arrive at recommendations. The process is the application of the function for u to the items I [**?**].

2.1.5 Knowledge-based

Knowledge-based systems, like utility-based systems, draw on the features of items in I as their background data. As input data they require information about u's needs. The process is to infer a need for one or more items in I [**?**].

The most widely Collaborative filtering is a process whereby a user's preferences for items are inferred by the comparison of their previous preferences with the preferences of others. In itself this is a fairly straighforward principle, the comparison of two vectors of items and ratings or preferences, but there are a plurality of approaches. Burke, 2002 [**?**],

Content-based, like collaborative filtering, builds up a long term profile of a user's interests and preferences [**?**].

— Movielens data sets —

Developed around the same time as Tapestry, GroupLens [**?**] was another influential early recommender system. The GroupLens' research lab, founded in 1992, is still active at the University of Minnesota, and continues to research the field of recommender systems [**?**].

— off the web (?) — on the web...

– What are the methods employed in recommender systems?

– Collaborative Filtering - User-based filtering - Item-based filtering

– Content-Based Filtering - Variants

– Characteristics of the domain

— Cold start problem — Sparsity problem — ... etc.

Recommending Wines.

Recommender systems for wines are not a new idea, being typical of the kind of item many systems are designed to recommend. Burke developed the VintageExchange FindMe recommender system in 1999 [**?**], and there is at

least one patent pending with the WIPO for a wine recommender system, which appears to be conceived as a point of sale aid [**?**].

Burke's FindMe, a knowledge-based recommender system, "required approximately one person-month of knowledge engineering effort"[**?**] in order to perform well, as such systems are required to hold knowledge of the importance of given product features [**?**].

Another wine recommender system is the Tetherless World Wine Agent (TWWA) [**?**]. The TWWA project is primarily concerned with knowledge representaion and the Semantic Web, presenting a common and collaborative ontology for wine with which users can share wine recommendations across their social networks [**?**]. The system does not automatically tailor recommendations to users, although this is stated as a target for future work [**?**].

Evaluating Recommender Systems

!! MAE, NEtflix RMSE etc.

!! Shani, Gurawander (M$): evaluating recommender systems

!! Also: McNee et al. "How Accuracy Metrics Have Hurt Recommender Systems"

3 Development Method

The two main aims of my project were to create a recommender system for wines, and to implement an API for that system which might act as a service back end for a web application.

3.1 Python

My first enquiries into recommender systems included looking at Segaran's code examples in *Collective Intelligence* [**?**], where the language he uses for his implementations is Python. Other authors on recommender systems also use Python, such as [**?**]. Looking into the language further it became apparent that there are many tools available to the Python programmer that are particularly useful for this kind of system, such as the libraries Scipy[**?**], Numpy[**?**] and, for natural language processing, NLTK[**?**].

For the most part my system would suit the stateless, non-persistent nature of a Python web application. The only concern in this regard would be that I would need to recreate objects in memory from scratch with each request rather than persist them as I might using another language, such as using Java with the JPA[**?**]. Should the lack of persistence prove problematic down the line I looked into the possibility of using a persistence mechanism such as Memcache[**?**] to serve this purpose, and found that there is wide support for such a solution using Python and Flask[**?**].

With Python there is also a solid heritage of web application frameworks, such as Django[**?**] which is widely used for enterprise websites. I felt that Django might be a bit too fully featured for my purposes however [WHY?], and instead decided to use the framework Flask[**?**], which is developed with small services in mind and would be ideally suited to my purposes.

3.2 MySQL

Originally I had envisaged a system backed by a PostgreSQL[**?**] RDBMS, but having received the Decanter.com data as a MySQL[**?**] database it did not seem, comparing the two systems, that there would be any significant benefit migrating the data to PostgreSQL. Both are widely used in production, and have similar feature sets. For a short time I considered using a "NoSQL" database such as MongoDB[**?**] for my project, but decided against such a solution, recognising that such document-oriented systems are not ideal when joining between tables in the way that I would need to for my wines and tasting notes. It seemed that an RDBMS was ideally suited to the purpose, and there was no reason why that shouldn't be MySQL.

3.3 GitHub

In order to back-up and version my code I chose to use GitHub[**?**], which is a web service providing Git version control. I chose to use GitHub for my notes and project files also, so that my project was stored and versioned in its entirety in a private repository on GitHub.

3.4 Methodology

It was clear early in the planning for this project that I would be doing a large amount of experimental programming, with the recommendations the system makes being a work in progress.

4 System

4.1 Data Cleanup

!!! Priorities: retain as many ratings and notes as possible. Don't worry about attributes.

The data source I have used for my project is the wines database belonging to Decanter.com[**?**]. The database contains nearly 40,000 professional ratings and tasting notes for wines from as far back as 1986, featuring vintages as far back as 1917.

The original database is highly inconsistent, displaying a mixture of design approaches and a variable quality of data. This is consistent with the fact that the database has been developed over a long period of time by a number of different developers with varying levels of skill, and that wine journalists making entries into the database have taken a number of idiosyncratic approaches to data entry.

Nevertheless I considered there to be a great deal of useful and interesting information in the database, with it to contain usable ratings and/or tastings for over 33,000 wines.

4.2 The Decanter Database

In the original database the WineInfo table is a mixture of foreign keys joining to very small tables, such as WineInfo.type_id joining to WineType.id where WineType is a table with only two attributes. This approach, stiving for a high degree of normalisation, contrasts with the fact that the same table also has the attribute second_wine, as a string which only holds data in 450 of the 38762 entries in the table.

4.3 Creating The Sommelier Dataset

For the new Sommelier database, Figure **??**, I decided to denormalise [explanation/citation needed!] the wine data. This enables the data to be queried with fewer joins, maximising the simplicity and execution speed of the queries [citation needed] used for data access. Denormalization makes data integrity more difficult to maintain however, as there are potentially a large number of records to update for any change in a duplicated value. In this case an appellation or sub-region name changing might require thousands of records to be amended. Creating and editing wines is not a requirement of my system though, so for the purposes of this project the wine and tasting data is static and will not be subject to updates. For this reason the duplication of

Figure 1: Decanter.com Database

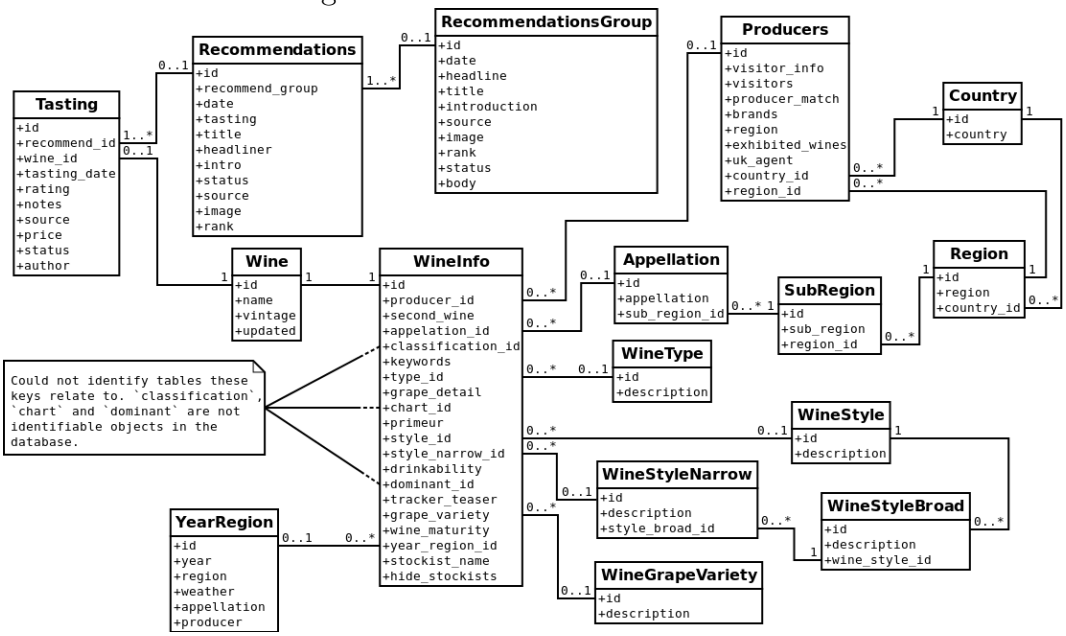
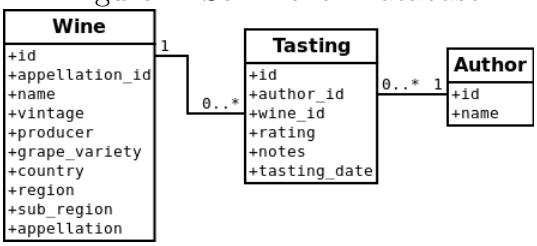


Figure 2: Sommelier Database



data within the Wine records is not problematic. In a real world setting this issue may need to be revisited.

Much of the data from the original database was disregarded entirely.

The tables WineStyleNarrow and WineStyleBroad contained generic text descriptions for wine ("rich and creamy", "crisp and tangy" etc.). I initially considered this to have potential for migration into tag data which I could reuse as part of my filtering. Unfortunately less than 6435 of the records in WineInfo had non-null values for their style_narrow_id field, and only 3397 of these had corresponding records in the Tasting table. This figure was only around 10% of the number of wines I expected the Sommelier database to contain so I decided that the WineStyle* tables were probably not worthwhile to migrate.

The WineType table was ignored because no wines corresponded to it; no WineInfo.type_id record matched any WineType.id.

The TasterRating and TastingId tables were discarded because they only referenced 1158 of the records in Tasting, too small a proportion of the Tasting table to make it worthwhile migrating them into the dataset.

4.4 The Author Problem

The biggest shortcoming of the dataset is that the author of a tasting note is often not recorded. The number of wines with notes and known authors is only 1401, with there being 18 named authors on the system.

Table **??** shows the distribution of tastings amongst authors, only 5 of which have tasted and rated more than 100 wines in the database.

Table 2: Authors of tasting notes and ratings

Author	Wines tasted	Wines also tasted by another
Amy Wislocki	28	1
Andrew Jefford	105	38
Beverley Blanning MW	13	-
Carolyn Holmes	1	-
Christelle Guibert	119	9
Clive Coates MW	6	-
David Peppercorn	44	-
Gerald D Boyd	7	-
Harriet Waugh	250	23
James Lawther MW	226	21
John Radford	2	-
Josephine Butchart	24	1
Norm Roby	4	-
Rosemary George MW	6	-
Serena Sutcliffe	31	15
Stephen Brook	19	3
Steven Spurrier	497	53

In some cases an author's initials or full name are recorded within the text of a tasting note. I decided that extracting and making use of these was impractical given the time constraints of this project.

DESCRIBE DATA SETS BEFORE AND AFTER

Table 3: Matrix of authors with wines tasted in common

Author	SS	JL	JB	SB	CG	SS	HW	AJ	AW
Steven Spurrier (SS)	-	6	1	1	7	0	15	30	1
James Lawther MW (JL)	6	-	0	0	0	15	0	0	0
Josephine Butchart (JB)	1	0	-	0	0	0	0	0	0
Stephen Brook (SB)	1	0	0	-	0	0	1	1	0
Christelle Guibert (CG)	7	0	0	0	-	0	1	5	0
Serena Sutcliffe (SS)	0	15	0	0	0	-	0	0	0
Harriet Waugh (HW)	15	0	0	1	1	0	-	10	0
Andrew Jefford (AJ)	30	0	0	1	5	0	10	-	0
Amy Wislocki (AW)	1	0	0	0	0	0	0	0	-

THE SOMMELIER DATASET

Having analysed the dataset and conceived an ideal schema, I needed to decide what the criteria to apply when extracting my new dataset from the source data.

Given that the purpose of the dataset is social recommendations, the first decision I made was to discard any wines without both tasting notes and a rating, whether.

4.5 Making Recommendations

Wine attributes, vintage, grape variety, region etc., are of limited use for making recommendations. Take a user who has rated a red Bordeaux wine highly. It is not interesting to that user for a system to simply recommend them other highly-rated red Bordeaux wines, which is what a system will do if it looks for items with similar attributes. It is likely that any drinker who enjoys Bordeaux wines will recognise that such wines are of a type, with geography, grape varieties and production processes in common. Thus the user will be capable of "recommending" Bordeaux wines to themselves, and will not have very much difficulty sourcing ones which are highly rated. They do not need a recommender system for that.

It is recommendations in spite of the attributes of the subject item that are of real interest. Recommending a Syrah from Chile's Colchagua Valley to someone who rated a red Bordeaux wine highly might be of more interest. It is relatively more likely that a user is unaware of the fine Syrah wines from that region, and that makes it a much more interesting recommendation; potentially a good one.

Similarly a recommendation of another Bordeaux red wine might be good, as long as it were possible to establish a commonality of appeal for that particular drinker other than that the wine is similar in attributes. For example there are many delicious Pauillacs from 2005, made of the same grape varieties in the same manner; finding the one which is most interesting to a given user is not aided by looking at its grape variety or appellation as they are identical. The appeal and interest of a wine recommendation lies in qualities beyond the item's attribute profile.

So, by disregarding the attributes of the wines it may be possible to make more interesting recommendations.

A corrollary benefit of this disregard for wine attributes is that I am able to make use of far more of the Decanter.com data, many of the wines within which have incomplete attributes.

TODO appendix of data

1. select a.name, count(t.id) from author a join tasting t on a.id = t.author_id where 0 ¡ (select count(*) from tasting t2 where t2.winei_id = t .wine_id and t2.author_id ¡¿ 0 and t2.author_id ¡¿ t.author_id) group by a.id; +————————+———–+ — name — count(t.id) — +————————+———–+ — Steven Spurrier — 53 — — James Lawther MW — 21 — — Josephine Butchart — 1 — — Stephen Brook — 3 — — Christelle Guibert — 9 — — Serena Sutcliffe — 15 — — Harriet Waugh — 23 — — Andrew Jefford — 38 — +————————+———–+

2. select a.name, count(t.id) from author a join tasting t on a.id = t.author_id where 0 ¡ (select count(*) from tasting t2 where t2.wine_id = t.wine_id and t2.author_id = N) group by a.id;

4.6 Experimentation

In Chapter 2 of Collective Intelligence[**?**], Segaran details basic methods for user- and item-based collaborative filtering. Following the guidelines from this chapter I recreated Segaran's recommendation methods and applied them to my dataset.

N.B.

Table data obtained by:

from lib import recommendations authorsims = recommendations.getAuthorSimilarities()

4.7 Implementation

Having so few wines tasted by more than one author in the dataset, and even fewer tasted by more than two, it was clear that my system needed to cope extremely well with sparsity. Su et al. [**?**] show their ...

Table 4: Pearson Similarity of Authors

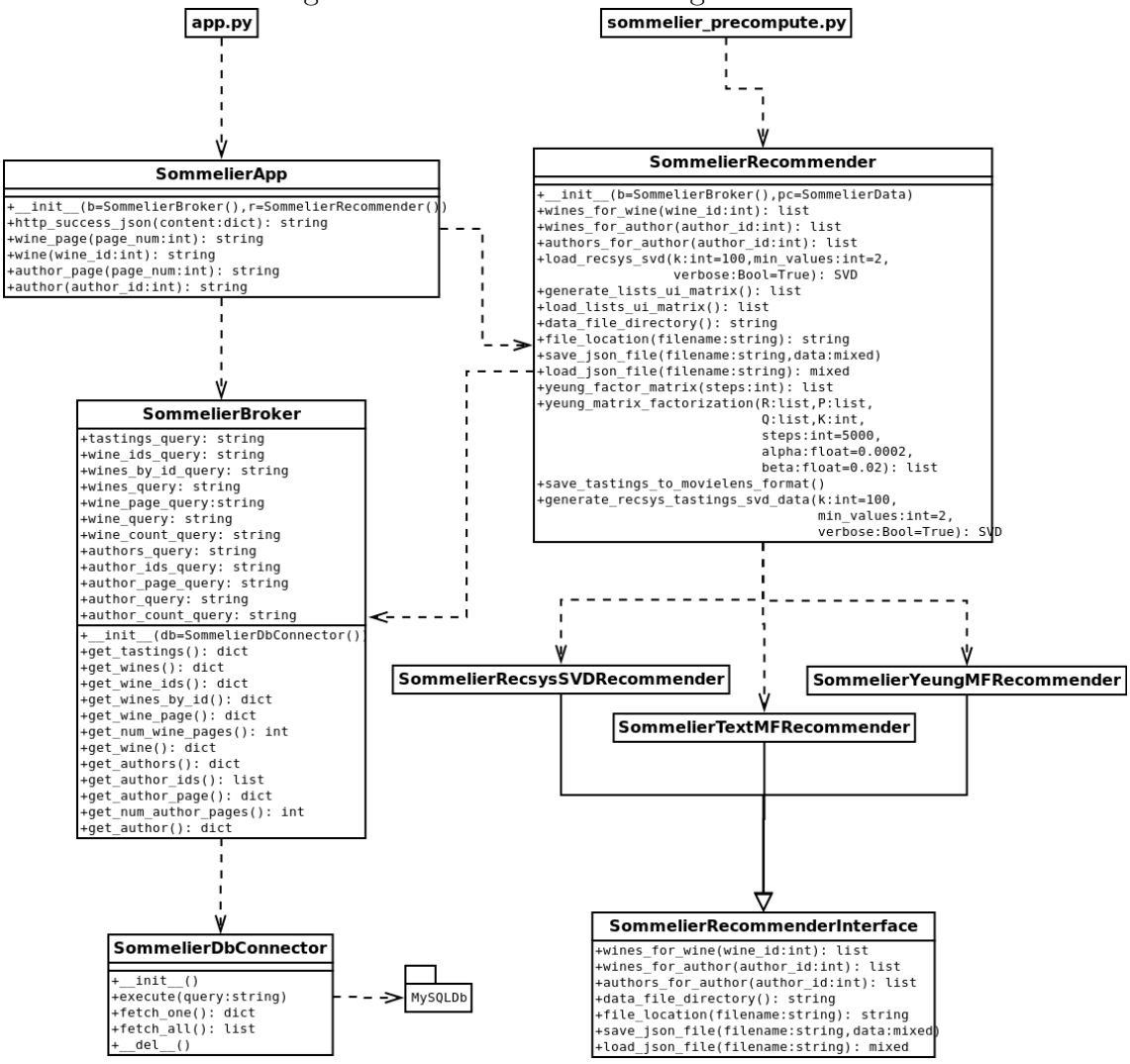
Author	SS	JL	JB	SB	CG	SS	HW	AJ	AW
Steven Spurrier (SS)	-	0.58	0.0	0.0	0.0	-	0.67	0.49	0.0
James Lawther MW (JL)	0.58	-	-	-	-	0.22	-	-	-
Josephine Butchart (JB)	0.0	-	-	-	-	-	-	-	-
Stephen Brook (SB)	0.0	-	-	-	-	-	0.0	0.0	-
Christelle Guibert (CG)	0.0	-	-	-	-	-	0.0	0.0	-
Serena Sutcliffe (SS)	-	0.22	-	-	-	-	-	-	-
Harriet Waugh (HW)	0.67	-	-	0.0	0.0	-	-	0.71	-
Andrew Jefford (AJ)	0.49	-	-	0.0	0.0	-	0.71	-	-
Amy Wislocki (AW)	0.0	-	-	-	-	-	-	-	-

!!! Would have been quixotic to attempt a complex approach involving text mining of tasting notes without first attempting to approximate the state of the art in imputation of preferences using matrix factorization, or a Bayesian approach...

!!! I decided firstly to imitate Su et al.'s [**?**] IDN-CF (eBMI) method (imputed densest neighbours collaborative filtering using extended Bayesian multiple imputation). This approach has been shown to mitigate the problem of sparsity, requiring very few neighbours to obtain its best recommendations compared to other methods.

!!! The IDN-CF (eBMI) implementation only utilizes the rating component of a tasting note, whereas I had set myself the objective of finding a use for the tasting note itself. Initially it occurred to me that I could build a Bayesian classifier, or similar, to identify ... Hu and Zhou (2008) etc.

Figure 3: Sommelier Class Diagram



5 Testing and Evaluation

How well does the system work? Details of testing and evaluation of the system...

6 Conclusion

Was the project successful?

7 Review

Review / reflections of the project on a personal level. What has been achieved? What were the problems, and how were they overcome?

Lessons learnt...- Data cleanup very time consuming - Literature vast -¿ plural techniques for recommendation: very difficult to work out what strategy is best for given situation.

References

[1] Burke, R., *The Wasabi Personal Shopper: A Case-Based Recommender System*, 1999. Submitted to the 11th Annual Conference on Innovative Applications of Artificial Intelligence.

[2] Burke, R., *Integrating Knowledge-Based and Collaborative-Filtering Recommender Systems*, 1999. In: Artificial Intelligence for Electronic Commerce: Papers from the AAAI Workshop (AAAI Technical Report WS-99-0 1), pp.69-72.

[3] Burke, R., *Knowledge-Based Recommender Systems*, Encyclopedia of Library and Information Systems, 2000. Marcel Dekker.

[4] Burke, R., *Hybrid Recommender Systems: Survey and Experiments*, User Modeling and User-Adapted Interaction, Volume 12 Issue 4, November 2002, Pages 331 - 370. Kluwer Academic Publishers: Hingham, MA, USA

[5] Debnath, Souvik and Ganguly, Niloy and Mitra, Pabitra, *Feature weighting in content based recommendation system using social network analysis*, Proceedings of the 17th international conference on World Wide Web, WWW '08, 2008, Beijing, China, Pages 1041 - 1042. ACM: New York, NY, USA,

[6] Decanter.com, *Wine Reviews*. Retrieved 1st May 2013 from http://www.decanter.com/wine/reviews/1

[7] Goldberg, D. Nichols, D., Oki, B. M., and Terry, D., *Using collaborative filtering to weave an information tapestry*, Commun. ACM 35, 12 (Dec. 1992), 61–70.

[8] Mangalindan, J. P., *Amazon's Recommendation Secret*, July 2012. Retrieved 1st May 2013 from http://tech.fortune.cnn.com/2012/07/30/amazon-5/

[9] Netflix: *About us*. Retrieved 1st May 2013 from https://signup.netflix.com/MediaCenter

[10] Netflix Prize Website: *Index*. Retrieved 1st May 2013 from http://www.netflixprize.com//index

[11] Netflix Prize Website: *Rules*. Retrieved 1st May 2013 from http://www.netflixprize.com//rules

[12] Patton, E., McGuinness, D., *Scaling the Wall: Experiences Adapting a Semantic Web Application to Utilize Social Networks on Mobile Devices*, 2010. In: Proceedings of the WebSci10: Extending the Frontiers of Society On-Line, April 26-27th, 2010, Raleigh, NC: US.

[13] Resnick, P., Iacovou, N., Sushak, M., Bergstrom, P., Riedl, J., *GroupLens: An open architecture for collaborative filtering of netnews*, 1994 ACM Conference on Computer Supported Collaborative Work, 1994. Association of Computing Machinery, Chapel Hill, NC.

[14] Resnick, P., Varian, H. R., *Recommender Systems*, 1997. Communications of the ACM, 40 (3), 56-58. Association of Computing Machinery, Chapel Hill, NC.

[15] Segaran, T., *Programming Collective Intelligence*, 2007. O'Reilly.

[16] Su, X., Khoshgoftaar, T. M., *A Survey of Collaborative Filtering Techniques*, Advances in Artificial Intelligence, vol. 2009, Article ID 421425, 19 pages, 2009.

[17] http://wineagent.tw.rpi.edu/index.php

[18] Ward, R., Towne, D., Stannard, D. H., LaChappelle, P., *Wine Recommendation System and Method*. International Application Number PCT/US2012/046480, filed 12/07/2012. Retrieved 1st May 2013 from http://patentscope.wipo.int/search/en/detail.jsf?docId=WO2013009990