Aggregating Information from the Crowd: ratings, recommendations and predictions

THÈSE N^{O} 6233 (2014)

PRÉSENTÉE LE 27 JUIN 2014

À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS

LABORATOIRE D'INTELLIGENCE ARTIFICIELLE

PROGRAMME DOCTORAL EN MANAGEMENT DE LA TECHNOLOGIE

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Florent Frédéric GARCIN

acceptée sur proposition du jury:

Prof. M. Finger, président du jury
Prof. B. Faltings, directeur de thèse
Prof. A. Bernstein, rapporteur
Prof. C. Tucci, rapporteur
Prof. L. Xia, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2014

Acknowledgements

I am very grateful to my thesis director, Prof. Boi Faltings, for his supervision, support, and insightful discussions. Thank you! My acknowledgements go to the members of my thesis jury for their valuable comments and suggestions: Profs. Abraham Bernstein, Matthias Finger, Christopher Tucci, and Lirong Xia. A special thanks goes to Lirong for the wonderful time across the Atlantic. I would like to express my gratitude to the lab assistants Marie Decrauzat and Sylvie Thomet for their help solving administrative mazes.

Thanks also go to former and current members of the AI lab, in particular to: Arnaud Jutzeler for discussions on how unique EPFL community is; Brammert Ottens for sharing the crazy organizations of awesome ski weekends; Christos Dimitrakakis for introducing me to context tree and for his precious expertise on statistics and probabilistic inference; Goran Radanovic for great discussions about incentive schemes; Immanuel Trummer for breaking stereotypes about Germans being so organized; Ludek Cigler for the awesome telemark theory and practice; Paul Dütting for tasty beers we have every time we meet; PuLi for sharing a great office and being the best office mate; Thomas Léauté for being (not?) so French, his unique sense of humour, and sharing the passion of cross-languages puns.

I would like to thank the many students I had the opportunity to work with: Abson Sae-Tang, Achraf Tangui, Anaël Fiaux, Baptiste Vinh Mau, Bernard Maccari, Céline Heldner, Clément Moutet, Egemen Vardar, Flavien Aubelle, Frederik Galle, GaoPeng, George Christodoulou, Grisha Rozhdestvenskiy, Hieu Pham, Jérémy Gotteland, Jonathan Link, Jonathan Rohrbach, Matthieu Fond, Michael Hobbs, Nadine Joswig, Phanindra Pisupati, Stéphane Bouquet, Sébastien Epiney, and last, but not least ZhouKai.

Finally, I cannot express how thankful I am to LiNa and my family for their unconditional love, help and support.

Lausanne, May 27, 2014 F. G.

Abstract

With an ever-growing amount of data generated on the web, aggregating information from the crowd into meaningful knowledge has become crucial to companies in order to create a competitive edge. Not only companies, but also organizations and governments can benefit from this aggregation.

In this thesis, we illustrate the benefits of aggregating information from the crowd with three applications. In the first application, we investigate different ways of aggregating users' ratings on review websites. With the help of gamification techniques, we introduce a new methodology for studying users' rating behaviour, and show that there is a balance between the amount of private information elicited from the crowd and the accuracy of the aggregated rating.

In the second application, we consider the aggregation of implicit feedbacks from the crowd in order to generate personalized recommendations of news articles. We propose a new class of recommender systems based on Context Trees (CT), specifically designed for a dynamic domain like the news. We show that CT recommender systems generate accurate and novel recommendations in an offline setting, but also in real time on the newspaper website *swissinfo.ch*.

In the last application, we address the problem of eliciting private information to predict outcomes of events. We report on an experimental platform called *swissnoise* that allows users to express their opinions on various topics ranging from sports to politics. It is the first platform to implement a peer prediction mechanism for online opinion polls. We show that peer prediction can be practically implemented, and discuss the design choices and variations made to such mechanism. Finally, we find that peer prediction achieves a performance comparable to that of prediction markets.

Keywords: aggregation, crowd, ratings, recommendation, news, opinion poll, prediction market, peer prediction

Résumé

Avec une quantité toujours croissante de données générées sur le web, l'agrégation de l'information provenant de la foule en connaissance structurée est devenue cruciale pour les entreprises afin de créer un avantage concurrentiel. Non seulement les entreprises, mais aussi les organisations et les gouvernements peuvent bénéficier de cette agrégation.

Dans cette thèse, nous illustrons les avantages de l'agrégation d'information de la foule avec trois applications. Dans la première application, nous étudions différentes façons de regrouper les notes des utilisateurs sur les sites web d'évaluations. Avec l'aide de techniques de ludification, nous introduisons une nouvelle méthodologie pour l'étude du comportement des utilisateurs lorsqu'ils donnent une évaluation, et montrons qu'il existe un équilibre entre la quantité d'informations privé extraite de la foule et la précision de l'évaluation agrégée.

Dans la deuxième application, nous considérons l'agrégation des évaluations implicites de la foule afin de générer des recommandations personnalisées d'articles de presse. Nous proposons une nouvelle classe de systèmes de recommandation basée sur les Arbres Contextuels (AC), spécialement conçus pour un domaine dynamique tel que les articles de presse. Nous montrons que les systèmes de recommandation AC génèrent des recommandations précises et nouvelles dans un cadre hors-ligne, mais aussi en temps réel sur le site Internet du journal *swissinfo.ch*.

Dans la dernière application, nous abordons le problème d'obtenir des informations privées pour prédire le résultat d'événements. Nous rendons compte des résultats obtenus sur une plate-forme expérimentale appelée *swissnoise* qui permet aux utilisateurs d'exprimer leurs opinions sur divers sujets allant du sport à la politique. Il s'agit de la première plate-forme qui met en oeuvre un mécanisme en ligne de prédiction par les pairs pour les sondages d'opinion. Nous montrons que la prédiction des pairs peut être pratiquement implémentée, et discutons des choix de conception et modifications apportés à ce mécanisme. Enfin, nous constatons que la prédiction par les pairs permet d'obtenir une performance comparable à celle des marchés de prédiction.

Mots clés : agrégation, foule, notations, recommandation, news, sondage d'opinion, marché de prédiction, prédiction par les pairs

Zusammenfassung

Angesichts der stetig wachsenden Datenflut im Internet, gewinnt das Extrahieren sinnvoller Informationen daraus eine immer grössere Bedeutung. Nicht nur für Unternehmen sind diese Informationen wertvoll, auch Organisationen und Regierungen können von den gewonnenen Informationen profitieren.
Diese Dissertation zeigt anhand von drei Anwendungen, wie sinnvolle Informationen aus der Datenflut extrahiert und genutzt werden können:
Die erste Anwendung untersucht unterschiedliche Möglichkeiten für zusammenfassende Darstellung von Rezensionen auf Bewertungsportalen. Mit Hilfe einer spielerischen Methode wird das Verhalten der Anwender studiert. Die Ergebnisse zeigen, dass zwischen der Menge der privaten Einzelinformationen und der Genauigkeit der Zusammenfassung ein Gleichgewicht herrscht.
Die zweite Anwendung befasst sich mit der Nutzung von impliziten Leserfeedback eines Nachrichtenportals zur Generierung von personalisierten Artikelempfehlungen. Es wird eine neue Klasse zur Generierung solcher Empfehlungen entwickelt, welche auf Context Trees (CT) beruht, die speziell für diese dynamische Umgebung entworfen wurden. Experimente mit den zur Verfügung gestellten Daten einer grossen Schweizer Zeitung sowie der Echtzeit-Test auf dem Nachrichtenportal *swissinfo.ch* zeigen, dass diese Systeme akkurate und abwechslungsreiche Empfehlungen generieren.
Die letzte Anwendung betrifft die Erhebung von privaten Informationen zur Vorhersage von Ergeignissen. Zur Untersuchung dieses Sachverhalts wurde die Plattform *swissnoise* entwickelt. Auf dieser können Anwender ihre Meinung zu unterschiedlichen Themen wie Sport und Politik formulieren. Die Plattform basiert auf einem Peer Prediction Mechanismus, also einer Technik die bisher nur theoretisch untersucht wurde. Hier wird nicht nur die Praxistauglichkeit dieser Technik bewiesen, sondern auch diverse Entscheidungen diskutiert, welche im Laufe der praktischen Umsetzung anfallen. Schliesslich wird das System mit den sonst üblichen Prediction Markets verglichen und festgestellt, dass vergleichbare Ergebnisse erzielt werden.

Schlagworte: Datenerhebung, Crowd, Bewertung, Empfehlung, Nachrichten, Prediction Market, Peer Prediction, Online-Abstimmung

摘要

随着网络数据量的日益增长，为了创造竞争优势，将群众信息集成为有意义的知识对公司、组织和政府部门都具有至关重要的意义。在本论文中，我们通过三个应用来阐述集成群众信息所带来的好处。在第一个应用中，我们研究评论网站上的用户评分的不同集成方法。通过游戏化技术，我们介绍了研究用户评分行为的新方法，并且揭示了从群众中所提取的隐私信息量和评分集成的精确度之间存在着权衡关系。

在第二个应用中，我们对隐性的反馈信息进行集成，以达到提供个性化新闻推荐的目的。我们提出一类新的基于上下文树（context tree）的推荐系统。这类推荐系统尤其适合于诸如新闻的动态领域。我们的研究结果显示，基于上下文树的推荐系统不仅能在离线环境设置下提供精确和新颖的推荐，并且在实时的新闻网站上也可以达到同样的效果（*swissinfo.ch*）。

在最后一个应用中，我们对通过获取隐私信息预测事件结果进行了阐述。我们设计了一个实验性平台*swissnoise*。用户可以通过该平台对体育、政治等各种不同的话题表达意见。该平台是第一个实现了通过同辈预测（peer prediction）机制进行在线意见调查的平台。我们揭示了同辈预测机制的可行性，并且对设计选择和对该机制的调适进行了讨论。最后我们发现，同辈预测机制可以达到与预测市场同等的性能。

关键词： 集成，群众，评分，推荐，新闻，意见调查，预测市场，同辈预测

Contents

List of Figures

List of Tables

1 Introduction

More than 2.5 billion gigabytes of data[1] are created every day in multiple forms. In one single minute on the internet, 72 hours of Youtube videos are uploaded, Google addresses 2 million search queries, 1.8 million items are liked on Facebook, 278 thousand tweets and 204 million emails are sent, Amazon makes $83'000 in sales and 17 thousand transactions take place at Walmart.

This tsunami of information is flooding straight into the world economy. Companies face a massive amount of data about their customers, and at the same time users will continuously generate new information. Indeed, in the era where everything is connected, the interactions between individuals and organizations generate a huge quantity of data that is most of the time, in fact, churned out by companies, unable to effectively process them.

It is thus crucial for companies to come up with efficient techniques to aggregate information into meaningful knowledge. Companies can leverage this knowledge in order to have a competitive edge. As a result they will be able to increase the quality of products and services for instance. Not only companies can benefit from aggregating information from the crowd, but also governments and organizations. What informations and how to aggregate them is not trivial and often context dependent. In this thesis, we illustrate these benefits with three applications of information aggregation from the crowd.

In the first application, we investigate various ways of aggregating users' ratings on review websites. Review websites play a major role in today's electronic commerce because they influence drastically customers' decision making. For instance on a review website where products are ranked by their overall aggregated rating, a product with a lower aggregated rating will have a lower chance to be seen and selected by potential buyers. Therefore, understanding how users give ratings is important. With the help of gamification techniques, we introduce a new methodology for studying users' rating behaviour. We explore different aggregating mechanisms with interesting properties, and we show that there is a balance between the amount of private information elicited from the crowd and the accuracy of the aggregated

[1] Estimations by IBM for 2012. http://www.ibm.com/big-data

rating. We illustrate the benefits of changing the way ratings are aggregated with two examples: ranking of products and product recommendations. We believe that our results are helpful in the design of human computation tasks in general, such as rating and reputation, community sensing, opinion polls or question-and-answer websites.

Review websites collect explicit feedback through ratings. In the second application, we consider the aggregation of implicit feedback from the crowd in order to generate personalized recommendations. We tackle the problem of recommending news articles to anonymous users solely based on their traces left on a newspaper website. Personalized news recommendations have specific challenges. First, users' preferences are subject to trends. Users do not want to see multiple articles with similar content, and often we do not have enough information to profile the users. Second, news evolves rapidly: stories and topics appear and disappear quickly, and old news are no longer interesting. Thus, recommendations should provide added value, and not just consist of the most popular stories that readers would have already seen on the front page. We address these issues by proposing a new class of recommender systems based on Context Trees (CT). CT recommenders employ an incremental algorithm that adapts the models continuously, and is thus better suited for such a dynamic domain as the context tree evolves over time and adapts itself to current trends and reader preferences. We conduct an *offline* evaluation to demonstrate that CT recommenders make accurate and novel recommendations. We then discuss how to apply CT recommenders to a live-traffic website, and explain the design and implementation of a new *online* evaluation framework. Finally, we deploy our framework on the newspaper website *swissinfo.ch* in order to run a live evaluation.

Business decisions are tightly linked to the information gathered behind them. Unfortunately, decision makers hardly ask employees who have direct contact with the customers how, for instance, a new product will fly. Hence, executives take decisions without this crucial information that could improve their analysis and reduce bad decision making. In the third application, we address the problem of eliciting private information to predict outcomes of events. It is usually done by rewarding participants for their responses. However, participants might not always give honest answers. There are two mechanisms that provide this desired property: prediction markets and peer prediction. Prediction markets have been extensively used in practice, but on the contrary peer prediction has not been much explored. We report on an experimental platform called *swissnoise* that investigates these two mechanisms. It is the first platform to implement a peer prediction scheme in public opinion polls. Therefore, we first show that peer prediction can be practically implemented. Second, we discuss the design choices and variations made to such mechanisms when implementing them. Finally, we show that peer prediction achieves a performance comparable to that of prediction markets. Our results are not only useful to companies, but also to organizations and governments. They can get insight on crucial issues in order to improve their analysis and reduce bad decision making. In particular, peer prediction has the advantage of not requiring observable outcomes, and is thus more suited for hypothetical questions related to product development, developing and selecting marketing campaigns, or policy making.

2 Aggregating Ratings

2.1 Introduction

Review websites play a major role in customers' decision making, superseding traditional word-of-mouth. The importance of online reviews is such that companies use them as proof of quality and do not hesitate to advertise their overall score. For instance, Tripadvisor[1] is issuing a "certificate of excellence" (Figure 2.1) based on the overall rating score achieved by a hotel. These certificates are then displayed on hotel websites and entrance doors.

In general, review websites order products according to their ratings, and user only consider those that are at the top of such rankings. Such ordering is obtained by aggregating individual ratings into a single value. By submitting a rating, users influence the overall score of the product they rate, and thus the final decision of potential customers.

Ratings are most commonly aggregated using the arithmetic mean. However, the mean is susceptible to outliers and biases [35], and thus may not be the most appropriate aggregator because reviews are often biased [45, 54]. As such, any website that collects users feedback, reviews or ratings has an interest in designing a robust and strategyproof mechanism to aggregate users preferences and ratings.

In this chapter, we study the influence of different aggregators on rating behavior and the accuracy of the final aggregate. It would be best if we could carry out a study using data from actual rating websites. However, nobody knows what the true quality of the rated items is, nor do we have access to the beliefs of the raters. Furthermore, current rating websites are polluted by spammers that have other motivations than obtaining accurate rankings. Our interest is to model the behaviour of raters whose intention is to make the website reflect the true quality, either because they are altruistic or because the website provides incentives based on its success.

[1] http://www.tripadvisor.com

Figure 2.1: Tripadvisor's certificate of excellence issued to an hotel reaching 4.5 stars out of 5.

2.1.1 Contributions

In this chapter, we study the influence of different aggregators on rating behavior and the accuracy of the final aggregate. To do so, we introduce a new way of studying users' behaviour. We place people in a game situation that closely mirrors the rating scenario. In the game, we can carefully control the beliefs and motivations of users, and observe their reaction.

Second, we study the properties of different aggregators (mean, median and truncated mean) on the rating behavior of users, and in particular on the degree of truthfulness and how well the aggregate rating matches the true quality of a product.

Finally, we illustrate the benefits of changing the way ratings are aggregated in two applications: ranking and recommendation. This chapter builds on the results published in [54, 35, 36, 37].

2.1.2 Related Work

In the literature on online ratings, some works focus on the underlying distribution of ratings and the potential biases [45, 46, 54, 29]. However, only a few researches explore how rating aggregation should be made [35, 71, 62].

Ratings are most commonly aggregated using the arithmetic mean. However, the mean is quite sensitive to outliers and biases and thus may not be the most informative aggregator. Garcin et al. [35, 36] show that other aggregating functions perform better with respect to different criteria such as the informativeness, robustness and truthfulness. On all these criteria, the mean seems to be the worst way of aggregating ratings. The median is more robust, truthful and improves recommendation accuracy.

McGlohon et al. [71] study how to aggregate reviews of different scales and from different sources. They look at statistical and re-weighting methods for aggregating ratings. For the

former, they use techniques such as the mean, the median and lower bounds on normal and bimodal confidence interval. For the latter, the idea is to give more weight to useful ratings. Because it is impossible to know the ground truth about the true quality of a product, they evaluate the accuracy of the proposed methods based on pairwise ranking of items and sampling from the existing users' ratings. The accuracy is then computed on the number of correctly ranked pairs of items. They conclude that the proposed methods do not outperform the mean, and the median performs poorly because of multiple ties.

Leberknight et al. [62] introduce a rating aggregation technique based on the rating volatility. It has the advantage to capture the temporal trend of a product or service, and to be more responsive than the mean.

In these works, the ground truth about the true quality of a product is never known, and thus it makes it difficult to know how an aggregator will behave when implemented on a real website.

The topic of aggregation in crowdsourcing scenarios is also relevant. The observation that a crowd of laymen could together obtain a better judgement than an expert was first made by Galton [30] and made popular by the book of [99]. Most work has focused on applications of this effect to specific problems.

It is known that workers are prone to dishonest behavior in crowdsourcing markets [98]. As a result, a flourishing literature on quality control for crowdsourcing tasks emerged [49, 61, 50, 76]. While some works target specific tasks such as classification [106, 49, 55], few address tasks with more complex outputs like ranking [58]. To our knowledge, none looked at the problem of rating aggregation.

Closely related to rating aggregation is the research on vote aggregation in social choice theory [17, 3, 60]. Clemen and Winkler discuss the combination of experts' probability distributions in risk analysis. They conclude that simple rules such as the mean are important because they are easy to use, have robust performance, and are easy to justify in public policy settings. Ariely et al. suggests to take averages for quantitative judgement because it is a powerful and robust way of reducing the judgement error. Larrick and Soll show that people in general have misconceptions about the average and it should be used to reduce judgement error.

Another related line of research is the Maximum Likelihood Estimation (MLE) point of view of social choice, where the agent's preferences are ordinal, that is, they report *orders* instead of ratings [19, 21, 20, 105, 81]. More recently, various voting rules and elicitation schemes have been evaluated in human computation scenarios [77, 70].

Our problem is also related to the pointwise approaches in *learning to rank* [67], which focuses on designing sensible error functions and fast algorithms for computing the ground truth. However, in MLE of social choice and learning to rank, the agents are assumed to be truthful, while in our study, agents can be strategic.

2.2 Aggregators

Reports are most commonly aggregated using the arithmetic mean. However, the mean is quite sensitive to outliers and biases and thus may not be the most appropriate aggregator. There exist aggregation functions such as the median with interesting properties [35, 36]. We consider the following aggregators:

- the mean is defined as the average value $\text{Mean}(R) = \frac{1}{|R|}\sum_{r\in R} r$
- the median is the value in the middle between the lower and upper half of the reports when the reports are ordered by their magnitude. When the number of reports is even, we break the ties by taking the average of them. Formally, consider $R = (r_{(1)}, r_{(2)}, ..., r_{(n)})$ the reports sorted in ascending order, and let $n = 2m-1$ if n is odd, and $n = 2m$ if n is even for some integer m. The median is the value such as:

$$\text{Median}(R) = \begin{cases} r_{(m)} & \text{if } n \text{ is odd,} \\ \frac{r_{(m)}+r_{(m+1)}}{2} & \text{if } n \text{ is even.} \end{cases} \tag{2.1}$$

- the α-truncated mean (with $\alpha \in [0,50)$) drops the highest and lowest $k = \lfloor \alpha(n-1)/100 \rfloor$ reports, and compute the mean of the remaining reports.

$$\text{Mean}_\alpha(R) = \frac{1}{n-2k}\sum_{i=k+1}^{n-k} r_{(i)} \tag{2.2}$$

During the experiments, we will vary $\alpha \in \{10, 20, 30, 40\}$. Note that the mean and median are special cases of the truncated mean, respectively at $\alpha = 0$ and the median is the limit as $\alpha \to 50$ (at 50% no ratings would remain to take the average).

2.3 Rating Model

The goal of a product rating website is to collect ratings and aggregated them in order to reveal the true quality of a product. The process of collecting ratings is illustrated in Figure 2.2. Here, a mobile phone has a true quality q^*. This true quality is unknown and hidden. A user u perceives the quality of the mobile phone by using it, and we write q_u this perceived quality. Note that the user's perceived quality q_u might not be the same as the true quality of the product q^*. The user u sends a report r_u to the product rating website. Finally, the product rating website aggregates ratings from users.

Following the scenario in Figure 2.2, we consider the problem of estimating a continuous signal drawn from a certain probability distribution. Reporters obtain noisy observations of samples of the signal according to this probability distribution and form an opinion on the most likely value of the signal.

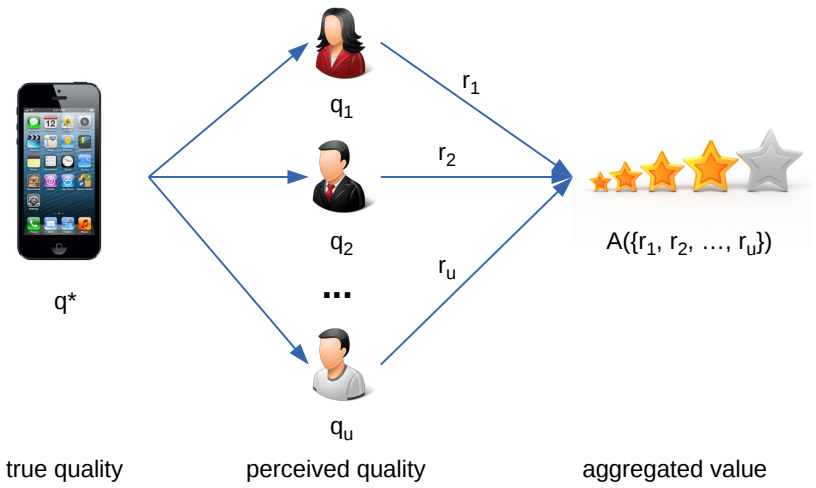


Figure 2.2: Typical rating scenario.

More formally, let q^* denote the true value of the signal, and $f(\cdot|q^*)$ a probability distribution of this signal. We denote $q_u \sim f(\cdot|q^*)$ the user's u observed value of the signal, and r_u the report of user u. The user u can actively get samples s such that $s \sim g(\cdot|q_u)$, where g is another (possibly the same as f) probability distribution. The range of g should be close to q_u in order to avoid too much noise, hence we bound g such that $s \in [q_u - \epsilon, q_u + \epsilon]$, with small ϵ. For example, in a product rating scenario the probability distribution g models the uncertainty of a user on *perceiving* the personalized quality of a product, whereas f models the noise in generating personalized qualities of an item.

We assume that errors in observation of the signal are independent of one another and follow a normal (Gaussian) distribution. It has been shown that in scenarios such as product reviews the reports follow other distributions: Hu et al. [45, 46] observe that Amazon product reports follow a U-shape distribution or a J-shape distribution, a normal distribution centered to the extreme report. However, these distributions are caused by purchasing and self-selection bias. Users with extreme opinions are more likely to "brag or moan", justifying a U-shape distribution. Users tend to purchase products with higher valuation, making a J-shape distribution. When users are systematically queried about their opinion, their distribution is close to a normal distribution [45].

However, when users are systematically queried about their opinion, their distribution is close to a normal distribution [45]. In this chapter, we focus on scenarios where users participate with the goal of giving the most accurate information without any external motives. Such scenarios could apply not only to product rating websites, but also community sensing, opinion polls or question-and-answer websites. Note that our methodology can be extended to other cases.

More precisely, we consider that q^* is uniformly distributed in the interval $[a,b] = [0,10]$ and the signal instances as well as their observations are perturbed by normally distributed noise:

$$q^* \sim \mathcal{U}(0,10) \tag{2.3}$$
$$q_u \sim f = \mathcal{N}(q^*, \sigma^2) \tag{2.4}$$
$$s \sim g = \mathcal{N}(q_u, 0.4) \tag{2.5}$$

In the above formulas $\mathcal{U}$ is the uniform distribution and $\mathcal{N}$ is the normal distribution. We choose σ^2 such that it varies with respect to the true quality of the item q^*. If q^* is located near the boundaries of the domain $[a,b] = [0,10]$ then σ^2 is smaller, making the distribution f narrower. Products with an average true quality tend to have a larger variance than products which are on the extremes [97].

2.4 Gamification

We are interested in observing how people behave when their task is to obtain a true estimate of the signal, as it would be the case in product rating or estimation of publicly observable phenomena. To this end, we construct an interactive game. This technique of *gamification* [25] allows us to conduct a user study and collect data from users while they are playing a game. This approach increases user engagement and data quality [16].

To avoid any influence from previous beliefs, the game places the players in a completely unfamiliar situation that closely mirrors the characteristics of a product rating scenario such that the reporter is motivated to make the result reflect reality. In this game (called the *fishing game*), the player is assumed to be hired by a fishing company, and her job is to estimate the concentration of fish in different regions of the lake. For a given region, the player u has to evaluate and report the number of fish r_u in this region. The player sees reports $R \setminus \{r_u\}$ submitted by other players. She can observe as much as she wants the different regions of the lake to obtain samples s of the number of fish. This rating scenario is illustrated in Figure 2.3.

The utility U depends on how closely the aggregate value corresponds to the true value as observed by the company. The player starts with a given budget and the goal of the game is to gather as much points as possible. As the game is simulated, we can place the player in situations where her own measurements are very similar or very different to the reports of other players, and observe her behaviour.

We model users who are concerned about making the most accurate aggregation. In this case, users want to see the aggregate as close as their own perceived signal. Thus, we model their utility function by the Euclidean distance as

$$U(R|q_u, \mathcal{A}) = 1 - \frac{|q_u - \mathcal{A}(R)|}{b-a} \tag{2.6}$$

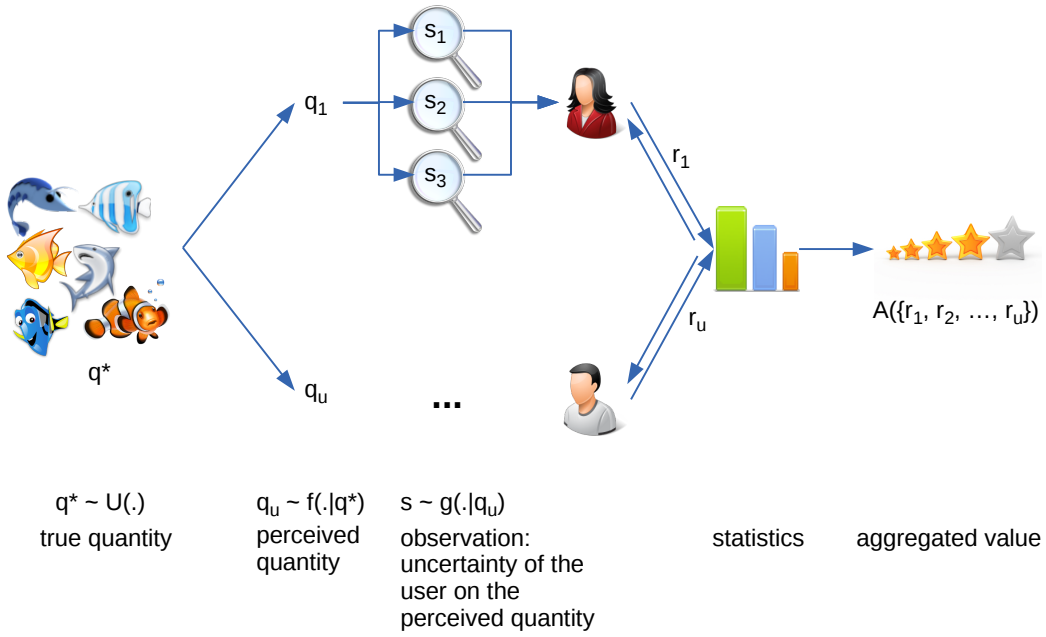


Figure 2.3: Gamified rating scenario.

where R is the set of all reports including the player's report, $\mathscr{A}$ the aggregator and q_u the user's perceived value of the signal, or in this case the user's perceived number of fish. For simplicity, we bound the probability distributions f and g between a and b, i.e. $q^*, q_u \in [a,b]$. Note that the utility is maximum when the user's perceived number of fish equals the aggregated value $q_u = \mathscr{A}(R)$.

In order to collect accurate data, we test different combinations of the initial parameters: the true amount of fish q^* and the aggregator $\mathscr{A}$. This defines a *game* as a tuple $G = (q^*, \mathscr{A})$. The rest of the parameters (i.e. q_u) are directly derived from the game, except for f, g, ϵ and the boundaries a, b which are fixed. For a given game, we collect 3 reports (one per region of the lake). Each region of the lake has different parameters generated from G, allowing us to collect more data and increase randomization.

We believe that the information about the aggregator available to the users plays a crucial role in the behaviour of the users. To ensure that users understand how reports are aggregated, we make them pass a short exam before entering the actual game.

The process of the game is the following: the player

1. sees a short description of the game with its goal,

2. sees a description of the aggregator with an example,

Figure 2.4: Screenshot of the game: reports of other players (top left), sampling system (bottom left) and area to estimate (right).

3. takes an exam to validate the understanding of the aggregator,
4. if she passes, plays one game G.

Each of these steps is displayed on a new panel (screen). If a player fails at step 3, she can take another exam. However, we track the success rate of each player to filter out those players who do not understand the aggregator, or who tend to play randomly. When playing one game (step 4), it is possible to access step 1 and 2 at any time.

Figure 2.4 is a screenshot of the game interface. On the right side, it shows which area of the lake is currently selected. On the left side, we display the distribution of reports from other players, and below the samples obtained from the observations. In this particular case, the true number of fish was set to $q^* = 6$, while for the user it was $q_u = 4$.

2.4.1 Metrics

We measure two behaviours. The first one concerns the user and her reports. We want to see if users tend to report their true observation or try to drive the aggregate value towards their true preference. Therefore, we will measure the percentage of *truthful* reports.

Definition 1 (Truthfulness)**.** *A report r_u from user u is* truthful *if*

$$|q_u - r_u| \leq \epsilon \tag{2.7}$$

Truthfulness is defined within a margin ϵ which goes for the noise involved in the sampling made by the user.

Second, we look at the capability for an aggregator to reveal the true value of the signal. Consider a probability density function f symmetric on some unknown point θ that we want to estimate. In theory, mean, median and truncated mean are consistent estimators of θ because they converge to this value as the sample size gets bigger. However, in practice it might not be the case. Thus we need to look at another way to measure their performance. We define the *Mean Absolute Error* (MAE) as a measure of how spread out the estimator (the aggregated value) of a reports distribution is from the true value of a product.

Definition 2 (Mean absolute error)**.** *The* mean absolute error *e of an aggregator* $\mathscr{A}(R)$ *for a set of reports* R, $n = |R|$ *is given by*

$$e(R|\mathscr{A}, q^*) = \frac{1}{n}\sum_{i=1}^{n} |q^* - \mathscr{A}(\{r_1, ..., r_i\})| \tag{2.8}$$

In addition, we measure the robustness of an aggregator to manipulation. We define a successful manipulation as a case where manipulators set the aggregate to a value of their choosing through malicious reports.

Definition 3. *The* robustness *of an aggregator is a total order* $\geq_r$ *over aggregators. For any pair of aggregators* $\mathscr{A}_1, \mathscr{A}_2$*, we say that* $\mathscr{A}_1 \geq_r \mathscr{A}_2$ *iff in all scenarios where there is a manipulation through malicious reports that succeeds with aggregator* $\mathscr{A}_1$*, there is also a manipulation that succeeds with aggregator* $\mathscr{A}_2$*. We say that* $\mathscr{A}_1 >_r \mathscr{A}_2$ *iff* $\mathscr{A}_1 \geq_r \mathscr{A}_2$ *but not* $\mathscr{A}_2 \geq_r \mathscr{A}_1$*.*

Thus, whenever $\alpha_1 > \alpha_2$, the α_1-truncated mean is more robust than the α_2-truncated mean, since it discards more reports and thus more reports of manipulators. This also shows that the mean is the least robust aggregator, while the median is the most robust aggregator [47].

2.5 Results

The first hypothesis is based on the observation that most people trust their own observations more than those of others, so that users would tend to want the aggregate to be as close as possible to their own beliefs. Thus, they would strategize less when the aggregator is more robust, leading to the following hypothesis:

Hypothesis 1. *The number of true reports (truthfulness) increases as the degree of robustness increases.*

Besides truthfulness of the reports, another interesting question is how well the aggregate reflects the true value. The lowest mean absolute error would be reached with truthful reports and the mean as an aggregator. However, the mean does not elicit truthful reports and the median does not perform the best aggregation. Thus, we hypothesize:

Hypothesis 2. *There is a tradeoff between truthfulness and mean absolute error: no aggregator achieves the highest accuracy as well as truthfulness.*

We conduct two types of experiments. In the first type, we look at a static setting in which a single instance of the signal is presented (with noise) to the users. In the game, we generate 10 artificial reports from the same distribution as q_u, and ask the users to play the game as described previously.

In the second type, we look at the dynamic process of estimating the signal from different observations. We conjecture that the behavior changes with the number of reports. When there are few reports, a fake report would influence the overall rating more than when the total number of reports is high. In addition, a malicious user might have greater incentives if she sees that it is easier to alter the aggregated value. So we will consider a dynamic setting where users provide reports in sequence.

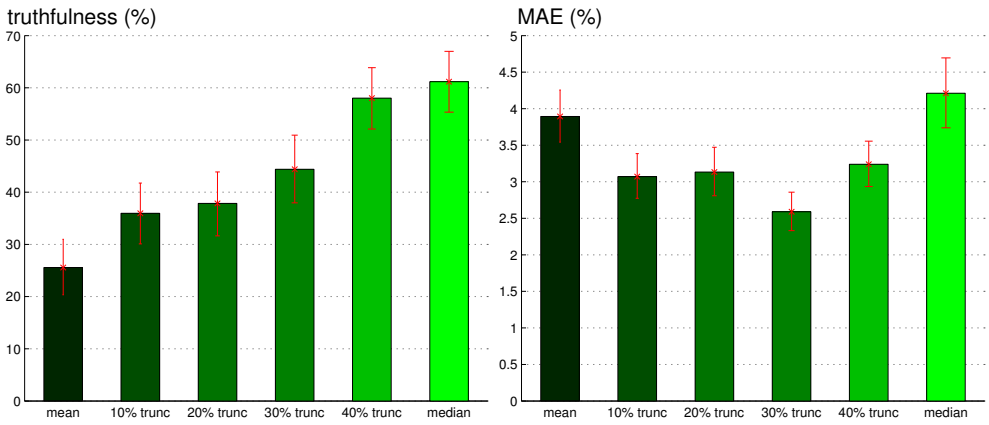
2.5.1 Static Experiment

The static experiment is the following. For a given user, we select one aggregator and we generate the true number of fish $q^* \sim \mathcal{U}(0,10)$. We draw 10 artificial reports from the distribution $\mathcal{N}(q^*,\sigma^2)$, plus one value which will be the user's true perceived value of the number of fish $q_u \sim \mathcal{N}(q^*,\sigma^2)$. q_u is the value we want to "inject" as a belief, and the user will be able to sample from. We do not take into account reports from users who do not sample.

We evaluate 6 aggregators: the mean, truncated mean at 10%, 20%, 30%, 40% and the median. Each user plays 3 scenarios per aggregator, for a total of 18 scenarios. Since there are 3 areas of the lake per scenario, it means she reports in total 54 ratings. 21 users from a technical university played the game. They are all scholars (Master and PhD students, PostDocs and Professors) with a background in computer science. They played altogether for 7 hours and 16 minutes, which makes it 21 min per user, for an average of 1'10" per scenarios. We present averages with confidence intervals at 90% (bootstrap sampling, 10'000 samples).

Figure 2.5(a) validates Hypothesis 1, and shows that indeed the truthfulness increases with the robustness. We fixed the threshold ϵ to 1. With the median which is the most robust aggregator, more than 60% of the reports are truthful, while with the mean we obtain only about 25% of truthful reports.

We might expect the truthfulness to be maximum at 100% with the median [75], but in practice it is not the case. This may actually be due to the fact that some participants do not fall prey to the fallacy of trusting their own observation, but also trust the reports of others. For normally distributed reports, the mean is the most accurate aggregator as it minimizes the mean square error. Users who trust the accuracy of other users' reports should be truthful with the mean, and non-truthful with the median because the median would not pick the best aggregated value.



(a) Truthfulness ($\epsilon = 1$)

(b) Mean absolute error

Figure 2.5: Truthfulness and mean absolute error for the static experiment.

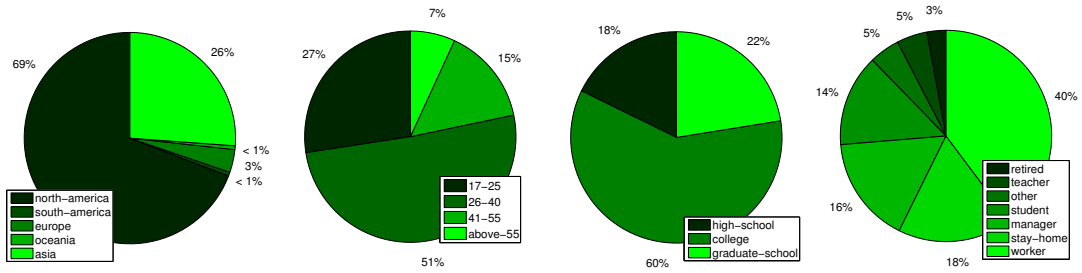
Intuitively, the mean has the lowest MAE, and the median the highest. This holds under the assumption that the reports are truthful and come from the same underlying distribution. In practice, Figure 2.5(b) demonstrates that it is not the case, leading to Hypothesis 2. The mean and median have about the same MAE (~4%). However, the 30%-truncated mean dominates with a MAE near 2.5%. It is important to note that these numbers are not surprisingly low because the other reports are artificial and come from the same underlying distribution. We normalized the MAE by the range of possible values.

2.5.2 Dynamic Experiment

In the static experiment, the reports attributed to other players were randomly generated so that our measurements of MAE were not representative of a realistic scenario where players influence one another. Thus, we also ran a dynamic experiment using Amazon Mechanical Turk where a player saw actual reports from other players. The players received a fixed payment to play the game, and a bonus based on their performance.

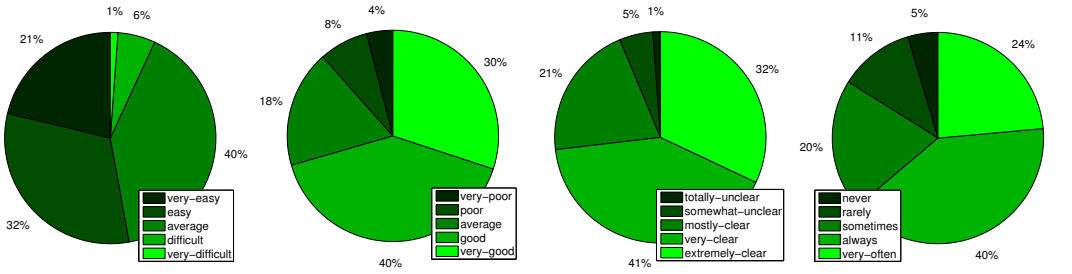
538 players played the game and gave 7902 reports (before filtering). We filtered players in two ways. First, they had to pass an exam on the understanding of the aggregators. The success rate was 96% with on average 1.5 attempts. Second, we discard reports of players who do not sample. On average, players made 3.9 observations per area. After this filtering, we had a total of 4921 reports from different players and we selected the first 120 reports for each aggregator, scenario and area. We present averages with confidence intervals at 90% (bootstrap sampling, 10'000 samples).

Figure 2.6 shows the demographics of the players: their origin, age, education and current occupation. About half of the players are male (53.5%). These statistics about Amazon Mechanical Turk are not surprising and follow the trends observed by Ross et al. [89].



(a) Origin (b) Age (c) Education (d) Occupation

Figure 2.6: Players' demographics



(a) Difficulty level (b) Game quality (c) Instruction clarity (d) Demand for similar tasks

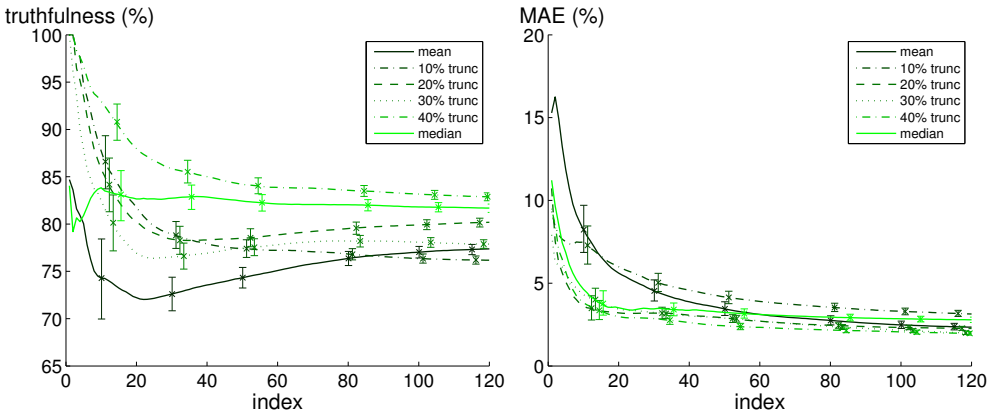
Figure 2.7: Game features

Regarding the game itself (Figure 2.7), players are satisfied. More than 80% of the players enjoyed playing this game. It was designed in such a way that it is not difficult to understand and play: more than 73% find the instruction clear, and about 53% find the game easy. The engagement of the players is such that they would like to see more tasks involving games in Amazon Mechanical Turk (Figure 2.7(d)). It is important to note that players thought they were playing a game, and they did not realized they were involved in a user behavior study.

Figure 2.8 shows the average degree of truthfulness, using a margin $\epsilon = 1$, and the MAE for different aggregators as the number of reports increases. At each time point, the player sees the reports of previous players and contributes its own report. The curves show the evolution of both as reports are gathered incrementally, with players always shown the earlier reports. It thus mirrors closely the behavior of an online rating website. We normalized the MAE by the maximum possible deviation (error) to the true value.

The biggest differences in truthfulness and MAE exist when the number of reports is small. This can be expected as here the potential for manipulation is the greatest. For example in a product rating scenario, it is also very crucial since the rating will determine the popularity of an item and thus the potential for even obtaining future ratings.

For both truthfulness and MAE, the mean is clearly not the best aggregator. Truthfulness drops



(a) Truthfulness ($\epsilon = 1$)

(b) Mean absolute error

Figure 2.8: Truthfulness and mean absolute error for the dynamic experiment.

dramatically when the number of reports is small. Players want the aggregate to be as close as possible to their own belief. However when the number of reports increases, some players tend to trust the reports of others (Figure 2.8(a)).

After 120 reports, the difference in MAE among aggregators is small (Figure 2.8(b)). The mean and 10%-truncated mean converge slower than more robust aggregators. More robust aggregators need less reports to reach an acceptable MAE: about 60 reports are required for the mean while less than 20 reports for the median.

Regarding Hypothesis 1, there is a difference between the most and least robust aggregators, i.e. the median and mean respectively. However it is difficult to conclude anything for the truncated mean methods.

Hypothesis 2 is also valid in the dynamic setting. However this tradeoff exists mostly when the number of reports is small because the difference among aggregators is more important for both truthfulness and MAE. Note however that in a product ranking, even small differences in ratings can have a huge influence on the position in the ranking, so that the differences in MAE can still have a big impact.

Overall, the median or 40%-truncated mean are good candidates for aggregators. They both achieve high truthfulness and low MAE over small and large numbers of reports.

2.6 Discussion

The analogy we are making between the fishing game and a product rating scenario is not perfect because the fishing game removes some biases that exist in a product rating scenario. However, although it would be possible to conduct such experiment on a product rating website with real products, we would not know what the true quality experienced by a user is.

Since it is a new and unique game, there are no prior expectations on the quantity of fish (quality) nor word-of-mouth effects [10]. Social influences on users' perception of quality can lead to biased ratings [18, 12]. Influences from other players (through displaying previous reports) and their need of conformity might drives players' behaviour. We have not explored this issue, but our methodology would be able to address it by, for instance, hiding the reports of other players.

This study illustrates a cognitive bias referred to the *Lake Wobegon* effect, in which human tends to overestimate one's capabilities. This illusory superiority appears for the mean aggregator: users who trust their own observation should report truthfully because the mean is the most accurate aggregator. Actually, in this study it is not the case, showing this effect [39].

We believe that our methodology is helpful to understand how rating aggregation happens, and to illustrate that another choice of aggregator could improve rating websites. In situations such as community sensing, opinion polls or question-and-answer websites, our analogy fits better.

In general, review websites order products according to their ratings, and user only consider those that are at the top of such rankings. Such ordering is obtained by aggregating individual ratings into a single value. By submitting a rating, users influence the overall score of the product they rate, and thus the final decision of potential customers. Most rating websites use the arithmetic mean as a way of aggregating ratings. We have seen that when the number of ratings is small, there are big differences between aggregators in terms of truthfulness and performance. We believe that rating websites should consider an adaptive strategy where aggregators are dynamically selected as the number of ratings increases.

We believe that our methodology is helpful to understand how rating aggregation happens, and to illustrate that another choice of aggregator could improve rating websites. Our analogy applies also in situations such as community sensing, opinion polls or question-and-answer websites. We illustrate our findings with two applications in the next section.

2.7 Applications

Recommender systems use ratings provided by users to recommend products. Recommendations can be uniform for the entire user population, such as hotel recommendations on Tripadvisor.com, or personalized to the taste of a specific user, such as product recommendations on Amazon.com. Common to both types of systems is that they collect ratings of items from their users, aggregate these ratings and allow users to filter the items that are ranked highest according to these aggregates.

In the first application, we investigate the effect of the aggregator on the ranking of uniform recommendations. In the second application, we study the robustness of aggregators for personalized recommendations.

Table 2.1: Tripadvisor dataset

City	Reviews	Hotels
Boston	5537	66
Las Vegas	28553	131
New York	40676	264
Sydney	3659	103

2.7.1 Ranking of Uniform Recommendations

We consider feedback from a popular travel website named Tripadvisor that collects reviews of hotels from users around the world. The reviews contain a textual comment with a title, an overall rating and numerical ratings from 1 (lowest) to 5 (highest) for different features such as cleanliness, service, location, etc. The site provides ranking of hotels according to their location. Like most of the reputation sites, it aggregates reviews into a single value for each hotel and, based on that value, sorts hotels in ascending order. It uses a simple arithmetic mean on the overall ratings to recommend hotels.

We selected four cities for this study: Boston, Las Vegas, New York and Sydney. Table 2.1 shows for each city the number of reviews and hotels. All data were collected by crawling the website in July 2007.

Robustness

It is well-known that distributions of reports are far from normal due to reporting biases [45]. Aggregators such as the mean, median and mode have relatively the same value for normal distributions. However, they should have a significant difference for non-normal distributions. To support this hypothesis, we conducted the following experiment. For each of the four cities considered in our study, we computed a full ranking of the hotels according to each of the aggregators explained in Section 2.2. Then, for every pair of aggregators we measured the *distance* between the corresponding orderings of hotels within a city. To measure the distance between the two rankings we chose the average absolute difference between the position of the same hotel in the two rankings.

The results are presented in Table 2.2. For example, the rank of a hotel varies on average with 14.7 positions (up or down) when the ranking is done according to the median instead of the mean. The average difference of ranks triggered by different aggregators is quite high: 1 to 15 ranks. Considering that most review websites display only the first 5 or 10 "best" items, the results of Table 2.2 show that different aggregators can completely change the list of candidates suggested to the users. Therefore it becomes important to better understand the properties of each aggregator.

We look at the robustness of each aggregator by taking the number of outliers required to alter the ranking of a given hotel. For each hotel, we inject outliers with the highest possible ratings,

Table 2.2: Average absolute difference of ranking

	mean	**10%**	**20%**	**30%**	**40%**	**median**
mean	-	1.391	3.422	5.134	7.989	14.736
10%	-	-	2.576	4.422	7.466	14.427
20%	-	-	-	2.729	6.386	14.183
30%	-	-	-	-	5.266	14.009
40%	-	-	-	-	-	10.119
median	-	-	-	-	-	-

Table 2.3: Average number of outliers (with highest ratings '5') required to alter the ranking (all p-values < 0.05). In bold, the highest values.

	Boston	**Las Vegas**	**New York**	**Sydney**
mean	3.210	3.927	2.096	2.134
10%	3.065	4.037	2.323	2.030
20%	2.871	3.817	2.458	2.507
30%	3.129	8.881	8.076	3.552
40%	8.016	23.266	30.462	6.791
median	**20.548**	**87.954**	**59.000**	**11.582**

i.e. 5, until the rank changes. Table 2.3 summarizes the results for each city.

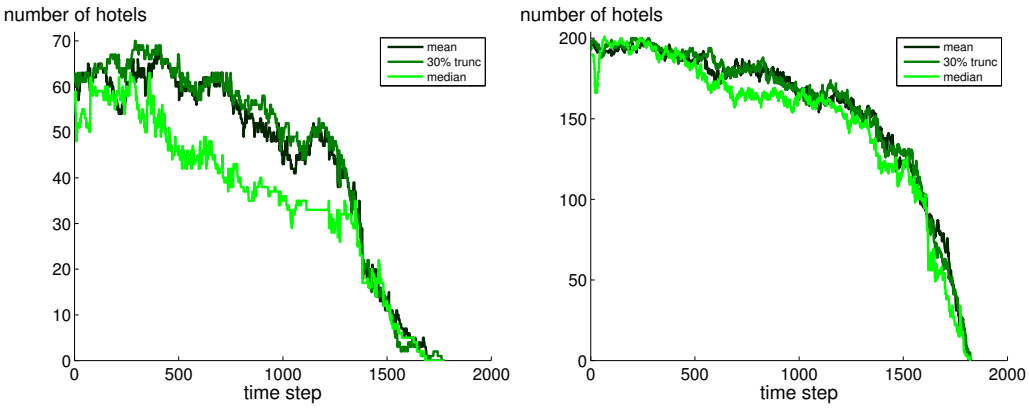
Two reviews are enough to change the rank when the aggregator is the mean while the median needs at least 11 outliers. This clearly shows that when we increase the percentage of truncated ratings, hence the robustness of the aggregator, we require more and more outliers to manipulate the aggregation.

Informativeness

In review websites, the goal of the aggregator is to reflect the user's reviews into one value. One assumption of aggregator is that users have reported their true experience. However, it is often not the case. For instance, the ratings are often part of discussion threads where past reviews influence future reports by creating prior expectations [54]. Therefore we can ask how an aggregator will continue to correctly reflect users' opinion.

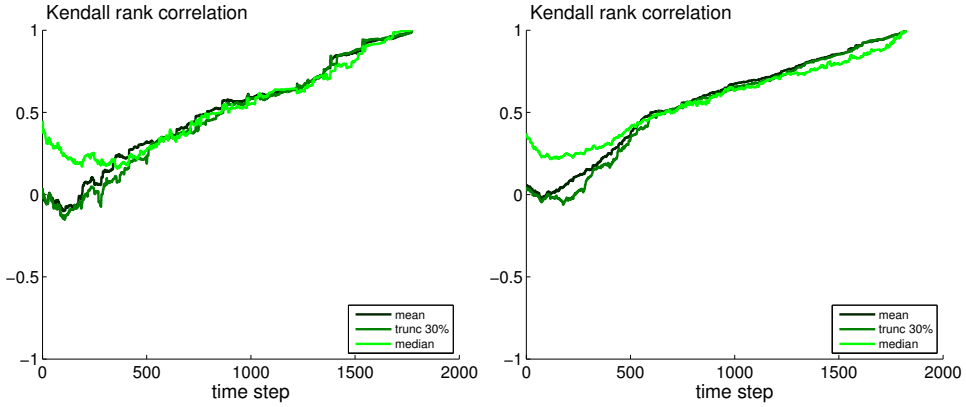
Figure 2.9 illustrates the stability of each aggregator by counting the number of hotels that deviate by more than 5 ranks from the final ranking. We selected hotels with at least 50 reviews and at each time step, one rating is given to one hotel. The median is the most stable aggregator. However, the 30%-truncated mean follows the standard mean.

Figure 2.10 shows the Kendall rank correlation over time. Although all aggregators have the same convergence speed, the median has a higher correlation to the final ranking when the number of ratings is small (less than 500 ratings).



(a) Las Vegas (b) New York

Figure 2.9: Number of hotels over time that deviate more than 5 ranks from the final rank.



(a) Las Vegas (b) New York

Figure 2.10: Kendall rank correlation over time.

2.7.2 Personalized Recommendations

Any collection of ratings is likely to contain outliers or even ratings that have been inserted with the purpose of manipulating the recommendations, therefore it is desirable that the aggregation function should be as robust as possible against them. We illustrate here how the aggregation method influences the accuracy and robustness of the recommendations. We measure robustness by the fraction of users whose recommendations are likely to be affected by outliers or malicious ratings (hit ratio).

We present results of empirical studies. We report on experiments with the MovieLens data that show that the median may also be helpful to defend against outliers and malicious attacks. We observe that the notions of average differ significantly. In particular, the median tend to be more robust to outliers and biased reviews but also result in much higher recommendation accuracy than the mean, and thus may be more informative for a user.

Settings

We constructed a user-based collaborative filtering system based on the k-Nearest Neighbour algorithm as outlined in [73]. Given a user u and a target item i for which the system must offer a recommendation, the algorithm first computes the 50 most similar users to u (neighbours of u) based on the available ratings. The similarity between users u and v is computed using Pearson's correlation coefficient:

$$sim_{u,v} = \frac{\sum_{i=1}^{n}(r_{u,i} - \overline{r}_u) \cdot (r_{v,i} - \overline{r}_v)}{\sqrt{\sum_{i=1}^{n}(r_{u,i} - \overline{r}_u)^2} \cdot \sqrt{\sum_{i=1}^{n}(r_{v,i} - \overline{r}_v)^2}} \tag{2.9}$$

where $r_{u,i}$ and $r_{v,i}$ are the ratings of some item i for u and v respectively, and $\overline{r}_u$ and $\overline{r}_v$ are the average ratings of u and v over the set of items.

The predicted rating for an item i not yet rated by user u is computed by aggregating the ratings of the u's neighbours for item i. We consider two aggregation rules that take advantage of the similarity:

- the mean of the neighbouring ratings weighted by their similarity:

$$pred_{u,i} = \overline{r}_u + \frac{\sum_{v \in V} sim_{u,v}(r_{v,i} - \overline{r}_v)}{\sum_{v \in V} |sim_{u,v}|} \tag{2.10}$$

  This is the aggregator commonly used in collaborative filtering today, and the one used in [73].

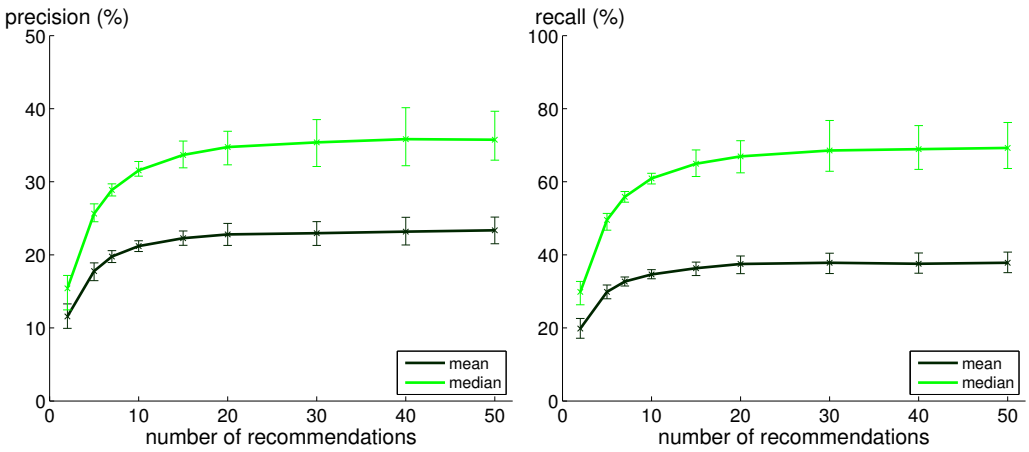
- the median of the neighbouring ratings weighted by their similarity. Assuming that the neighbours are ordered by increasing rating, the median is the rating r_i given by the smallest user i such that

$$\sum_{j=1}^{i} sim_{n_j,u} \geq \sum_{j=i+1}^{n} sim_{n_j,u} \tag{2.11}$$

In cases of ties, we break them at random.

The following results are based on an empirical study of a collaborative filtering system using the MovieLens[2] dataset. The dataset contains 1682 movies rated by 943 users. 100'000 ratings ranging from 1 to 5 were given by these users, and each user rated at least 20 movies.

[2] http://grouplens.org/datasets/movielens/



(a) Precision

(b) Recall

Figure 2.11: Average precision and recall of recommendations, with confidence intervals at 95%.

Accuracy

We evaluate the quality of the different algorithms for computing the predictions using precision and recall [42]. The mean absolute error (MAE) is difficult to compare since the median uses only a restricted set of values. Therefore, the MAE is not indicative of the actual performance of the recommender system.

We ran a 5-fold cross validation with disjoint test sets (80% training and 20% testing). We present average performances with confidence intervals at 95% (bootstrap sampling, 10'000 samples).

Figure 2.11 shows that the mean is by far not the best aggregator for recommendations; both precision and recall are significantly higher when the median is used. This difference is more marked because the Pearson similarity metric is more likely to include outliers among the neighbours, and it seems that the robustness of the median allows the recommender to take advantage of the Pearson metric in a stronger way than when the mean is used.

Resilience Against Attacks

Since recommendations are different for each user, we can no longer give a single number of ratings required to change this ranking. Instead, we consider the robustness of the average recommendation received by each user. We considered scenarios where an attacker wants to push an item into users' recommendations by inserting fake user profiles that provide high ratings for that item. In particular, we implemented the *average attacks* described by Mobasher et al. [73] with 150 (15%) attack profiles, and 5% of filler items. We characterise robustness by the *hit ratio*, defined as the percentage of times that the promoted item is recommended in a

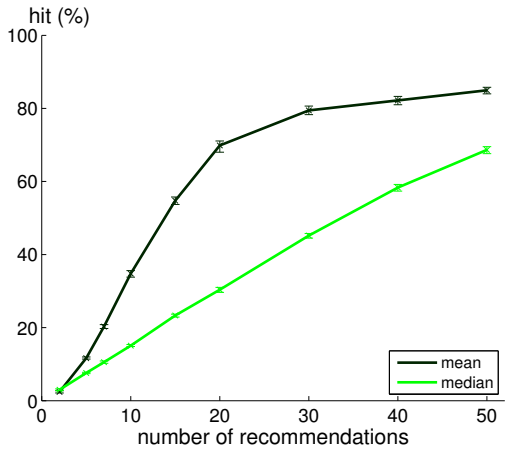


Figure 2.12: Average hit ratio as a function of the size of recommended items, with confidence intervals at 95%.

recommendation list as a result of the attack. We randomly select 50 target items on which we perform the attack, and select at random 50 users to evaluate the hit ratio. We present average performances over 10 runs and confidence intervals at 95% (bootstrap sampling, 10'000 samples).

Figure 2.12 shows the hit ratios as a function of the number of recommendations. We can see that when the number of recommended items is small, aggregating ratings by the median makes the recommender very resilient against attack, and it remains more resilient than the mean even when many recommendations are given. This again shows the much greater robustness of the median as a rating aggregator.

2.8 Conclusion and Future Work

We have used a new form of experiment based on a game to understand the behaviour of people in certain scenarios that exploit the wisdom-of-the-crowd effect. We have focused on the behaviour of users who are well-intentioned and act so as to make the aggregate reflect their belief of the true signal value, either through altruism or through incentives based on the success with other users. The fishing game closely mirrors eliciting opinions or observations, but in a completely artificial context where players' beliefs can be perfectly manipulated. Using two experiments, we have demonstrated that while the median aggregator elicits more truthful reports, its overall accuracy is not better than that of the mean since it does not perform the most accurate aggregation. We have shown that a different aggregator, the 40%-truncated mean, provides the best tradeoff between accuracy and truthfulness for the static setting, while for the dynamic setting the answer is not clear.

The fact that the median elicits the most truthful reports can be explained by the *Lake Wobegon* effect [39], i.e. people tend to trust their own observations more than others'. Actually, a user

who trusts others' reports should be most truthful when aggregation is via the mean, as this is the most accurate aggregator.

Our results are useful in the design of human computation tasks in general, such as rating and reputation, community sensing, opinion polls or question-and-answer websites, where users are motivated by the quality of the final result. Here there are often very few reports and so the difference between the results obtained with different aggregators is significant: using the median cuts the average error in half. To obtain this improvement, it is important that users know and understand the aggregator that is being applied. An important application of our results is in crowdsourcing tasks where the wisdom-of-the-crowd effect is important. Our findings show that by carefully choosing the right aggregator, it is possible to collect less reports and at the same time achieve a good quality of the output, thus it lowers the cost of crowdsourcing tasks by recruiting less workers.

As future work and with the help of this methodology, it would be possible to study different signal distributions such as the bimodal, explore new aggregators like the geometric mean, and investigate the effect of discretization of the reports. It would be also interesting to extend the framework to multi-dimensional signals, such as present when products have overall ratings and ratings per feature.

3 Recommending News Articles

3.1 Introduction

The proliferation of online news creates a need for filtering interesting articles. Compared to other products, however, recommending news has specific challenges: news preferences are subject to trends, users do not want to see multiple articles with similar content, and frequently we have insufficient information to profile the reader.

The first recommender systems were originally designed for news forums. Since then, they have been used with considerable success for products such as books and movies, but have found surprisingly little application in recommending news articles, due to the unique challenges of the area.

When users are identifiable as regular visitors to a news website, techniques from product recommendation can be adapted [23, 48]. However, most websites operated by individual newspapers do not have a strong base of electronic subscribers. Visitors to these websites are casual users, often accessing them through a search engine, and little is known about them except what can be gathered through an ephemeral browsing history.

The main page of a news website is already a set of recommended articles, which simultaneously addresses the needs of many users (Figure 3.1). More specific recommendations are sometimes available to readers of individual articles. There are two shortcomings to this strategy: first, recommendations are usually edited manually, and second, they only consider the last article read. Our goal is to construct recommendations automatically and use the complete browsing history as a basis for giving personalized recommendations.

In principle, common recommender techniques such as collaborative filtering could be applied to such a task, and have been adapted to temporal sequences [108, 95, 84]. However, they face several challenges specific to news. First, news are rapidly evolving: new stories and topics appear and disappear quickly, and old news are no longer interesting. Second, recommendations should provide added value, and not just consist of the most popular stories

Figure 3.1: A story with dynamic recommendations on the right side, and manually-generated recommendations on the bottom left (red-dashed areas).

that the reader would have already seen on the front page.

In the first part of this chapter, we address these issues by proposing a new class of news recommendation systems based on Context Trees (CT), which provide recommendations and are updated fully incrementally. We emulate the process involved in implementing a recommender system on a real website by using *offline* datasets. We show that context-tree recommendations have state-of-the-art performance both with respect to prediction accuracy and to recommendation novelty, which is crucial for news articles since users want to read stories they do not know.

Unfortunately, an offline evaluation is very artificial because it is not possible to assess the impact of recommendations on real users. In an online environment, the evaluation are motivated by different factors, and the way to measure the performance of a recommender system is different.

In the second part of this chapter, we focus on *online* evaluation of our state-of-the-art algorithms. When conducting online evaluation, one must pay careful attention to the requirements of such evaluation. The framework involved in this online evaluation has to be fast. The framework must provide real-time recommendations as soon as possible, without making the users wait. Second, it must be reliable. It is not acceptable for a newspaper website to suffer from crashes. Third, a flexible design is important. It should be easy to add new components or extend recommender systems. Finally, it must be scalable. News websites are subject to unpredictable visit peaks and the framework must be able to handle them by delivering recommendations on time and without problems.

Unfortunately, it is not possible to use current open-source platforms because they are not tailored to the specific needs of news recommendations and thus are difficult to adapt to the news domain. We introduce the PEN recsys framework for online evaluation of news recommender systems. The PEN recsys framework is currently (as of April 2014) in use by the news website swissinfo.ch[1] for evaluating the recommender systems presented in this chapter.

3.1.1 Contributions

In this chapter, we study how recommendations for news articles should be performed with the help of implicit feedback of the users. First, we introduce a new class of recommender systems based on context trees. In an *offline* evaluation setting, we show that they make accurate and novel recommendations, and that they are sufficiently flexible for the challenges of news recommendations.

Second, we discuss how to apply news recommender systems to live-traffic website. In order to do so, we described the design and implementation of a new online evaluation framework.

Finally, we illustrate the difference between offline and online evaluations. This chapter builds on the results published in [38, 32, 34, 33].

3.1.2 Related Work

In general, there are two classes of recommender systems: collaborative filtering [96], which use similar users' preferences to make recommendations, and content-based systems [68], which use content similarity of news items.

The Grouplens project is the earliest example of collaborative filtering for news recommendation, applied to newsgroups [87]. News aggregation systems such as Google News [23] also implement such algorithms. Google News uses probabilistic latent semantic indexing and MinHash for clustering news items, and item covisitation for recommendation. Their system builds a graph where the nodes are the stories and the edges represent the number of covisitations. Each of the approaches generates a score for a given news, aggregated into a single score using a linear combination.

Content-based recommender system is more common for news personalization [7, 1, 48]. NewsWeeder [59] is probably the first content-based approach for recommendations, but applied to newsgroups. NewsDude [7] and more recently YourNews [1] implemented a content-based system.

It is possible to combine the two types in a hybrid system [11, 65, 64]. For example, Liu et al. [65] extend the Google News study by looking at the user click behaviour in order to create

[1] www.swissinfo.ch

accurate user profiles. They propose a Bayesian model to recommend news based on the user's interests and the news trend of a group of users. They combine this approach with the one by Das et al. [23] to generate personalized recommendations. Li et al. [64] introduce an algorithm based on a contextual bandit which learns to recommend by selecting news stories to serve users based on contextual information about the users and stories. At the same time, the algorithm adapts its selection strategy based on user-click feedback to maximize the total user clicks.

We focus on a class of recommender systems based on context trees. Usually, these trees are used to estimate Variable-order Markov Models (VMM). VMMs have been originally applied to lossless data compression, in which a long sequence of symbols is represented as a set of contexts and statistics about symbols are combined into a predictive model [88]. VMMs have many other applications [4].

Closely related, variable-order hidden Markov models [102], hidden Markov models [74] and Markov models [79, 92, 24] have been extensively studied for the related problem of click prediction. These models suffer from high state complexity. Although techniques [107] exist to decrease this complexity, multiple models have to be maintained, making these approaches not scalable and not suitable for online learning.

Few works [108, 95, 84] apply such Markov models to recommender systems. Zimdars et al. [108] describe a sequential model with a fixed history. Predictions are made by learning a forest of decision trees, one for each item. When the number of items is big, this approach does not scale. Shani et al. [95] consider a finite mixture of Markov models with fixed weights. They need to maintain a reward function in order to solve a Markov decision process for generating recommendations. As future work, they suggest the use of a context-specific mixture of weights to improve prediction accuracy. In this work, we follow such an approach. Rendle et al. [84] combine matrix factorization and a Markov chain model for baskets recommendation. The idea of factoring Markov chains is interesting and could be complementary to our approach. Their limitation is that they consider only first-order Markov chains. A bigger order is not tractable because the states are baskets which contain many items.

Over the last few years, researchers in the recommender systems community have developed open-source libraries which try to bring a growing number of recommender algorithms under one roof [31, 28, 2, 69, 86, 101]. Most of these libraries are designed for research purposes to conduct offline evaluations and only a few target online evaluations on production websites [101, 86]. In the following, we present the ones that are still actively under development and free/open source.

MyMediaLite [31] is an actively maintained C# library. It contains simple baseline techniques such as slope-one or random/most popular item, several variants of k-nearest neighbour models and some matrix factorization methods. Unfortunately, MyMediaLite is not thread-safe, and thus is not suitable for concurrent access required in a highly-dynamic environment such as the news domain.

Apache Mahout [2] is a distributed machine learning library in Java. Although it is not specifically developed for recommendations, it contains some collaborative filtering algorithms. This library implements the Map/Reduce paradigm and is tailored for distributed systems. As we discuss in Section 3.6.2, standard news websites do not need such complex distributed framework because it brings an heavy overhead when deployed on a single server.

LensKit [28] is a Java-based library developed for offline research environments. It focuses on collaborative filtering techniques.

GraphLab [69] is a C++ collection of machine learning toolkits on graphs. One toolkit is dedicated to collaborative filtering with implementations of Alternating Least Squares, Stochastic Gradient Descent and Singular Value Decomposition.

easyrec [86] is a web service in Java generating recommendations through an API. *easyrec* is designed for online evaluation. However, it is not clear which algorithms are available. Recently, *easyrec* reached the final stage of its research project and it is also not clear whether this engine will stay free and open source or not.

reclab [101] provides a Java API to build recommender models. It does not come with any implemented algorithms, and it is designed to run together with a cloud service provided by the developers. Unfortunately, this library seems no longer active.

These libraries are all useful for research purposes but cannot be used on production websites due to restrictions on licenses for commercial usages. In addition, most of them are designed for offline evaluation.

Although the first recommender systems were originally designed for news forums [87], the literature describing actual implementations and evaluations on live news websites is scarce.

Liu et al. [65] analyse the deployment of a hybrid recommender system on the news aggregator Google News. They compare their method against the existing collaborative filtering system implemented by Das et al. [23], and consider only logged-in users for the evaluation. They show a 30% improvement over the existing collaborative filtering system.

Kirchenbaum et al. [57] conduct an online evaluation with logged-in users of several recommender systems for news articles on Forbes.com. They report that an hybrid-based system performs the best, with a 37% improvement over popularity-based methods.

Said et al. [91, 90] study the weekly and hourly impressions and click-through rates in the Plista news recommender framework [56], which delivers recommendations to multiple news websites. They observe that live evaluation is sensitive to external factors not necessarily related to recommendations. They also identify trends in recommendations related to the type of news websites (traditional or topic-focused news sources). Readers of topic-focused websites are less likely to take recommendations than readers of traditional news websites [90]. Unfortunately, it is not clear which recommender algorithm is in used in Plista framework and

during their analysis.

Our live evaluation is similar to the one of Kirchenbaum et al. [57], but differs in two crucial points. First, we consider anonymous users for whom it is impossible to track across multiple visits. Second, the nature of the websites (Forbes.com and swissinfo.ch) are not the same, and thus readers' behaviours are different [90].

3.2 Context Trees

Because of the sequential nature of news reading, it is intuitive to model news browsing as a k-order Markov process [95]. The user's state can be summarised by the last k items visited, and predictions can be based only on this information. Unfortunately, it is not clear how to select the order k. A *variable-order Markov model* (VMM) alleviates this problem by using a context-dependent order. In fact, VMM is a special type of context-tree model [4].

There are two key ideas behind a CT recommender system. First, it creates a hierarchy of contexts, arranged in a tree such that a child node completely contains the context of its parents. In this work, a context can be the set of sequences of news items, sequence of topics, or a set of topic distributions. As new articles are added, more contexts are created. Contexts corresponding to old articles are removed as soon as they disappear from the current article pool.

The second key idea is to assign a local prediction model to each context, called an expert. For instance, a particular expert gives predictions only for users who have read a particular sequence of stories, or users who have read an article that was sufficiently close to a particular topic distribution.

In the following, we first introduce the notion of context tree. Then, we describe various prediction models, how to associate them with the context tree and combine them in order to make recommendations.

3.2.1 Sequence Context Tree

When a user browses a news website, we track the sequence of articles read.

Definition 4. *A* sequence $\mathbf{s} = \langle n_1, \ldots, n_l \rangle$ *is an ordered list of articles* $n_i \in \mathcal{N}$ *read by a user, and we denote* $\mathbf{s}_t$ *the sequence of articles read until time* t*. We write the set of all sequences by* $\mathcal{S}$*.*

Note that a sequence can also be a sequence of topics of articles.

A context tree is built based on these sequences and their corresponding suffixes.

Definition 5. *A* k*-length sequence* ξ *of is a* suffix *of a* l*-length sequence* $\mathbf{s}$*, if* $k \leq l$*, and the last elements of* $\mathbf{s}$ *are equal to* ξ*, and we write* $\xi \prec \mathbf{s}$ *when* ξ *is a suffix of* $\mathbf{s}$*.*

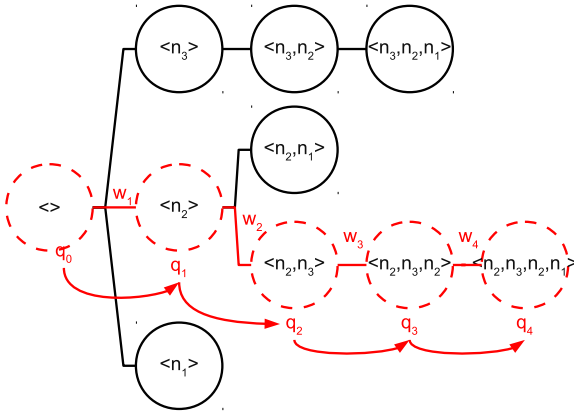


Figure 3.2: VMM context tree for the sequence $\mathbf{s} = \langle n_1, n_2, n_3, n_2 \rangle$. Nodes in red-dashed are active experts $\mu \in A(\mathbf{s})$.

For instance, one suffix ξ of the sequence $\mathbf{s} = \langle n_1, n_2, n_3, n_4 \rangle$ is given by $\xi = \langle n_3, n_4 \rangle$.

If two sequences have similar context, the next article a user wants to read should also be similar.

Definition 6. *A* context $S = \{\mathbf{s} \in \mathscr{S} : \xi \prec \mathbf{s}\}$, $S \subset \mathscr{S}$ *is the set of all possible sequences* $\mathscr{S}$ *ending with the suffix* ξ.

We can now give a formal definition of a context tree.

Definition 7. *A* context tree $\mathscr{T} = (\mathscr{V}, \mathscr{E})$ *with nodes* $\mathscr{V}$ *and edges* $\mathscr{E}$ *is a partition tree over the contexts* $\mathscr{S}$. *It has the following properties: (a)* The set of contexts at a given depth forms a partition*: If* $\mathscr{V}_k$ *are the nodes at depth* k *of the tree, then* $S_i \cap S_j = \emptyset \ \forall i, j \in \mathscr{V}_k$, *while* $\bigcup_{i \in \mathscr{V}_k} S_i = \mathscr{S}$ *(b)* Successive refinement*: If node* i *is the parent of* j *then* $S_j \subset S_i$.

Thus, each node $i \in \mathscr{V}$ in the context tree corresponds to a context S_i. Initially the context tree $\mathscr{T}$ only contains a root node with context $S_0 = \mathscr{S}$. Every time a new article n_t is read, the active leaf node is split in a number of subsets, which then become nodes in the tree. This construction results in a variable-order Markov model, illustrated in Figure 3.2.

The main difference between news articles and products is that articles continuously appear and disappear, and the system thus maintains a current article pool that is always changing. The model for recommendation changes along with the article pool, using a dynamically evolving context tree. As new articles are added, new branches are created corresponding to sequences or topic distributions. At the same time, nodes corresponding to old articles are removed as soon as they disappear from the current pool.

3.2.2 Topic Distribution Context Tree

Because of the large number of news items relative to topics, a context tree on topics might make better predictions. In particular, stories that have not been read by anyone can be recommended thanks to topic similarity. In this type of tree, each context represents a subset of the possible *topic distributions* of the last read article. The structure of the tree is slightly different and is modelled via a k-d tree.

A k-d tree is a binary tree that iteratively partitions a k-dimensional space $\mathscr{S}$ into smaller sets [6]. The i-th node corresponds to a hyper-rectangle $S_i \subset \mathscr{S}$ and has two children j, j' such that $S_j \cup S_{j'} = S_i$ and $S_j \cap S_{j'} = \emptyset$. In particular, the two children are always defined via a hyperplane splitting S_i in half, through the center of S_i, and which is perpendicular to one principal axis. In practice, we simply associate each node to one of the k axes based on the depth such that we cycle through all possible axes: $a = depth \bmod k$. The set $\mathscr{S}$ is $[0,1]^k$, the set of k-dimensional multinomial distributions on the possible topics.

In analogy to the sequence CT, a context is a hyper-rectangle S_i and a suffix is the center θ of S_i in a topic distribution CT.

For instance, consider a node i with center $\theta \in S_i$ and associated axis a. Its two children correspond to two sets of topic distributions: Its left child j contains the distributions $\theta' \in S_i$ with $\theta'_a < \theta_a$, while its right child j' is the set on the other side of the hyperplane: $S_{j'} = \{\theta' \in S_i : \theta'_a \geq \theta_a\}$. When the system observes a new topic distribution θ, the distribution is added to the tree, and possibly the tree expands.

3.2.3 Experts

We assign a local prediction model called *expert* to each context (node) in the tree. More formally,

Definition 8. *An* expert μ_i *is a function associated with a specific context* S_i *that computes an estimated probability of the next article* n_{t+1} *given that context, i.e.* $\mathbb{P}_i(n_{t+1} \mid \mathbf{s}_t)$.

The user's browsing history $\mathbf{s}_t$ is matched to the context tree and identifies a path of nodes (see Figure 3.2). All experts associated with these nodes are called *active* and are responsible for the recommendation.

Definition 9. *The set of* active *experts* $\mathscr{A}(\mathbf{s}_t) = \{\mu_i : \xi_i \prec \mathbf{s}_t\}$ *is the set of experts* μ_i *associated to contexts* $S_i = \{\mathbf{s} : \xi_i \prec \mathbf{s}_t\}$ *such that* ξ_i *are suffix of* $\mathbf{s}_t$.

3.2.4 Combining Experts for Predictions

The active experts $\mathscr{A}(\mathbf{s}_t)$ are combined by marginalizing to obtain a mixture of probabilities of all active experts:

$$\mathbb{P}(n_{t+1} = x \mid \mathbf{s}_t) = \sum_{i \in \mathscr{A}(\mathbf{s}_t)} u_i(\mathbf{s}_t)\, \mathbb{P}_i(n_{t+1} = x \mid \mathbf{s}_t), \tag{3.1}$$

with $u_i(\mathbf{s}_t) = \mathbb{P}(i|\mathbf{s}_t)$ being the probability of the i-th expert relevant for this context. These probabilities are derived as follows.

With each node i in the context tree we associate a weight $w_i \in [0,1]$ that represents the usefulness of the corresponding expert. Given a path in the context tree, we consider experts in the order of the most specific to the most general context, i.e. along the path from the most specific node to the root. In this process, with probability equal to the weight w_i we stop at a node without considering the more general experts. Thus, we take into account the relative usefulness of the experts.

Letting w_j be the probability of stopping at j given that we have not stopped yet, we thus obtain the probability u_i that the i-th expert is considered as $u_i(\mathbf{s}_t) = w_i \prod_{j:S_j \subset S_i}(1 - w_j)$ if $\mathbf{s}_t \in S_i$ and 0 otherwise.

The calculation of the total probability can be made efficiently via the recursion $q_k = w_k \mathbb{P}_k(n_{t+1} = x|\mathbf{s}_t) + (1 - w_k) q_{k-1}$, where q_k is the combined prediction of the first k experts. In Figure 3.2, the prediction of the root expert for the next item x is q_0, while q_4 is the complete prediction by the model for this sequence.

The weights are updated by taking into account the success of a recommendation. When a user reads a new article x, we update the weights of the active experts corresponding to the suffixes ending before x according to the probability $q_k(x)$ of predicting x sequentially via Bayes' theorem [26]:

$$w'_k = \frac{w_k \mathbb{P}_k(n_{t+1} = x \mid \mathbf{s}_t)}{q_k(x)}. \tag{3.2}$$

No other weights are updated[2]. Finally, we also update the local models of the active experts (see Section 3.3).

3.3 Expert Models

Recommending news articles depends on multiple factors: the popularity of the news item, the freshness of the story, the sequence of news items or topics that the user has seen so far. Thus, each expert is decomposed into a set of *local models*, each modelling one of these properties. The first model ignores the temporal dynamics of the process. The second model

[2] $w_0 = 1$ since we must always stop at the root node.

assumes that users are mainly looking at popular items, and the last model that they are interested in fresh items (i.e. breaking news).

3.3.1 Standard Model

A naïve approach for estimating the multinomial probability distribution over the news items is to use a Dirichlet-multinomial prior for each expert μ_i. The probability of reading a particular news item x depends only on the number of times α_x it has been read when the expert is active.

$$\mathbb{P}_i^{std}(n_{t+1} = x|\mathbf{s}_t) = \frac{\alpha_x + \alpha_0}{\sum_{j \in \mathcal{N}}(\alpha_j + \alpha_0)}, \tag{3.3}$$

where α_0 is the initial count of the Dirichlet prior.

The dynamic of news items is more complex. A news item provides new content and therefore has been seen by few users. News is subject to trends and frequent variations of preferences. We improve this simple model by augmenting it with models for *popular* or *fresh* news items.

3.3.2 Popularity Model

A news item $x \in \mathcal{P}$ is in the set of popular items $\mathcal{P}$ when it has been read at least once among the last $|\mathcal{P}|$ read news items. We compute the probability of a news item x given that x is *popular* as:

$$\mathbb{P}_i^{pop}(n_{t+1} = x|\mathbf{s}_t) = \frac{c_x + \alpha_0}{\sum_{j \in \mathcal{N}}(c_j + \alpha_0)}, \tag{3.4}$$

where c_x is the total number of clicks received for news item x. Note that c_x is not equal to α_x (Equation 3.3). α_x is the number of clicks for news item x when the expert is *active*, while c_x is the number of clicks received by news item x in *total* whether the expert is active or not.

The number of popular items $|\mathcal{P}|$ is important because it is unique for each news website. When $|\mathcal{P}|$ is small, the expert considers only the most recent read news. It is important to tune this parameter appropriately.

3.3.3 Freshness Model

A news item $x \in \mathcal{F}$ is in the set of fresh items $\mathcal{F}$ when it has not been read by anyone but is among the next $|\mathcal{F}|$ news items to be published on the website, i.e. a breaking news. We compute the probability of news item x given that x is *fresh* as:

$$\mathbb{P}_i^{fresh}(n_{t+1} = x|\mathbf{s}_t) = \begin{cases} \frac{1}{|\mathcal{F}|+1}, & \text{if } x \in \mathcal{F} \\ \frac{1}{(|\mathcal{F}|+1)(|\mathcal{N}|-|\mathcal{F}|)}, & \text{if } x \notin \mathcal{F}. \end{cases} \tag{3.5}$$

The number of fresh items $|\mathscr{F}|$ influences the prediction made by this expert, and is unique for each news website.

3.3.4 Mixing the Expert Models

We combine the three expert models using this mixture:

$$\mathbb{P}_i(n_{t+1} = x|\mathbf{s}_t) = \sum_{\tau \in \{std,pop,fresh\}} \mathbb{P}_i^\tau(n_{t+1} = x|\mathbf{s}_t) p_i^\tau. \tag{3.6}$$

There are two ways to compute the probabilities p_i^τ: either by using a Dirichlet prior that ignores the expert prediction or by a Bayesian update to calculate the posterior probability of each expert according to their accuracy.

For the first approach, the probability of the next news item being *popular* is:

$$\begin{aligned} p_i^{pop} = \mathbb{P}_i(n_{t+1} \in \mathscr{P}) &= \frac{\alpha_{pop} + \alpha_0}{(\alpha_{pop} + \alpha_0) + (\alpha_{notpop} + \alpha_0)} \\ &= \frac{\alpha_{pop} + \alpha_0}{2\alpha_0 + \sum_j \alpha_j}, \end{aligned} \tag{3.7}$$

where $\sum_j \alpha_j$ represents the number of times the expert μ_i has been active, α_{pop} and α_{notpop} the number of read news items which were respectively popular and not popular when the expert μ_i was active.

Similarly, the probability of the next news item being *fresh* is given by:

$$p_i^{fresh} = \mathbb{P}_i(n_{t+1} \in \mathscr{F}) = \frac{\alpha_{fresh} + \alpha_0}{2\alpha_0 + \sum_j \alpha_j}, \tag{3.8}$$

where α_{fresh} is the number of read news items which were fresh when the expert μ_i was active.

Noting that $\mathscr{P} \cap \mathscr{F} = \emptyset$, the probability of the next news item being neither popular nor fresh is:

$$p_i^{std} = \mathbb{P}_i(n_{t+1} \notin \mathscr{P} \cup \mathscr{F}) = 1 - \mathbb{P}_i(n_{t+1} \in \mathscr{P}) - \mathbb{P}_i(n_{t+1} \in \mathscr{F}). \tag{3.9}$$

It might happen that by using the Dirichlet priors, predictions are mainly made by only one expert model. To overcome this issue, we compute the probabilities p_i^τ, $\tau \in \{std, pop, fresh\}$ via a Bayesian update, which adapts them based on the performance of each expert model:

$$p_i^\tau \leftarrow \frac{\mathbb{P}_i^\tau(n_{t+1} = x|\mathbf{s}_t) p_i^\tau}{\mathbb{P}_i(n_{t+1} = x|\mathbf{s}_t)}. \tag{3.10}$$

3.4 Context-tree Recommender Systems

We describe here the general algorithm to generate recommendations for the class of context-tree recommender systems. This algorithm can be applied to domains other than news in which timeliness and concept drift are of concern. We then focus on the news domain and describe in more details three VMM-based recommender systems and one based on the k-d context tree.

3.4.1 General Algorithm

Algorithm 1 presents a sketch of the CT recommender algorithm. For simplicity, we split our system in two procedures: *learn* and *recommend*. Both are executed for each read article x of a user with browsing history $\mathbf{s}$ in an online algorithm such as [85, 66], without any further offline computation. The candidate pool $\mathscr{C}$ is always changing and contains the popular $\mathscr{P}$ and fresh $\mathscr{F}$ stories. The system estimates the probability of each candidate and recommends the news items with the highest probability. In order to estimate the probability of a candidate item, the system *1)* selects the active experts $\mathscr{A}(\mathbf{s})$ which correspond to a path in the context tree from the most general to the most specific context, *2)* propagates q from the root down to the leaf, i.e the most specific context. q at the leaf expert is the estimated probability of the recommender system for the candidate item x, i.e. $\mathbb{P}(n_{t+1} = x|\mathbf{s}_t)$ (see Equation 3.1).

3.4.2 Recommender Systems

We consider three VMM variants of recommender systems, and one based on the k-d context tree.

VMM Recsys

The standard VMM recommender builds a context tree on the sequences of news items. That is $\mathbf{s}_t = \langle n_1, \ldots, n_t \rangle$ is a sequence of news items, and each active expert predicts n_{t+1}, the next news item.

Content-based VMM (CVMM) Recsys

In order to build a CT on topic sequences, we find a set of topics for each story, and assign the most probable topic to the news item. We then perform predictions on *topics*.

More precisely, we use the Latent Dirichlet Allocation (LDA) [9] as a probabilistic topic model. After concatenating the title, summary and content of the news item together, we tokenize the words and remove stopwords. We then apply LDA to all the news stories in the dataset, and obtain a topic distribution vector $\theta^{(n)}$ for each news item n.

Algorithm 1 CT recommender system

procedure LEARN(x, $\mathbf{s}$, context set Ξ)
  $q \leftarrow 0$ and $t \leftarrow |\mathscr{A}(\mathbf{s})|$
▷ loop from most general expert μ_0 to most specific expert μ_t
  for $i \leftarrow 0, t$ **do**
    $p_i \leftarrow \mathbb{P}_i(n_{t+1} = x|\mathbf{s})$ ▷ ith expert prediction
    $q \leftarrow w_i p_i + (1 - w_i) q$ ▷ combined prediction
    $w_i \leftarrow \frac{w_i p_i}{q}$ ▷ weight update
    update $p_i^{std}, p_i^{pop}, p_i^{fresh}$ (Equation 3.7-3.9 or Equation 3.10)
  end for
  if $x \circ \mathbf{s} \notin \Xi$ **then** ▷ is the context in the tree?
    $\Xi = \Xi \bigcup \{x \circ \mathbf{s}\}$ ▷ add a new leaf.
  end if
end procedure
procedure RECOMMEND($\mathbf{s}$)
  for all candidate $n \in \mathscr{C}$ **do**
    $q^{(n)} \leftarrow 0$ and $t \leftarrow |\mathscr{A}(\mathbf{s})|$
▷ loop from most general expert μ_0 to most specific expert μ_t
    for $i \leftarrow 0, t$ **do**
      $q^{(n)} \leftarrow w_i \mathbb{P}_i(n|\mathbf{s}) + (1 - w_i) q^{(n)}$
    end for
  end for
  $\mathscr{R} \leftarrow$ sort all $n \in \mathscr{C}$ by $q^{(n)}$ in descending order
  return first k elements of $\mathscr{R}$
end procedure

We now define a context tree as follows. Let z_t be the most probable topic of the t-th news item. Then the *topic sequence* is $\mathbf{s}_t = \langle z_1, ..., z_t \rangle$ and ξ is a suffix of topic sequences. The context tree generates a topic probability distribution $\mathbb{P}(z_{t+1} = j | \mathbf{s}_t)$, while the LDA model provides us with a topic distribution $\mathbb{P}(z = j|n)$ for each news item n. These are then combined into the following score:

$$score(n \mid \mathbf{s}_t) = \max_j \{\mathbb{P}(z_{t+1} = j \mid \mathbf{s}_t) \cdot \mathbb{P}(z = j \mid n)\}. \tag{3.11}$$

The system recommends the articles with the highest scores.

Hybrid VMM (HVMM) Recsys

We combine the standard VMM with the content-based system into a hybrid version. The context tree is built on *topics*, similarly to the CVMM system, but the experts make predictions about *news items*, like the VMM system.

HVMM system builds a tree in the space of *topic sequences*. Each suffix ξ of size k is a sequence of most probable topics $\langle z_1, z_2, ..., z_k \rangle$. However, all predictive probabilities (Equation 3.1 and later) are defined on the space of news items.

k-d Context-Tree (k-CT) Recsys

The CVMM and HVMM structures make predictions on the basis of the sequence of most probable topics. Instead, we consider a model that takes advantage of the complete topic distribution of the last news item. We use a k-d tree to build a context model in the space of *topic distributions*.

Baselines

In addition, we have the following baselines:

Z-k is a fixed k-order Markov chain recommender similar to the ones by Zimdars et al. [108].
MinHash is the minhash system used in Google News [23].
MostPopular recommends a set of stories with the highest number of clicks among the last read news items.

3.5 Offline Evaluation

We investigate whether the class of CT recommender systems has an advantage over standard methods and if so, what is the best combination of partition and expert model.

We measure performance both with respect to accuracy and novelty of recommendations. Novelty is essential because it exposes the reader to relevant news items that she would not

	News stories	Visits	Clicks
TDG	10'400	600'256	1'069'131
24H	8'613	249'099	509'978

Table 3.1: Datasets after filtering.

have seen by herself. Obvious but accurate recommendations of most-popular items are of little use.

We evaluate our systems on two datasets described below. We examine on the first dataset the sensitivity of the CT models to hyperparameters. The second dataset is used to perform an unbiased comparison between the different models. We select the optimal hyperparameters on the first dataset, and then measure the performance on the second dataset. This methodology [5] mirrors the approach that would be followed by a practitioner who wants to implement a recommender system on a newspaper website.

3.5.1 Datasets

We collected data from the websites of two daily Swiss-French newspapers called *Tribune de Genève* (TDG) and *24 Heures* (24H)[3]. Their websites contain news stories ranging from local news, national and international events, sports to culture and entertainment.

The datasets span from Nov. 2008 until May 2009. They contain all the news stories displayed, and all the visits by anonymous users within the time period. Note that a new visit is created every time a user browses the website, even if she browsed the website before. The raw data has a lot of noise due to, for instance, crawling bots from search engines or browsing on mobile devices with unreliable internet connections. Table 3.1 shows the dataset statistics after filtering out this noise, and Figure 3.3 illustrates the distribution of visit length for each dataset.

3.5.2 Metrics

There has been a lot of discussions on the best way of evaluating recommender systems [43, 94]. The best would be to implement them on an actual site and measure the click rate on recommended items. Unfortunately, this is usually far too costly to do, and evaluation has to be carried out based on the behaviours that was observed without the recommender being present. In our case, we have visit histories from the newspaper websites, and we can evaluate how well our recommendations match the news items that readers selected themselves. It is clear that this is a somewhat inaccurate measure: *a)* the user may not have liked all the items she visited; *b)* the user may have preferred one of the recommended items to the one she clicked, so the fact that a recommended item was not visited does not mean the

[3] www.tdg.ch and www.24heures.ch

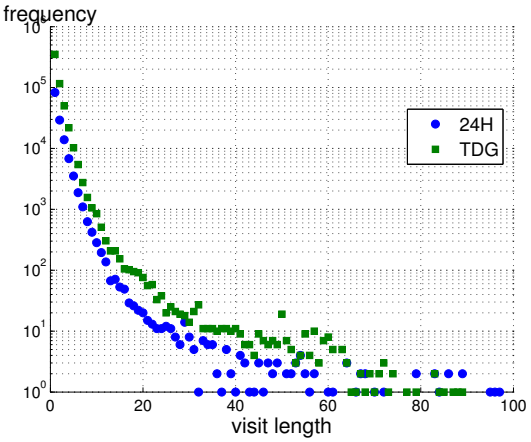


Figure 3.3: Distribution of the length of visits.

recommendation is bad.

We evaluate how good the systems are at predicting the future news a user is going to read. Specifically, we consider sequences of news items $\mathbf{s} = \langle n_1, n_2, ..., n_l \rangle$, $n_i \in \mathcal{N}, \mathbf{s} \in \mathcal{S}$ read by anonymous users. The sequences and the news items in each sequence are sorted by increasing order of visiting time. When an anonymous user starts to read a news item n_1, the system generates 5 recommendations. As soon as the user reads another news item n_2, the system updates its model with the past observations n_1 and n_2, and generates a new set of recommendations. Hence the training set and the testing set are split based on the current time: at time t, the training set contains all news items accessed before t, and the testing set has items accessed after t.

We consider three metrics. The first is the **Success@5** (s@5). For a given sequence $\mathbf{s} = \langle n_1, n_2, ..., n_t, ..., n_l \rangle$, a current news item n_t in this sequence, and a set of recommended news items $\mathcal{R}$, s@5 is equal to 1 if $n_{t+1} \in \mathcal{R}$, 0 otherwise.

The second metric is **personalized s@5**, where we remove the popular items $\mathcal{R}_\mathcal{T}$ from the set $\mathcal{R}$, to get a reduced set $\mathcal{R}_P = \mathcal{R} \setminus \mathcal{R}_T$. This metric is important because it filters out the bias due the fact that data is collected from websites which recommend the most popular items by default.

The final metric is **novelty**, defined by the ratio of unseen and recommended items over the recommended items: $novelty = |\mathcal{R} \cap \mathcal{F}|/|\mathcal{R}|$. This metric is essential because users want to read about something they do not already know.

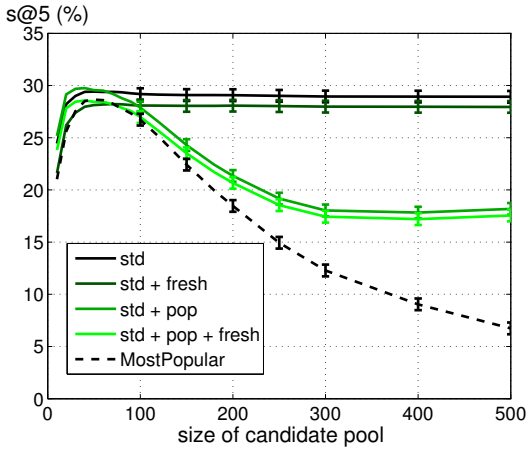
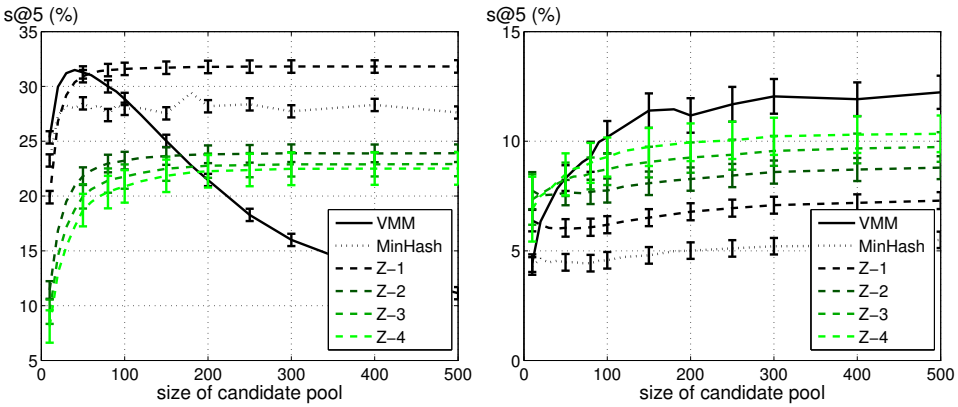


Figure 3.4: VMM recommender system: different mixtures of experts (Bayesian update, $|\mathscr{F}| = 10$).



(a) Non-personalized news items

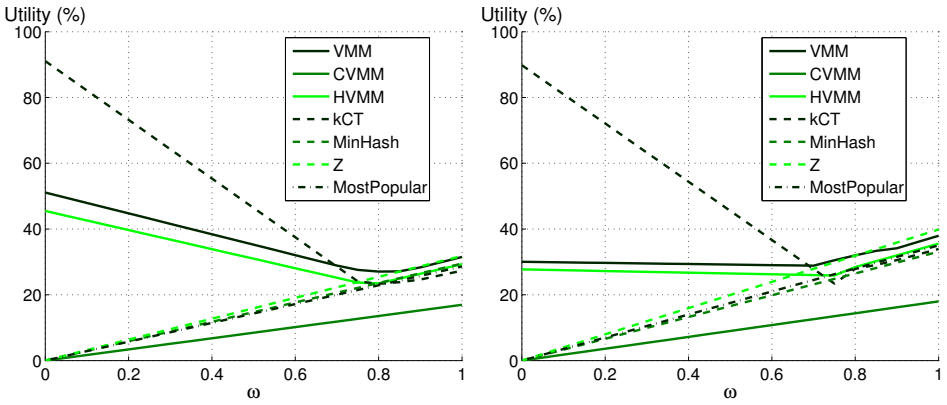
(b) Personalized news items

Figure 3.5: Accuracy for personalized and non-personalized news items.

3.5.3 Results

Sensitivity Evaluation

For all systems, we use a prior $\alpha_0 = 1/|\mathscr{N}|$ for the Dirichlet models, and the initial weights for the experts as $w_k = 2^{-k}$, where k is the depth of the node. We evaluated experimentally the optimal number of topics in the range of 30 to 500, and found that 50 topics bring the best accuracy. We varied the size of candidate pool: number of popular items $|\mathscr{P}|$ from 10 to 500, and fresh items $|\mathscr{F}|$ from 10 to 100. When the candidate set is small, the experts consider only the most recent read stories. We report averages over all recommendations with confidence intervals at 95%. We omit figures for the TDG dataset because we witnessed the same behaviours.



(a) Tuning: 24H dataset

(b) Testing: TDG dataset with optimal parameters from 24H dataset.

Figure 3.6: Expected performance curves: accuracy and novelty trade-off.

Although naïve, the approach of recommending the most popular stories is actually used very often on newspaper websites. This strategy does not pay off when the size of candidate pool increases. "Good" recommendations are drowned in popular items. This can also be seen by the fact that mixture of expert models integrating the popularity model are very sensitive to the number of popular items while others are more robust (e.g. Figure 3.4 for VMM recsys).

We noticed that, when using the Dirichlet priors to update the mixture probabilities, the prediction was mostly made by the popularity model, resulting in the same behaviour as the most-popular recommender system as the size of candidate pool increases. However, as the Bayesian update (Equation 3.10) adapts the probabilities based on the performance of each expert model, it is more robust when we increase the candidate set. We also observed that as the number of fresh items increased, CT models were getting slightly better.

When we look at the general accuracy of CT recommender systems (Figure 3.5(a)), their performance is close to the existing techniques. However when we consider only personalized items (Figure 3.5(b)), CT recommender systems outperform current techniques, showing that the order of the model is important. Indeed, we observed that the weights of the experts are well distributed over the space even for long sequences. If the sequence is not important, the weights of the experts would have been 0 except for the root expert, resulting in a performance similar to Z-1.

We have seen that recommending popular news is easy and relatively accurate. However, a recommender system with a high accuracy on an offline evaluation does not imply that in practice it will give useful recommendations to the users. In the context of news, novelty plays a crucial role, and the users expect both personalized and novel recommendations. In the next section, we study this trade-off.

Comparison

In practice, we may be interested in some particular mix of accuracy and novelty. We formalize this by defining a utility function: $U(\omega|D,A) = \omega * s@5 + (1-\omega) * novelty$, where ω specifies the trade-off between accuracy and novelty, D is the dataset (24H or TDG) and A is an assignment of parameters. For CT systems, the parameters are the number of popular $|\mathscr{P}|$ and fresh $|\mathscr{F}|$ items, whether the probabilities are computed via a Bayesian update or not, the mixture of experts (standard, popularity and/or freshness). It might be the case that different parameters are optimal for different utilities. The same holds for the parameters of the other methods we compare against.

To perform the comparison, we simulate the process of a designer who is going to tune each system on a small dataset (24H), before deploying the recommender online (on the TDG dataset). For any value of ω, we find the best parameters for the 24H dataset, and then measure the performance on the other dataset. This gives the Expected Performance Curve (EPC) [5], which provides an unbiased evaluation of the performance obtained by different methods.

Figure 3.6 illustrates the EPCs for 24H and TDG datasets. Figure 3.6(a) shows the optimal utility $U(\omega|D,A^*(\omega,D))$ with $A^*(\omega,D) = \arg\max_A U(\omega|D,A)$ for D = 24H dataset. Figure 3.6(b) shows the corresponding utility $U(\omega|D',A^*(\omega,D))$ achieved on the test dataset D' = TDG using the parameters found for the tuning dataset. First, we observe that all methods are robust in that they have similar performance in the testing dataset. Second, we observe that the purely content-based method (CVMM) performs poorly both with respect to novelty and accuracy. The hybrid approach (HVMM) is significantly better. Third, the approaches that disregard the content (VMM and Z) perform similarly in terms of accuracy, but only the VMM has a reasonable novelty. Finally, the k-d tree approach (kCT) has a much higher novelty than anything else. Thus, if one were to select a method based on performance on the tuning set, one should choose kCT for smaller values of ω and VMM for larger values.

3.6 PEN Recsys Framework

We are interested in *online* evaluation of state-of-the-art algorithms for *news* recommendations. Unfortunately, it is not possible to use current open-source platforms because they are not tailored to the specific needs of news recommendations and thus are difficult to adapt to the news domain [32].

The most important challenge of news recommendation is that news items are evolving very quickly. Stories and topics emerge and vanish rapidly and outdated stories are no longer interesting. Hence the recommender system must take into account these changes.

To this end, we present the PEN recsys framework for online evaluation of news recommender systems. PEN recsys is designed with 4 criteria in mind. First, it has to be *fast*. The framework must provide real-time recommendations as soon as possible, without making the users wait.

Second, it must be *reliable*. It is not acceptable for a newspaper website to suffer from crashes. Third, a *flexible* design is important. It should be easy to add new components or extend recommender systems. Finally, it must be *scalable*. News websites are subject to unpredictable visit peaks and the framework must be able to handle them by delivering recommendations on time and without problems.

Implementing a recommender system for production websites is challenging [78]. In the following, we describe our framework and explain our design choices. We discuss important factors to take into account when conducting online evaluation and report on our experience when deploying recommendations with a live-traffic website. The PEN recsys framework is currently in use by the news website swissinfo.ch[4] for evaluating various recommender systems.

3.6.1 Architecture

The PEN recsys framework consists of 6 main components. Figure 3.7 gives a brief overview of these components and their interactions.

The *dispatcher* randomly assigns a recommender system to a user, performing A/B or multivariate testing. The *recsys* $1, 2, 3, ..., k$ are the different algorithms to evaluate. Some algorithms rely on click statistics. The component *statistics* gathers click statistics about the stories. Other recommender systems need the content of news articles, or more specifically its topic distribution.

The component *topic model* is in charge of keeping the topic model up-to-date. We use the Latent Dirichlet Allocation (LDA) method as probabilistic topic model [9, 44]. Building a topic model takes time and resources. We implemented the offline LDA [9] in which we build the topic model at regular intervals and based on the number of new items. This option is useful in the case of a small corpus of news items or when the number of new items per day is small. In the case of a larger corpus or when the number of new items per day is more important, we also implemented an online version [44] which is useful for news items arriving in a stream.

Candidate items for recommendations are provided by the *candidates* component. This component periodically queries the news website for fresh, unseen items and keeps track of the clicked ones. Let $\mathscr{F}$ be the set of fresh items and $\mathscr{P}$ the set of clicked items such that $\mathscr{F} \cap \mathscr{P} = \emptyset$. We define the candidate set $\mathscr{C}$ as $\mathscr{C} = \mathscr{F} \cup \mathscr{P}$. It thus contains fresh items but also old ones. It is possible to control the size of $\mathscr{C}$ thanks to two parameters: the size of the fresh set $\mathscr{F}$ and the size of the clicked set $\mathscr{P}$. The clicked set contains the latest $|\mathscr{P}|$ clicked news items.

The *performance* component generates periodically performance reports of the algorithms under evaluation. The *database* stores the clicks, statistics and performance reports for offline

[4] www.swissinfo.ch

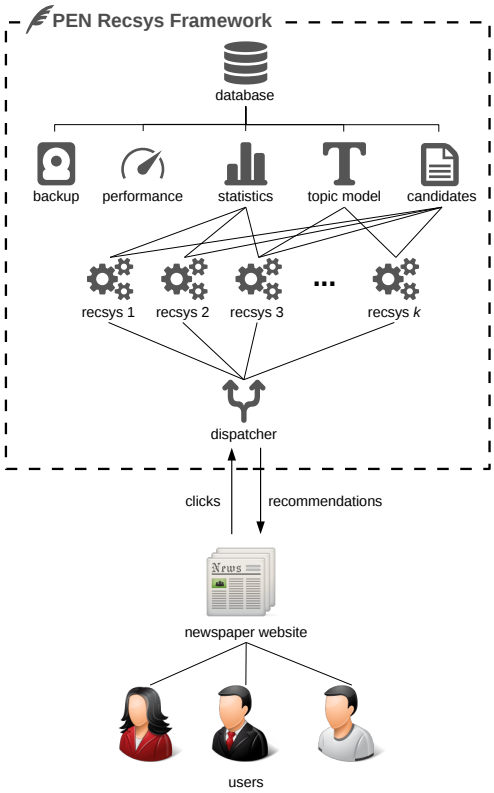


Figure 3.7: System architecture and components

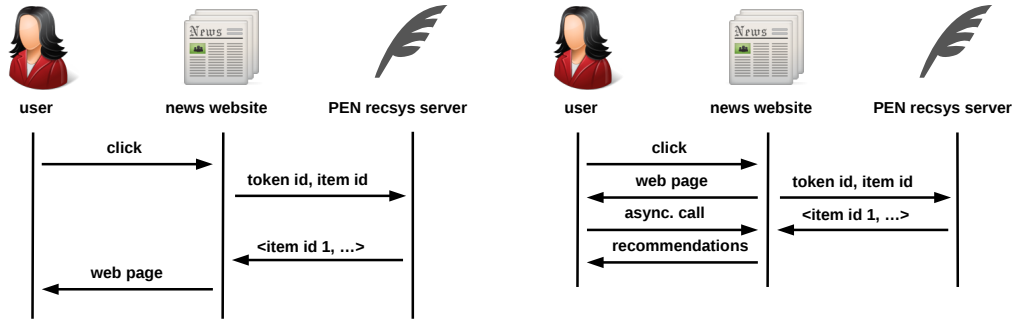
analysis. It provides a simple abstraction so that algorithms can access useful information such as the user history, preference or item statistics.

Finally, the *backup* component periodically triggers backup to the hard disk of the various states of the system such as the current topic model and click statistics. This is useful if we want to roll back to a previous set of parameters.

It is important to deliver recommendations to the user as soon as possible. With that in mind, the platform is designed to reduce this latency to the minimum. Hence tasks that are not essential for generating recommendations are run in the background. For instance, the database is known to be a bottleneck. Thus statistics are first cached in memory and later stored in the database when resources are available.

When a reader clicks on a news item, the website sends to the PEN recsys a token id representing the reader and the respective item id. Note that the token id can be the reader id when the reader logs into the website or a completely anonymized string representing the current session of a non-logged reader. In our case, the website sends the latter.

The PEN recsys framework follows the software-as-a-service paradigm and is implemented



(a) Synchronous

(b) Asynchronous

Figure 3.8: Two possible solutions to deliver news recommendations.

using Java EE technologies. It scales very well since each component can be physically located on different sites.

3.6.2 Real-time Recommendation and Latency

In a highly-dynamic domain such as news, providing *real-time* recommendations is crucial and challenging. Most of the standard recommender algorithms cannot be applied directly to the news domain [8, 38]. They are designed for product recommendations where updating a model once a day or week is acceptable. However, in our case the models need to be reactive and updated on-the-fly as fresh news stories come in.

We paid careful attention to optimize the PEN recsys framework in such a way that it generates recommendations very quickly. From a server-side perspective, it is important to remove any system bottleneck. For instance, storing information into a database is a known bottleneck. It is thus better to use caching methods which allow to keep information in memory first, and save them in a database when resources are available later. Moreover, the PEN recsys framework relies on threads and concurrent data structures. This allows to generate recommendations simultaneously to multiple readers.

A naïve solution to deliver a news content to the user is to use a synchronous approach. In Figure 3.8(a), the user clicks on a news item. Then, the swissinfo.ch's server queries the PEN recsys server. The PEN recsys generates recommendations and sends them back to the swissinfo's server, which encapsulates them in the web page. The swissinfo.ch's server sends the page with the recommendations back to the user. As a result, the user has to wait to see the content of the page until the recommendations are ready. This approach is not efficient.

We choose an asynchronous solution depicted in Figure 3.8(b). When a user clicks on a news item, swissinfo.ch's server sends the content of the page to the user's browser and at the same time queries the PEN recsys server. At that moment, the PEN recsys server generates

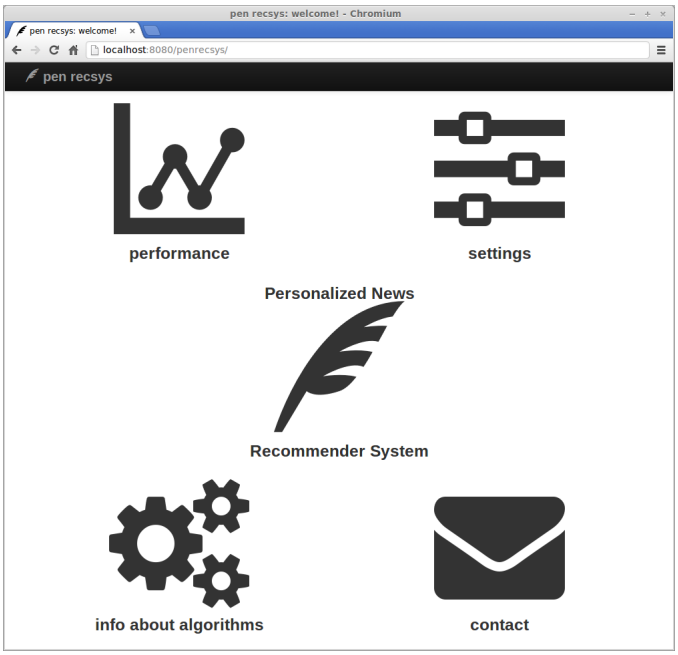


Figure 3.9: Screenshot of the main panel

recommendations and sends them back to the swissinfo.ch's server. The browser makes an asynchronous call, i.e. with Ajax technology, to the swissinfo.ch's server in order to fetch these recommendations. swissinfo.ch forwards the recommendations to the browser as soon as they are ready. If they are ready before the asynchronous call, the swissinfo.ch's server stores them temporarily. The advantage of this approach is that the original content of the page is not blocked while the recommendations are generated.

swissinfo.ch had another requirement: deliver recommendations within 1 second. When this threshold expires, no recommendations are displayed. So far, we never witnessed such behaviour.

For our online evaluation, the PEN recsys framework is running on a standard workstation with a dual-core CPU @ 2.6Ghz and 4GB of RAM. At the time of writing, the current memory usage is 48% with a CPU load of 51%. It handles visit peaks smoothly and recommendations are always delivered on time in less than 30ms (including connection overhead).

Consequently, we believe that standard news websites do not need an heavy system such as Apache Mahout [2] but special cares must be taken by the researcher when designing and implementing the system.

3.6.3 Interfaces

The PEN recsys framework has a web-based control panel (Figure 3.9) allowing the researcher to configure the general behaviour of the framework, enable/disable an algorithm or fine tune

its parameters. In Figure 3.10(a), some recommender systems are enabled for an online multivariate evaluation (random articles, most popular recommender and VMM recommender [32]), and the specific options of the VMM recommender are displayed.

The researcher can also check the performance of the enabled algorithms. Figure 3.10(b) shows the performance panel with 3 metrics: success@k, mean average precision and the average clicks per visit.

3.7 Live Evaluation at swissinfo.ch

Since July 2013, we are evaluating CT-based recommender systems with live traffic on the news website *swissinfo.ch*. swissinfo.ch is a 10-language news website owned by the Swiss broadcasting corporation. Its content is produced specifically for an international audience with interests in Switzerland. swissinfo.ch gives priority to in-depth information on politics, society, business, culture and science & technology. It has more than 1.7 million clicks per month. We have deployed recommendations only on the English version of the website, which accounts for more than 25% of the total traffic.

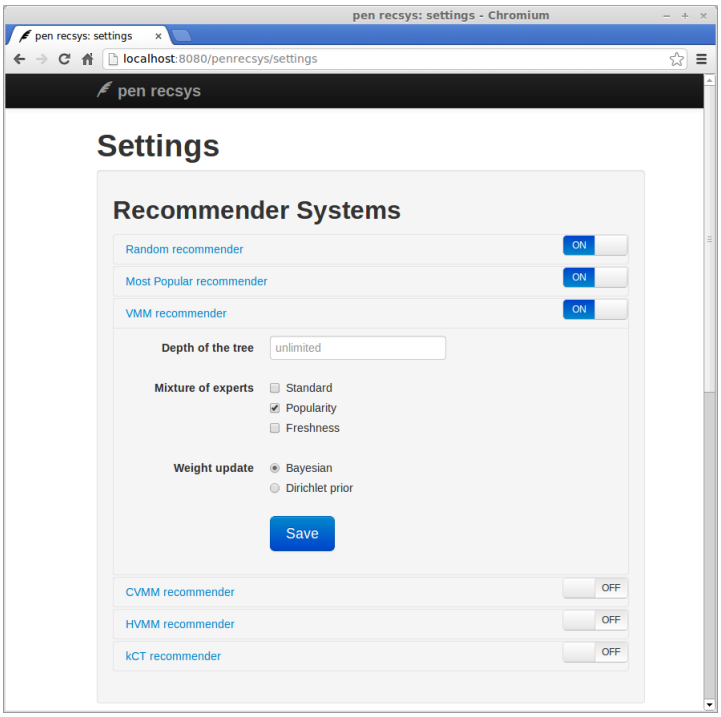
3.7.1 Baselines

In addition to context-tree recommender systems, we consider for the evaluation at swissinfo.ch the following baselines:

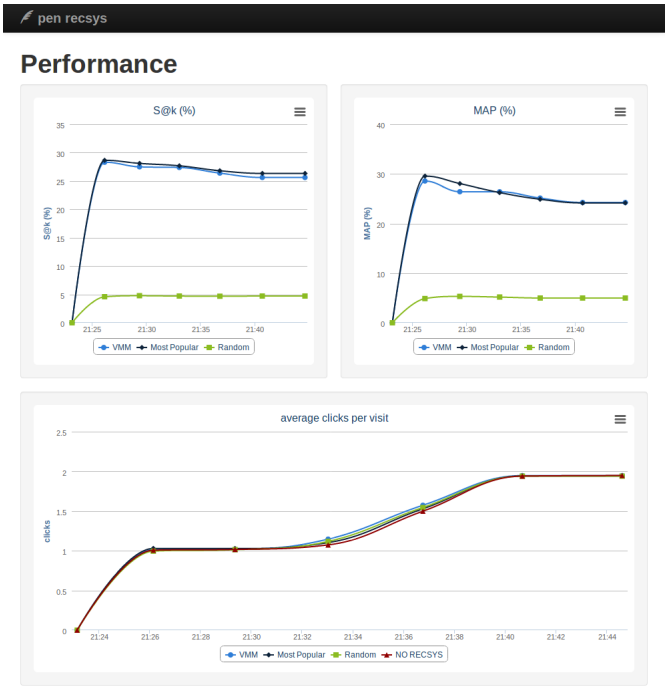
Most Popular recommends a set of stories with the highest number of clicks among the last read news items. This strategy is commonly implemented on most newspaper websites. It does not recommend any new items, but only the ones that a reader would have already seen on the front page.

Random recommends a set of stories at random. This strategy has the advantage of recommending very diverse and potentially novel items.

Although the literature on news recommender systems is rich [7, 11, 1, 48, 64], we decided to select only these baselines for the following reasons. First, the most popular and random strategies are de-facto standards, very easy to implement and compare performance. Researchers can easily compare their results to ours relative to these baselines. Second, reproducing previous research takes time and correctly implementing an existing approach is tricky, notwithstanding it is well described. Third, most of these algorithms have not been tested on live traffic websites and thus there is no guarantee of scalability and if they meet real-time requirements. Last, we need to limit ourselves to compare simultaneously 3 to 4 algorithms. More algorithms would dilute the clicks and would not guarantee statistical significance of the evaluation.



(a) Setting panel (partial)



(b) Performance panel

Figure 3.10: Screenshots of settings and performance panels

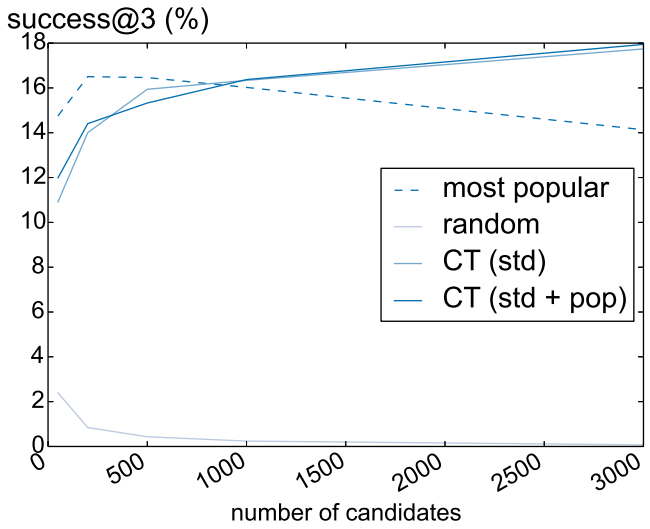


Figure 3.11: Offline *predicted* accuracy for different sizes of candidate set.

3.7.2 Offline

Only a small set of recommender systems can be evaluated on a live website, and it has to be carefully selected. The offline evaluation of Section 3.5 helped us to select the recommender systems and their parameters that are the most likely to bring good recommendations. However, the evaluation of Section 3.5 was conducted on two different newspaper websites, and not on swissinfo.ch. Thus, we had access to a 3-weeks "recommendations-free" clicks log from swissinfo.ch. During these 3 weeks, the website did not have any recommender systems deployed such as popularity-based methods ("top 10 popular items"). The log contains 227'831 clicks on 28'525 stories.

For a given recommender system and some fixed set of parameters, we imitated the online environment by sending clicks in the same sequence as saved in the log to the PEN recsys framework, and recorded the performance. When the recommender system receives a click on some article, it generates 3 recommendations. The performance is measured as the *success@3*, i.e. whether the next clicked article is in the recommendation set or not.

There are many different metrics to assess the performance of a recommender systems [42, 22]. For this offline evaluation, we selected the success@3 because we want to compare it with the actual click-through rate of the live evaluation.

In addition to the baselines that recommend most popular and random items, we implemented a context-tree system with the *std* model, and one with a mixture of *std* and *pop* models. We were not able to add the *fresh* model because the log does not contain information about freshness.

Figure 3.11 illustrates the predicted accuracy for various sizes of the candidate set. When the number of candidates increases, the performance of the random strategy drops because interesting stories are diluted in the set. There is an optimal size for recommending the most popular items around 200 candidates. Increasing the number of candidates after this point does not improve the performance for the most popular strategy. However, CT recommender systems take advantage of more candidates to bring better recommendations. In the rest of the evaluations, we decided to pursue a conservative approach where we select parameters such that baselines are optimal, and we pick 200 as size of candidate set. This conservative choice of parameters is not in favour of CT recommenders but in that way, the performance of CT recommenders can only be improved.

Figure 3.12(a) shows the predicted accuracy over time. In this figure, we only implemented the *std* model in the CT recommender as it performs equally good as when we add the popular model. It takes about one week to stabilize the performance. As expected, recommending the most popular items outperforms other strategies. The random strategy is the poorest with an accuracy close to 1%.

3.7.3 Online

All recommender systems are deployed with the PEN recsys framework, specifically designed to conduct multi-variate online evaluation of news recommender systems. With the online evaluation, we want to address the following questions:

Question 1. *Do context-tree recommender systems bring an improvement in click-through rate over baseline methods?*

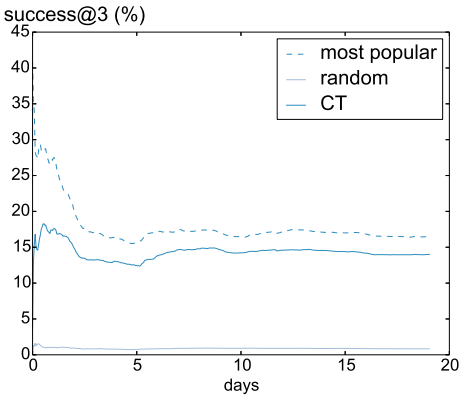
Question 2. *Do recommendations in general, and recommendations made by context-tree systems in particular, increase the page views by the user on the news website?*

Question 3. *Among all CT-based recommender systems, which version and set of parameters are optimal?*

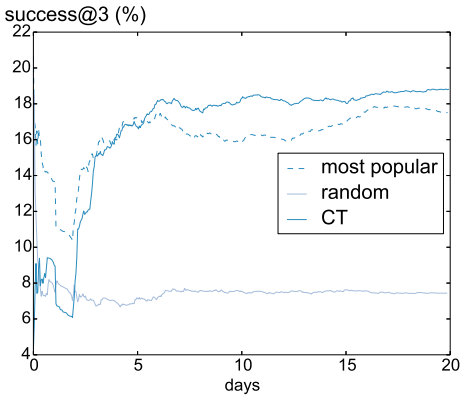
The answers to these questions are not trivial, and are specific to each website. Similar to our study, the results and conclusion of the evaluation on Forbes.com [57] are unique to this website. In addition, the reasons behind each study are not the same.

The online evaluation is split in two phases where we test different strategies against each other. The first phase consists of comparing a standard context-tree system against the baselines, and aims at answering Questions 1 and 2. The second phase investigates Question 3.

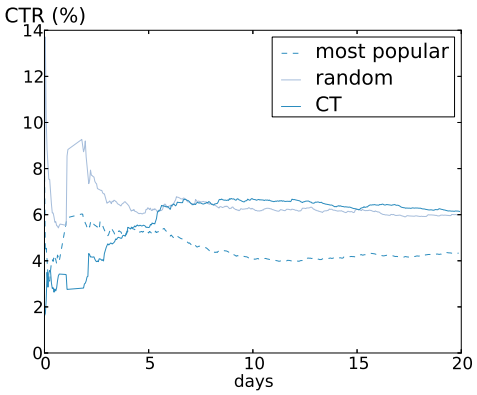
For each evaluation phase, we bootstrapped the recommender systems during one week, and evaluated them on the next two weeks. We estimated this time frame based on the previous offline evaluation, the expected traffic volume, and the time required to learn the model. It is crucial to run as many methods as possible in parallel to avoid biases in the evaluation due to



(a) Offline *predicted* accuracy



(b) Online *actual* accuracy



(c) Online *actual* CTR

Figure 3.12: CTR of online (bootstrap + phase 1) and offline evaluations.

trends and variations in the candidate set. However, we had to split the evaluation because we could not run more than 4 recommender systems in parallel in order to get statistical significant results.

The PEN recsys framework assigned randomly one recommender system to each visit, in the matter described in Section 3.6, performing a multivariate evaluation. Note that visits are anonymous and we were not able to track a user across multiple visits. For all recommender systems, the candidate set was composed of the last 200 clicked items and 50 fresh items. We chose these conservative parameters based on our previous offline analysis. When a recommender system receives one click, it returns 3 recommendations.

As opposed to the offline setting, we report for the online evaluation the click-through rate (CTR) on the recommendations, i.e. the number of clicks on recommendations over the number of impressions. Note that in the live evaluation, the CTR is computed with the actual *clicks* of the users on the recommendations, and is not a prediction.

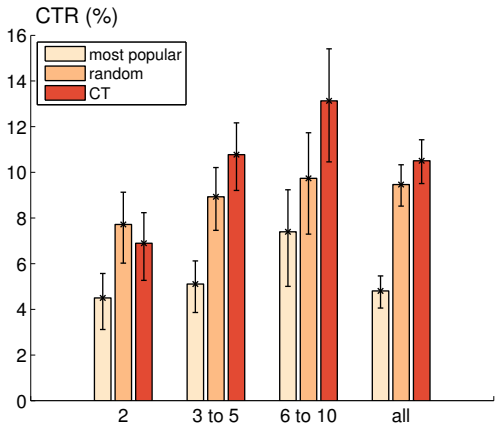


Figure 3.13: Online *actual* CTR over visit length, with confidence intervals at 95%.

Phase 1

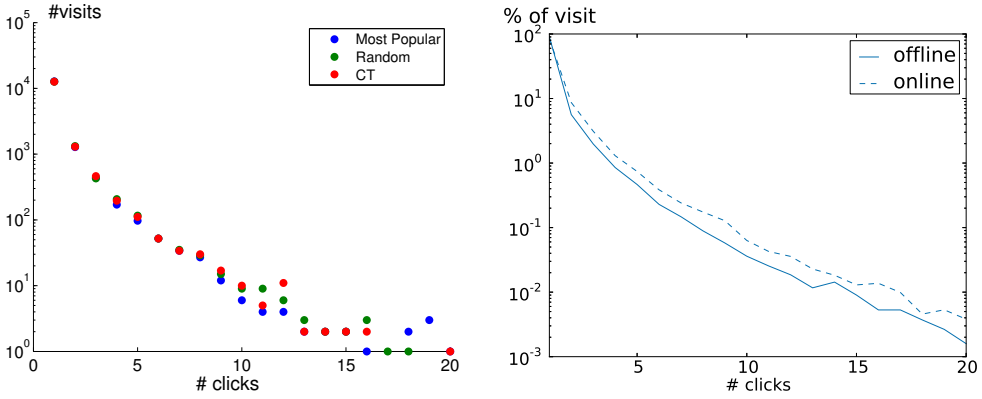
For the first phase, we selected the context-tree recommender system with the standard model. For the baselines, we implemented the strategies that recommend the most popular stories and that recommend stories at random. The overall click-through rate of recommendations was 8.4%. Figure 3.12(c) illustrates the bootstrap phase required by the context-tree system. It takes about one week to reach a comparable CTR to the one of random. Figure 3.13 shows the click-through rate per recommender system and over the length of the visit. Context-tree recommenders slightly outperform the baseline methods in general. When the visits are very short (exactly 2 articles), recommending random stories hardly increases the click-through rate (7.7%). However, for medium and long visits, the context-tree system greatly improves the CTR. For medium-length visits, the CT system has a 10.8% CTR while the random baseline is at 8.9%. For long visits, CT system is at 13.1% while the random baseline is at 9.7%. The strategy of recommending the most popular articles is definitely not the best solution: regardless of the visit length, the click-through rate stays below 7%. This is because users will have already read these stories elsewhere.

When the visit length increases, CT recommenders outperform baseline methods because they take advantage of the personalized sequence of each user to create an accurate profile. No matter what the user's interest is, the baseline strategies will always perform the same. Overall, context-tree recommender system brings a 11% improvement over recommending random items, and more than a 20% improvement when considering medium to long visits (21% and 35% for medium and long visits respectively).

To address Question 2, we measure the impact on the visit length by looking at the difference in the number of articles clicked. We break the visits into two groups: visits containing at least one click on a recommendation, and visits without any click on recommendations. Table 3.2 summarizes our findings. The length of the visits without taking a recommendation averages 1.25 clicks, while visits with at least one click on a recommendations jumped by more than 2.5

Table 3.2: Average visit length with and without recommendations

Recommender system	w/o recommendations	w/ recommendations
Most Popular	1.25	3.36
Random	1.25	3.93
CT	1.25	**3.96**



(a) Online only.

(b) Online and Offline (in percent of the total number of visits).

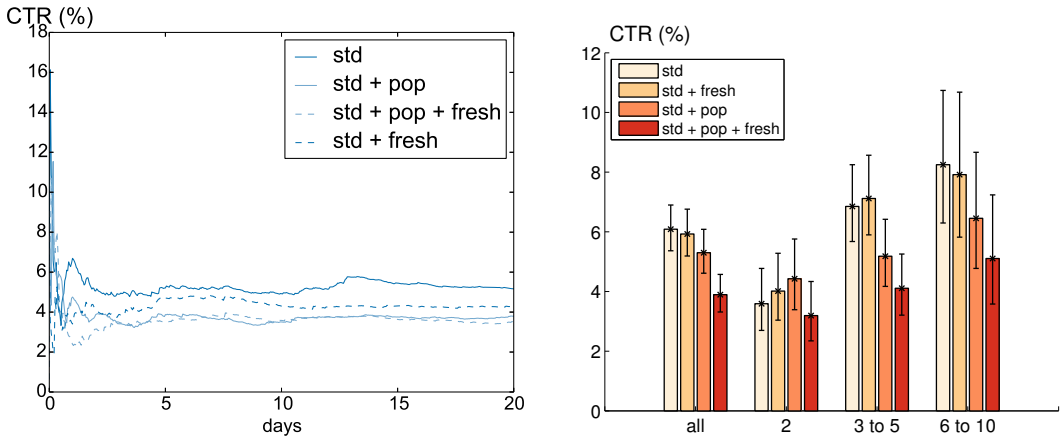
Figure 3.14: Distribution of visit length.

clicks, with an overall average at 3.75 clicks.

As depicted in Figure 3.14, most sessions are very short. Users reach the website through a search engine or a news aggregator, read one article, and leave. If we remove such behaviour by considering visits of length at least 3, the visit length increases by about 1 click, from 4.31 to 5.22 clicks. 90.4% of the visits have only one click, while with recommendations it drops to 85.0% of the visits. It is possible that users who click on recommendations are more likely to have longer visits. Since we are not tracking the users, but only their current session, it is difficult to answer this question. In Forbes.com for instance, this tendency exists [57].

Phase 2

For the second phase, where we compare different versions of CT recommenders (Question 3), the overall click-through rate of recommendations was slightly lower than during the first phase at 5.0%. Figure 3.15 illustrates the click-through rate with different mixtures of models. Again, we need a bootstrap phase of roughly one week until the model is correctly learnt. Incorporating the popularity model decreases the click-through rate, except when the visit is very short. This is not surprising since users with a very short visit are mostly interested in what is displayed on the main page of the website, thus reinforcing popular stories. However, when the length increases, users are no longer interested in popular stories. It is not clear whether the fresh model improves recommendations. However, this might be due to the fact



(a) bootstrap + phase 2

(b) Phase 2 only, with confidence intervals at 95%.

Figure 3.15: Online *actual* CTR of different mixtures of experts for phase 2.

that the number of fresh items considered here (30) is not optimal. We believe that tuning this parameter will improve the accuracy of recommenders with the fresh model.

3.8 Discussion

We discuss here important factors that influence directly the recommendations, and report on our experience in developing and deploying the PEN recsys framework on swissinfo.ch website.

3.8.1 Online vs Offline Evaluations

We used the offline evaluation to select the recommender systems and their parameters that could most likely generate good recommendations in a live environment. However, the difference between online and offline evaluations is striking (Figures 3.12(c) and 3.12(a)). In the offline setting, it is difficult to do better than recommending most popular articles because the recommendations do not directly influence the users. This method mimics the best the behaviour of the readers when no recommender systems are in place because items on the front page attract the most clicks. In the live evaluation, the popularity-based strategy is clearly not the most interesting because users do not want to read articles they have already seen on the front page.

Some recommender systems could have been ruled out of the live trial due to their poor performance during the offline evaluation. However, the conclusion would not have been the same in a live setting. In an offline setting, the *predicted* accuracy alone is thus not enough to glimpse the possible live performance. We believe that when evaluating recommender systems in an artificial environment, researchers should study more than one metric. In

particular, diversity and novelty are important factors that influence recommendations in a live setting.

Figure 3.12(b) illustrates the *actual* accuracy in an *online* setting. In this figure, the accuracy incorporates clicks on recommendations but also on predicted articles. The *actual* CTR in Figure 3.12(c) is approximately equal to the difference between the online *actual* and offline *predicted* accuracies (Fig. 3.12(c) = Fig. 3.12(b) - Fig. 3.12(a)). Indeed, the online *actual* accuracy incorporates both offline and online performances. However, this observation does not hold for the popularity-based strategy. The reason behind might be that there is a major shift in reading behaviour between the offline dataset and the live website.

We conjecture that a random strategy works better for short visits because users want to have diverse items. When the visit length increases, users might be less interested in diversity, and tend to read more about the same topic [38].

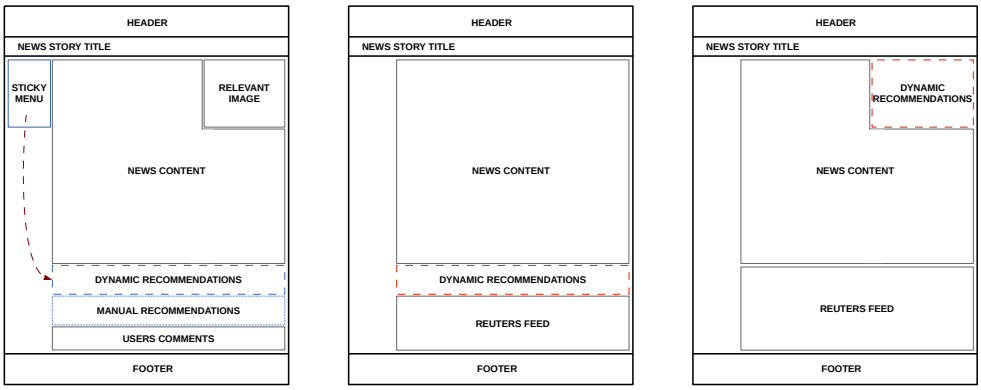
Although there exist many metrics [42, 94] to assess the performance of a recommender system, the click-through rate is a de-facto standard in the industry because it is often correlated to the revenues generated by the news website. Indeed, these revenues come from either advertisements (ads) displayed on the website or paid articles (or sometimes both). In general for online advertisement, there are two revenue models: pay per impression or pay per click. With the former, the news website receives monetary compensation for displaying ads while with the latter every time a user clicks on the ad. In all cases, news websites are incentivized in increasing page views. So one way to justify the performance of a recommender system is through its generated revenue, or click-through rate.

The difference in performance between online and offline for the popularity-based method is explained by the fact that offline dataset does not capture user preferences. Instead, offline datasets have a very strong bias towards popular items. These items are displayed on the front page of the website, and attracts most of the clicks. Thus, it would be interesting to modify the recommender algorithms to take into account aspects of the page design [63].

3.8.2 Page Layout

A recommender system is a small piece of a bigger and complex environment. In our case, we have little to no control over the other elements of the environment such as the presentation layer.

swissinfo.ch's website has two types of news items: *stories* and *tickers*. The former are in-depth news articles written by swissinfo.ch's journalists. The latter are unedited news items coming from press agencies. Of course, swissinfo.ch wants to recommend their content instead of third-party items. Unfortunately, it was not possible to have different layouts for stories because stories are of highest importance than tickers for swissinfo.ch (they receive 7 times more clicks) and it would have decreased the content quality of the page. However, we were able to test different placements on ticker pages. Note that the generated recommendations



(a) story (b) ticker (bottom) (c) ticker (top right)

Figure 3.16: swissinfo.ch's page layout for a story and a ticker. For a story, dynamic (blue-dashed) and manual (blue-dotted) recommendations are at the bottom of the page. For a ticker, the position of dynamic recommendations (red-dashed) changes: bottom or top right.

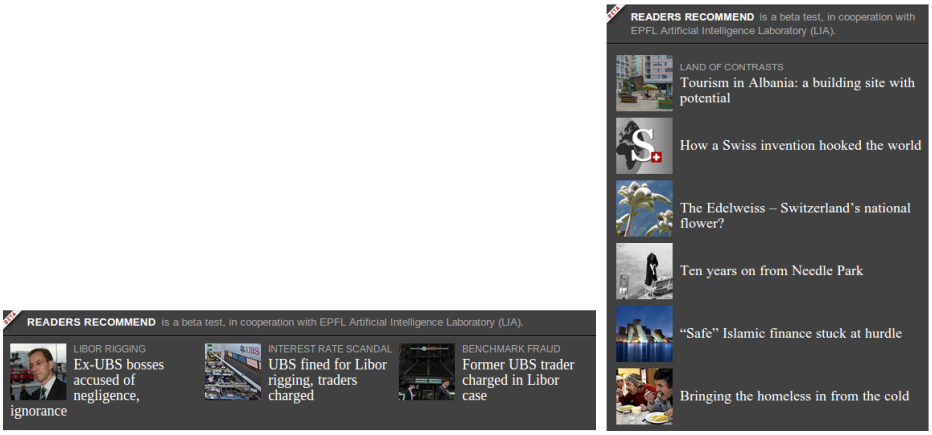
are always composed of stories.

For a *story*, the web page is split into several areas, sketched in Figure 3.16(a). The header has the menu buttons to browse the website and jump to different sections and topics, while the footer contains information about swissinfo.ch, links to social media sites, links to swissinfo.ch mobile applications and legal notices. Below the header, the title of the news story is displayed, then follows the content with one relevant image.

The layout of the page is extremely important [82, 13]. Most of the news stories are very long and do not fit on the displayed area of the screen (see Figure 3.17). Thus, the placement of recommendations on the page plays an important role in drawing users' attention. Although there is a direct link in the sticky menu pointing to the recommendations, users need to scroll down the page to see the recommendation box, and we believe that users tend not to read articles down to the end.

For a *ticker*, the web page is similar to the one of a story (Figure 3.16). Nonetheless, we have the opportunity to alter the page in such a way we can have two different placements of the dynamic recommendation box: top right and bottom (see Figure 3.18). The top-right version allows us to display more recommendations (6 stories) than the bottom version (3 stories). Figure 3.19 illustrates the difference in click-through rate between the two placements. When the recommendations are at the top-right corner of the page, the CTR is more than double (2.25) the one at the bottom. Hence, displaying recommendations at the top-right is significantly (χ^2-test with $p < 0.01$) better than at the bottom. The placement might not be the only reason of this improvement. The width of the page limits the bottom recommendation box to only three items, while the top-right version contains 6 recommendations. So the larger choice available to the user might also play a role. An easy way to verify this claim would

Figure 3.17: A swissinfo.ch's story of average length (2758 characters)



(a) bottom (stories and tickers) (b) top right (only tickers)

Figure 3.18: swissinfo.ch's dynamic recommendation boxes: bottom and top-right position. Note that the number of recommendations is not the same.

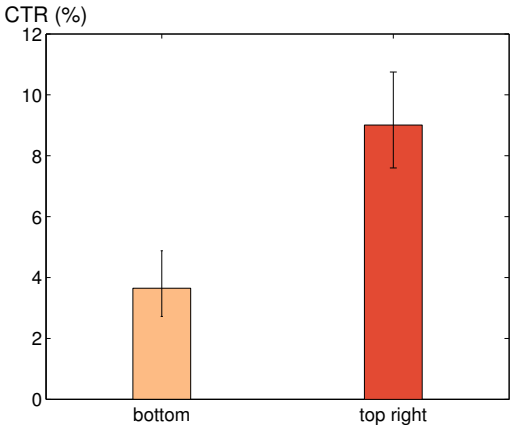


Figure 3.19: Click-through rate on tickers for the two different placements, with confidence intervals at 95%.

be to place the bottom version at the top of the page, but this solution has been rejected by swissinfo.ch.

Recommendations generated by the PEN recsys framework are displayed in the *dynamic* recommendations box. The original design of the page limits to three recommended items due to aesthetic constraints (see Figure 3.18(a)). We believe that three recommendations restrain the user's choice. We do not know what is the optimal number of recommendations, but would be an interesting future direction.

The author of a story can select related stories and place them in the *manual* recommendations box situated below the dynamic recommendations. We discuss the differences between manual and dynamic recommendations in the next Section.

3.8.3 Manual and Dynamic Recommendations

Manual recommendations are constructed by the author of the current news item. They take into account only the current article, but not the history and preferences of the reader. It is desirable to have an automatic way to create more personalized recommendations.

When generating dynamic recommendations, the system can take into account the set of manual recommendations in order to refine its recommendations. We believe that manual recommendations can be replaced by a simple content-based recommender or a content retrieval system that do not rely on user's preferences. However, dynamic recommendations are more challenging to design and implement.

Manual recommendations might attract more users than personalized recommendations. Researchers should take into consideration the fact that one might drives more click than the other. However, it does not mean that the recommendations are bad, but that the interest of the users are different. To study this issue, it would be interesting to look at the click ratio of manual versus the click ratio of dynamic recommendations.

3.9 Conclusion and Future Work

News recommendation is challenging due to the rapid evolution of topics and preferences. We introduced a class of recommender systems based on context trees that accommodate a dynamically changing model. We considered context trees in the space of sequences of news, sequences of topics, and in the space of topic distributions. We defined expert models which consider the popularity and freshness of news items, and examined ways to combine them into a single model. We proposed an incremental algorithm that adapts the models continuously, and is thus better suited to such a dynamic domain as the context tree evolves over time and adapts itself to current trends and reader preferences. Our approach requires only one tree (the context tree), and thus scales very well. Our work is not restricted to the history of logged-in users, but considers a one-time session for recommendation, where users do not log in. Surprisingly, we do not know of any existing research that considers context-tree models for recommender systems.

Each proposed variant has its strengths and weaknesses. To evaluate them in an offline setting, we used the expected performance curve methodology, whereby each method is tuned in a training set according to a parametrized utility metric. In doing so, we showed that if we are interested in accuracy in a static dataset, a context tree that implements a variable-order Markov model is ideal, while novelty is best served with a k-d tree on the space of topics. In addition, we showed that a large order is mainly important when we are not interested in recommending highly popular items.

The online evaluation on the newspaper website swissinfo.ch does not show the same behaviour than in an offline setting. In particular, recommending most popular items is definitely

not the best strategy. In an online setting, context-tree recommender systems improve by up to 35% the click-through rate over the best baseline. The visit length more than double when readers click on recommendations. As expected, the placement of recommendations is crucial: the click-through rate more than double when recommendations are displayed at the top-right corner of the article.

At the time of writing, the evaluation is still ongoing. For instance, we are missing results about content-based methods. An interesting future direction would be to extend the set of expert models to take into account social and demographic data.

4 Predicting Outcomes of Events

4.1 Introduction

The outcome of many important events depends on detailed information that is only known to certain individuals. For example, the outcome of a vote depends on the sympathies of voters for different options, the success of a project depends on a combination of details, and the sales of a new product are determined by how much an average consumer likes it.

Such questions are typically answered by polling a significant number of people who each provide a different judgement based on their perception of these details. Such polls provide the best results when they are carried out on an unbiased sample. However, this requires that every member of the sample makes the effort to answer the questions, which is not easy to enforce. Thus, most online polls are based on voluntary participation or even self-selection, where people respond to a poll out of their own initiative. Responses are often given for ulterior motives, resulting in biased and questionable results. For example, in product review websites most reviews have either a positive or negative bias [45, 54], so that it is not clear whether the average rating actually reflects the true quality as we have discussed in Chapter 2.

One way to encourage participation by a broader sample of the population is to reward participants for their response. However, this raises a question of quality control: if random answers carry the same rewards as honest answers, why would anyone make an effort to give a correct answer?

This question has been addressed extensively in AI research but has not been applied very much in practice. In this chapter, we report on an experimental platform, *swissnoise*[1] (Figure 4.1), for conducting opinion polls on questions of public interest. Swissnoise experiments with two different models: prediction market [40, 41, 15] and peer prediction [51, 103, 104, 83]. To our knowledge, it is the first platform to implement a peer prediction scheme in a public opinion poll. Peer prediction is a new scheme that can be applied more broadly than prediction markets. The goal of our platform is to show that it can be practically implemented and

[1] http://go.swissnoise.ch

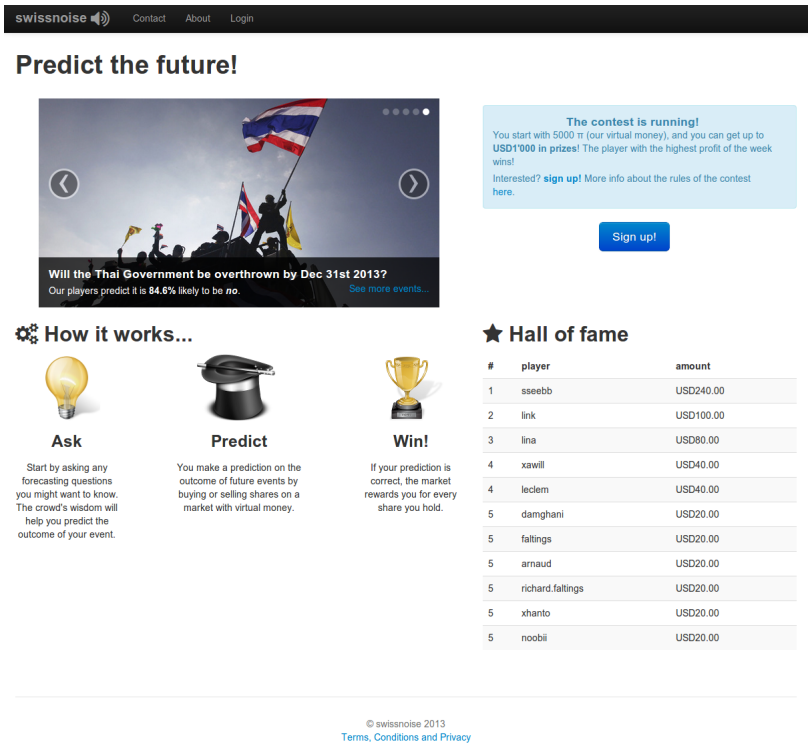


Figure 4.1: swissnoise's homepage.

achieve performance that is comparable to prediction markets.

4.1.1 Contributions

In this chapter, we study the practical implementations of two aggregation mechanisms to elicit private information. First, we show that the peer prediction scheme can be practically implemented in an online public opinion poll. Second, we discuss the design choices and variations made to such mechanisms when implementing them. Finally, we show that peer prediction achieves a comparable performance to prediction markets.

4.1.2 Related Work

The simplest mechanism to measure the accuracy of predictions is a scoring rule [93]. The score (or payment) rewards participants based on their predictions such that it incentivizes truthful reporting. Hanson et al. [40, 41] build on these scoring rules and proposed a family of rules known as Market Scoring Rules (MSR). MSR and in particular the logarithmic MSR are the most commonly used prediction market algorithm. As a results, they gave birth to a very rich literature on prediction markets [15, 100].

There are a lot of commercial online prediction market platforms. Among the non-profit

platforms, the Iowa Electronic Markets[2] is probably the first and most famous. More recently, Scicast[3] is another non-profit platform targeted to prediction of scientific topics. Tziralis et al. [100] provide an extensive survey of the literature on prediction markets until 2006. The literature on this subject continued to grow. Recently, Dudik et al. [27] implemented a combinatorial version of prediction market for the 2012 US election.

Miller et al. [72] proposed the original peer prediction mechanism, but their scheme relies on strong assumptions about the common knowledge of the participants. A stream of researches [80, 104, 83] improved this version by relaxing some assumptions. Prelec [80] introduced the Bayesian Truth Serum (BTS) which assumes that participants share a common prior, but the scheme does not need to know it. However, participants need to make two reports: one about their own information and one on the prediction of the overall population. Witkowski et al. [104] defined a robust version of the BTS by rewarding participants based on how well their own information report is used to update the overall prediction report. Radanovic et al. [83] improved the BTS such that the information and prediction scores are mutually independent. This results also apply to the robust BTS.

4.2 Prediction Mechanisms

The goal of a prediction mechanism is to aggregate information in order to predict outcomes of future events. We introduce here two types of mechanisms: *prediction market* and *peer prediction*.

4.2.1 Prediction Market

The idea behind prediction markets [40, 41, 15] is to encourage self-selection of individuals who are the most knowledgeable about a topic, and pay more for information that makes the outcome more accurate. Participants answer a poll with respect to the already known information. They trade securities linked to each possible outcome. When one outcome materializes, the securities for that outcome pay a reward of 1, whereas those that did not realized pay nothing. Thus, a participant can expect to gain by

- buying securities at a price that is below the probability of the associated outcome, and
- selling securities at a price that is above this probability.

If all participants evaluate the outcomes in the same way, a prediction market is thus in equilibrium whenever the price of the securities is equal to the predicted probability of the outcome.

[2] http://tippie.uiowa.edu/iem
[3] www.scicast.org

Another function provided by a prediction market is that of aggregating information. When participants do not agree on a single probability - and usually they will not - the aggregation is determined by how much money they are willing to risk on their prediction: as buying and selling securities changes the price, a participant may need to buy a large number of shares to move the price to her believed probability. If other participants have a strong opposite belief, they will readily sell their shares so that the price moves only very slowly. In practice, there are often not enough simultaneous participants, and thus this liquidity is simulated by an automated market maker. An automated market maker is based on a scoring rule and adjusts the price of securities so that the expected reward for changing the probability of an outcome is proportional to the difference of what a logarithmic scoring rule (Equation 4.1) would pay for the new probability and for the old probability. The scaling factor b that determines the actual amount is called the *liquidity parameter* and an important element of the design of the prediction market.

A major issue with practical deployment of prediction markets on public platforms is that at least when real money is used, many countries consider them a form of online gambling that is illegal. This is because participants have to place bets on particular outcomes that may or may not pay off. This could be overcome by just using scoring rules, so that payoff occurs only at the end, but this would mean that rewards are only paid very much later and make the market less interesting.

Definition

The simplest prediction mechanism that rewards participants in return to their accurate information is a *proper scoring rule* [93]. More formally, let X be a discrete random variable to be predicted with N mutually exclusive outcomes. A participant submits a probability estimate $\mathbf{r} = \langle r_1, r_2, ..., r_N \rangle$ of the value of X.

A scoring rule $S = \{s_1(\mathbf{r}), s_2(\mathbf{r}), ..., s_N(\mathbf{r})\}$ assigns a score $s_i(\mathbf{r})$ to the participant who reports $\mathbf{r}$ if outcome i is realized. A proper scoring rule enforces truthful reporting for risk-neutral participant. A widely used proper scoring rule is the logarithmic scoring rule:

$$s_i(\mathbf{r}) = a + b\log(r_i) \tag{4.1}$$

where a and b are constant with $b > 0$.

Hanson [40, 41] proposes an automated market maker mechanism based on this scoring rule, called the market scoring rule. This mechanism can be seen as a sequential and shared version of the proper scoring rule. The market maker starts the market with some initial probability estimate $\mathbf{r}^0$ over the outcomes. Every participant can change the current estimate $\mathbf{r}$ to a new estimate $\mathbf{r}'$ as long as she pays $s_i(\mathbf{r})$ and receives $s_i(\mathbf{r}')$.

We present here an equivalent formulation of the market maker based on a cost function [14]. Let q_i be the total quantity of security i held by all participants, and $\mathbf{q} = \langle q_1, q_2, ..., q_N \rangle$ be the

vector of all quantities. The cost function $C(\mathbf{q})$ defines the total amount of money spent as a function of the total number of shares held of each security. A participant who buys or sells any security will change the total number of securities from $\mathbf{q}$ to $\mathbf{q}'$. Thus, the market maker charges her $C(\mathbf{q}') - C(\mathbf{q})$. Note that a negative value represents a sell order, or a payment from the market maker to the participant.

The cost function $C(\mathbf{q})$ for the market maker with the logarithmic scoring rule (Equation 4.1) is

$$C(\mathbf{q}) = b\log(\sum_{i=1}^{N} e^{q_i/b}) \tag{4.2}$$

where b is the liquidity parameter of the market. The larger the liquidity is, the more shares a participant can buy without affecting too much the price. If the liquidity is too small, the price swings dramatically after a small number of shares is bought.

The corresponding instantaneous price function is

$$p_i(\mathbf{q}) = \frac{\partial C}{\partial q_i} = \frac{e^{q_i/b}}{\sum_{j=1}^{N} e^{q_j/b}} \tag{4.3}$$

Note that this price applies for buying a infinitesimal number of shares. As soon as the participant starts buying, the price immediately increases.

4.2.2 Peer Prediction

Besides the legal issues, another problem that is common to both scoring rules and prediction markets is that they can only be applied when the predicted outcome can be verified with certainty. This makes it impossible to collect predictions for outcomes that will never be verified, such as product quality or appeal.

Peer prediction [72] solves this issue. The idea is to consider the reports of other agents that observed the same variable, or at least a *stochastically relevant* variable, as the missing ground truth. A proper scoring rule is then used for the incentives. Provided that other agents truthfully report an unbiased observation of the variable, such a reward scheme makes it a best response to provide truthful and unbiased reports of the observations, and truthful reporting thus becomes a Nash equilibrium. Miller et al. [72] describe such a mechanism and several variants, and Jurca and Faltings [52] discuss further optimizations and variants.

An important limitation of peer prediction methods based on proper scoring rules is the need to know the agents' posterior probability distributions after each measurement. Zohar and Rosenschein [109] investigate mechanisms that are robust to variations of these distributions, and show that this is only possible in very limited ways and leads to large increases in payments.

The Bayesian Truth Serum [80, 104, 83] is a mechanism that elicits both the estimate itself as well as the beliefs about other's estimates. This elicitation of extra information eliminates the need to know the prior beliefs, but also requires participants to provide more information than just the answer to the question, which makes their task cognitively more difficult and too complex to implement.

We therefore took inspiration from prediction markets to implement a peer prediction scheme that assumes a common prior probability given by the current poll result. Similar to a prediction market, we display a current probability for each outcome. This probability is obtained by Bayesian updating from the reports received from different participants so far. Periodically, new reports are integrated into the current prediction to make it more accurate.

Definition

A reward is paid whenever the report matches a *peer report* that is randomly chosen among the reports that have been received in the same time period between updates of the public distribution. The amount of the reward is scaled so that it is inversely proportional to the currently estimated probability of this outcome. More formally, let $\mathbf{r}^t = \langle r_1, r_2, ..., r_N \rangle$ be the *public* prior probability distribution over N outcomes at time t. The reward function for reporting outcome x, given the reference report y is

$$\tau(\mathbf{r}, x, y) = \begin{cases} \min(\theta, a + \frac{b}{r_x}) & \text{if } x = y, \\ a & \text{if } x \neq y \end{cases} \tag{4.4}$$

where a and $b > 0$ are constants to scale the reward, and θ is an upper bound on the reward. We call this reward scheme the *peer truth serum*: as the Bayesian Truth Serum, it does not require knowledge of prior probabilities, but rather than requiring extra reports from participants, it takes this prior probability from the poll itself.

This reward scheme will reward accurate reports whenever the participant a) believes that the current probabilities reflect the true prior distribution of other participants' reports, and b) believes that the true distribution of other participants' answers is actually shifted in the direction of her own opinion so that:

$$\frac{Pr_x(x)}{Pr(x)} > \frac{Pr_x(y)}{Pr(y)} \tag{4.5}$$

where $Pr(x)$ is the prior probability of outcome x while $Pr_x(x)$ and $Pr_x(y)$ are the posterior probabilities for the outcomes x respectively y, when the agent believes the true outcome should be x. This condition is satisfied for example if the participant performs Bayesian updating to combine its own belief with the current poll outcome. This can be shown easily by considering that the probability of outcome x matching that of another randomly chosen participant is equal to $Pr_x(x)$, and the reward is equal to $r_x = Pr(x)$ - the condition is then equal to the incentive-compatibility condition.

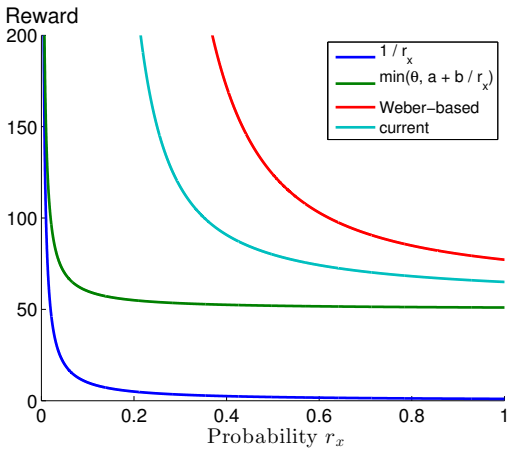


Figure 4.2: Different reward schemes for peer prediction ($a = 50, b = 1$).

A major issue with the original scheme ($1/r_x$) proposed by Jurca et al. [53] and the generalized version (Equation 4.4) is that the difference of rewards between one outcome and another is so small that participants are indifferent (Figure 4.2). As a result, the participants opt for the less risky outcome. This risk-averse behaviour is naturally explained among individuals: in a noisy environment, we need to shout to be heard, while a whisper is enough in a quiet room. Actually, the perception of sound is proportional to $\log_{10}$. In psychophysics, this effect is known as Weber's law.

To compensate this behaviour, we modify the reward scheme by taking into account Weber's law:

$$\tau(\mathbf{r}, x, y) = \begin{cases} \min(\theta, a + e^{\frac{b}{r_x}}) & \text{if } x = y, \\ a & \text{if } x \neq y \end{cases} \tag{4.6}$$

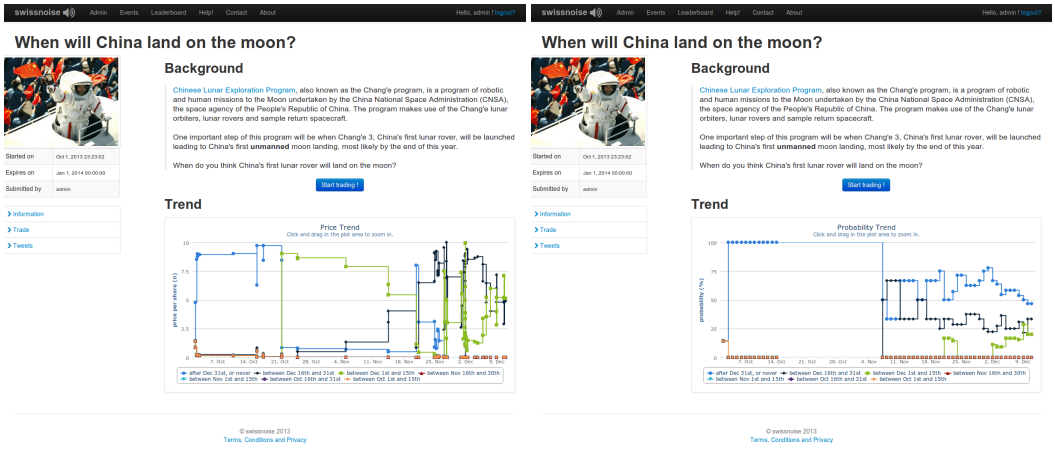
Unfortunately, this modification created risk-seeking behaviours because when $r_x < 0.4$ the reward is maximum. In order to decrease such behaviours, we average Equation 4.4 and Equation 4.6 such that

$$\tau(\mathbf{r}, x, y) = \begin{cases} \min(\theta, a + \frac{\frac{b}{r_x} + e^{\frac{b}{r_x}}}{2}) & \text{if } x = y, \\ a & \text{if } x \neq y \end{cases} \tag{4.7}$$

4.3 Swissnoise

We designed a platform called *swissnoise*[4] with the goal of predicting results of Swiss ballots. However, swissnoise contains now more diverse events ranging from sports, entertainments to politics. Swissnoise was open to the public on April 22nd 2013. As of Jan 27th 2014, the

[4] http://go.swissnoise.ch



(a) Prediction market.

(b) Peer prediction.

Figure 4.3: swissnoise's event description panels.

platform had 200+ active users with a total of 132 events (20 are currently open). The platform is free to signup and use. Each user starts with π5000. π is our virtual currency on the platform. Every week, we assign one gift card of USD20 to the user who achieves the highest profit during that week.

Swissnoise implements two mechanisms to elicit information from the crowd: prediction markets and peer prediction. Since we are interested in comparing these two schemes, for a given event each user is assigned randomly to one of them.

4.3.1 Implementation and Design

Prediction Markets

We implemented the logarithmic market scoring rule [40, 41]. We determined the liquidity parameter $b = 100$ empirically in such a way that it allows newcomers to still be able to influence markets although their starting amount is lower than advanced users. We also scaled up the payments (by 10) to make it more attractive to the users.

Due to the way we rewarded the users, strategic behaviours emerged. Some users clear their trading positions at the last moment, on Sunday night, right before we determine the weekly winner in order to cash their profit. Figure 4.3(a) illustrates these dramatic price swings for one event.

Peer Prediction

Since swissnoise is the first platform to implement a peer prediction scheme for public opinion polls, it was not clear a) how to implement the peer truth serum, and b) what are the best

design choices.

In swissnoise, the peer truth serum is implemented as a "lottery". This analogy has the advantage of being well-understood among the majority of the users. However, we had to adapt it slightly to match the peer prediction scheme.

The key idea behind our implementation of the peer truth serum is that the user controls an agent which plays for her. Every day at midnight, we collect the statistics of the current day about an event and run a lottery. We randomly match a user's lottery ticket (report) to another user's ticket (peer report), and if their opinions are the same, the user is rewarded according to Equation 4.7.

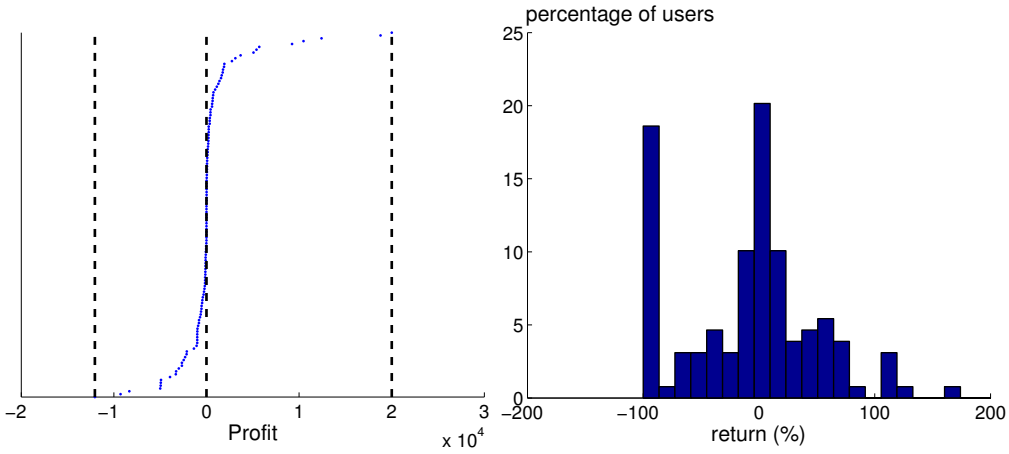
For a given event, the user selects first the number of days the agent is going to play, and buys lottery tickets accordingly. One lottery ticket is used per day. Then, the user selects the outcome she thinks is the best. She can update her choice at any time, but only the most updated information is taken into account for matching the tickets.

In the initial phase of the implementation, the potential reward of the lottery was very low (Equation 4.4), and it was making the peer prediction not interesting compared to the prediction market (Figure 4.3(b)). So we had to scale up the reward in such a way that the probability of winning the lottery would bring about the same reward as the average reward on a prediction market event. The popularity of peer prediction events increased, but a risk-seeking behaviour emerged among our users. Users started to choose the less likely outcome in order to get the highest reward (Equation 4.6). This behaviour resulted in daily oscillation around the 0.5 probability of the outcome. We observed this behaviour only for events with binary outcomes. Indeed, collusion/synchronization of behaviours among users on events with more than 2 outcomes is difficult because it requires to observe more than one signal.

To tackle this issue, we adjusted the reward for the peer prediction (Equation 4.7) and, instead of taking the current statistics, we computed a running statistics over 5 days. As a result, it decreased the oscillation effect.

Another issue was that some users did not update their votes, and the peer truth serum contained stale opinions. Thus, we decided to limit the number of possible tickets that a user can buy to 5. A user needs first to buy 5 tickets, and comes back after 5 days to buy again more tickets, and at the same time, update her opinion if necessary.

The additional π earned on lottery events can be used to buy shares in prediction market events or tickets in peer prediction events. The profit made during one week with the peer prediction and prediction market determines whether she is the winner of the week or not.



(a) Profit of each user displayed by ascending order. Dashed lines are min, median and max profit, respectively.

(b) Distribution of return across the user population.

Figure 4.4: Profits and returns.

4.4 Results

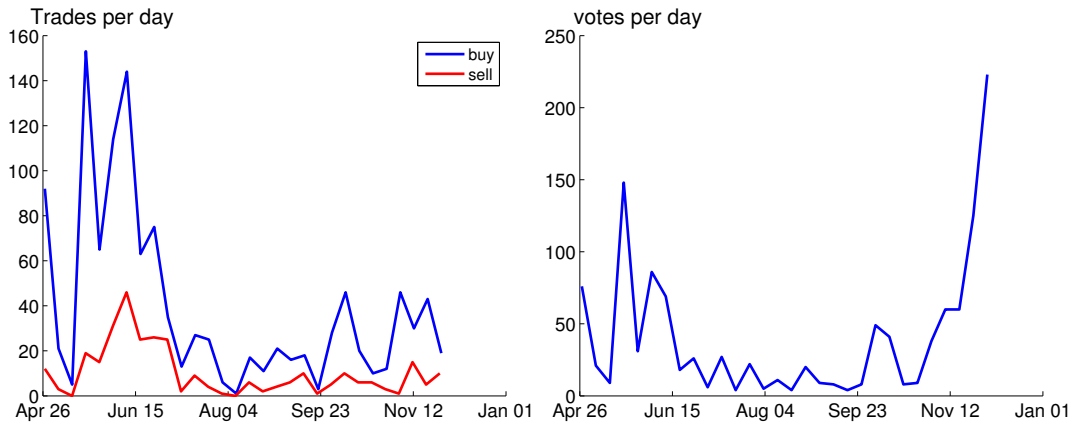
The performance of the users on the prediction market is depicted in Figure 4.4. The revenue is defined as the total amount received from selling shares and from payouts when a market is closed. The spending is the total amount spent in buying shares of events. The profit of a user is her revenue minus her spending, and her return is her profit over her spending. The profit and return indicates how good a user is on the platform. Note that the return has a lower bound at -100%, but the profit does not have a lower bound because users can get extra π by the means of lottery. For Figure 4.4(a), the minimum profit is -12051, the maximum profit is at 20000. The median profit is slightly positive at 1.47 which shows that the median user improved the market's forecast accuracy. Figure 4.4(b) illustrates the distribution of users' return. The first peak at -100% corresponds to users whose predictions did not happen. The second peak on the right hand side of 0% is for users who slightly improved the market accuracy. A third smaller peak exists at around 50% showing that a small portion of users has improved the prediction more than the median user. Finally, another peak lies after 100% which could corresponds to risk-seeking users whose risky predictions actually paid off. These curves are similar to the ones reported recently by Dudik et al. [27].

To illustrate that peer prediction achieves a forecast accuracy comparable to the one by the prediction market, we focus our study on three events about the Swiss ballots because we can also compare our results with the ones from a traditional opinion poll company, named gfs.bern[5].

On November 24th 2013, Swiss citizens voted on 3 items[6]: two items where popular initiatives

[5] www.gfsbern.ch

[6] More informations on the Swiss government portal https://www.ch.ch/en/federal-vote-of-the-24th-of-



(a) Number of trades on prediction market events.

(b) Number of votes on peer prediction events.

Figure 4.5: User activity on swissnoise.

proposed by some eligible persons or a group of persons, and one was a mandatory referendum proposed by the Swiss Federal Assembly. At ballot stage, voters vote *yes* or *no*. The item comes into force if it is accepted by a double majority: majority of votes and majority of Swiss states. The 3 items are

Initiative on fair wages. This initiative asks that the highest salary paid in a company should not exceed twelve times the amount of the lowest salary. The question asked was *Are you in favour of the initiative on fair wages?* This item has been rejected at 65.5%.

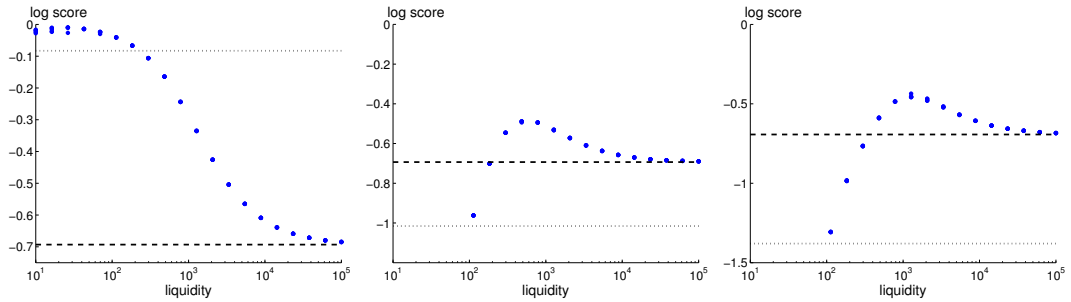
Initiative for families. This initiative asks that parents who look after their children could deduct the same or a higher amount from their taxes as parents who pay for childcare. The question asked was *Are you in favour of the initiative on tax deductions for families?* This item has been rejected at 58.5%.

Amendment on tax for highways. This amendment aims at increasing the charge for using the highways from CHF40 to CHF100 a year, and introduce a two-month highway tax sticker costing CHF40. The extra revenue will be used to finance the running, maintenance and expansion of around 400km of roads. The question asked was *Are you in favour of the amendment on tax for highways?* This item has been rejected at 60.5%.

We posted these three items on swissnoise and asked what would be the outcome of the final vote.

We analyse the choice of the liquidity parameter by running a counterfactual simulation and following the same protocol as described by Dudik et al. [27]: we transform buy/sell transac-

november-2013



(a) Initiative on fair wages. (b) Initiative for families. (c) Act on highway tax.

Figure 4.6: Forecast accuracy for different liquidity parameters. The dashed and dotted lines represent the log score for the initial probabilities and the actual market, respectively.

tions into a sequence of limit orders, and execute this sequence with different parameters. We then compute accuracy as the log score, i.e. the log of the probability of the realized outcome.

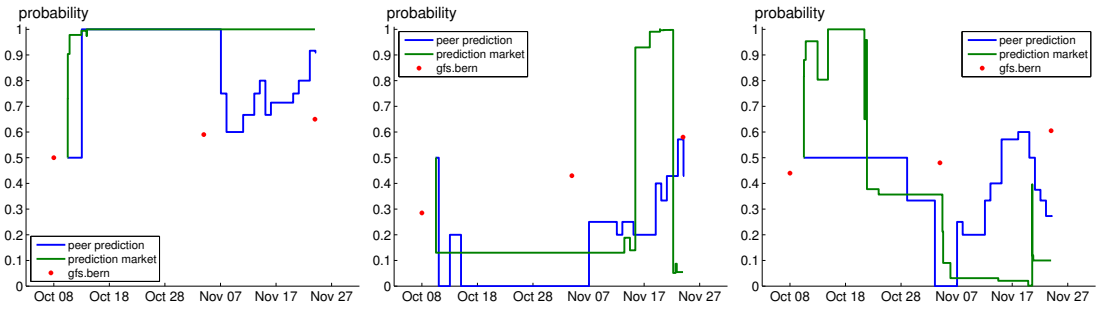
Figure 4.6 presents the forecast accuracy (average over 5 runs) for different liquidity parameters. Although the market accuracy for the first item (Figure 4.6(a)) was very close to the optimal performance, the optimal liquidity is around 25 which indicates a low activity. Actually, the users knew at an early stage that this item would be rejected and it is also reflected by gfs.bern's opinion poll (Figure 4.7(a)). Among the three items, the first item was the most certain to be rejected. On the other hand, for the last two items, our choice of liquidity parameter was suboptimal. The optimal liquidities were around 480 and 1250 for the second and third item, respectively. This high liquidity reflects a higher activity for these two items. The gfs.bern's opinion polls (Fig. 4.7(b) and 4.7(c)) show that these two items were very uncertain.

Other factors influencing the user activity on the platform are: a) the number of open events, b) the popularity of these events, and c) how close we are to determine the winner of the week. In addition, we see in Figure 4.5 a decrease in activity during the summer vacation.

The accuracy of both schemes are depicted in Fig. 4.7 and 4.8. For the first item (Fig. 4.7(a) and 4.8(a)), they both predicted correctly the outcome, while the opinion poll is more balanced. On Nov 7th, the forecast by peer prediction dropped to the same level as the opinion poll, and then increased until the end of the event, following the trend of the opinion poll. We believe that participants of the peer prediction scheme have been aware of this opinion poll and adapted their opinion, while users on the prediction market did not.

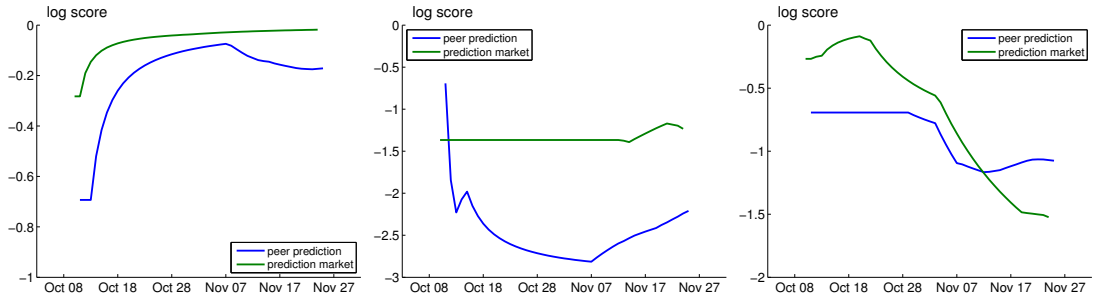
Regarding the second item, both schemes and the opinion poll predicted that the item would be accepted. The closer we get to the realization of the event, the better we get to the actual prediction. One day before the event, the peer prediction touched the opinion poll. The price swings on the prediction market might be due to the strategic behaviours reported in the previous section. The peer prediction is better for the third item.

Both mechanisms have similar behaviour and accuracy, but it is difficult to compare them to



(a) Initiative on fair wages (b) Initiative for families (c) Amendment on highway tax

Figure 4.7: Reject probability of the 3 items.



(a) Initiative on fair wages. (b) Initiative for families. (c) Amendment on highway tax.

Figure 4.8: Average log score of the 3 items.

traditional opinion polls for three reasons. First, the question asked to the users on swissnoise fundamentally differ from the one asked by the opinion poll company. Indeed, gfs.bern asked what *you* are going to vote, while on swissnoise we ask what you think the *outcome* is going to be.

Second, it is not clear how gfs.bern samples the population and handles selection bias. Swissnoise's users might not be representative of the Swiss population. However, most users are related to Switzerland and follow local media. So they have a feeling of what could be the outcome.

Third, although anonymous, traditional opinion polls face the fact that people might still lie and do not reveal what they are going to vote. With this in mind and considering Switzerland's strong privacy awareness, we designed swissnoise in such a way that it is not possible for a user to check the current or past opinions of other users. Thus, contrary to most implementations of prediction market platforms, swissnoise preserves the privacy of its users.

4.5 Conclusion and Future Work

Prediction markets have been applied with success for predicting events with a verifiable outcome. Recent research has developed the alternative technique of peer prediction which allows incentives without a verifiable final outcome. We have described how to adapt the peer prediction schemes developed in AI research to online opinion polls using the analogy of lotteries. This has been tested in the first experimental platform that implements peer prediction for online polls, called *swissnoise*. It shows that peer prediction has comparable performance to prediction markets, and thus constitutes a viable alternative. We are continuously collecting data about an increasing number of events.

An interesting direction for future work would be to study users' behaviour about hypothetical questions when rewarded with the peer prediction mechanism, and explore the possibility of an adaptive liquidity parameter for prediction markets.

5 Conclusion

In this thesis, we selected three applications to illustrate the possibilities offered by aggregating information from the crowd. The applications that we explored were the aggregation of explicit ratings for review websites, the aggregation of implicit feedback for personalized news recommendations, and the aggregation of opinions for predicting outcomes of events.

In the first application, we considered the aggregation of ratings from users of review websites. We proposed a new methodology based on a game to study users' rating behaviour. We have restricted our study to users who are well-intentioned and act so as to make the aggregate reflect the true value, either through altruism or through incentives based on the success with other users. An interesting future avenue for research would be to lift this restriction, and study users with different motives.

The second application focused on the aggregation of implicit feedback in order to generate personalized recommendations of news articles. We introduced a new class of recommender systems based on context trees that adapts the models continuously and is thus better suited to such a dynamic domain. We demonstrated that context-tree recommender systems generate accurate recommendations in an offline setting as well as live on the newspaper website *swissinfo.ch*. One obvious future direction would be to explore other domains than the news.

In the last application, we addressed the problem of eliciting private information to predict outcomes of events. We developed an experimental platform called *swissnoise* which implements prediction market and peer prediction mechanisms. We demonstrated that peer prediction achieves a performance comparable to that of prediction markets. During this evaluation, we only considered events with observable outcomes. An interesting future direction would be to study users' behaviour when rewarded on hypothetical events.

Selecting what to aggregate and how to aggregate it is crucial and changes from one application to another. Organizations should carefully analyse their needs and define their requirements prior to process any data. Once they are able to target the right informations and aggregation technique, they will create meaningful knowledge and possibly a competitive edge.

Bibliography

[1] J. Ahn, P. Brusilovsky, J. Grady, and D. He. Open user profiles for adaptive news systems: help or harm? In *International Conference on World Wide Web*, pages 11–20, 2007.

[2] Apache Software Foundation. Apache mahout. http://mahout.apache.org.

[3] D. Ariely, W. Tung Au, R. Bender, D. Budescu, C. Dietz, H. Gu, T. Wallsten, and G. Zauberman. The effects of averaging subjective probability estimates between and within judges. *Journal of Experimental Psychology: Applied*, 6(2):130–147, 2000.

[4] R. Begleiter, R. El-Yaniv, and G. Yona. On prediction using variable order markov models. *Journal of Artificial Intelligence Research*, pages 385–421, 2004.

[5] S. Bengio, J. Mariethoz, and K. M. The expected performance curve. In *International Conference on Machine Learning*, pages 9–16, 2005.

[6] J. Bentley. Multidimensional binary search trees used for associative searching. *Communication of the ACM*, 1975.

[7] D. Billsus and M. Pazzani. A hybrid user model for news story classification. In *Conference on User Modeling*, pages 99–108, 1999.

[8] D. Billsus and M. Pazzani. Adaptive news access. In *The adaptive web*, pages 550–570. Springer, 2007.

[9] D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[10] P. Bone. Word-of-mouth effects on short-term and long-term product judgments. *Journal of Business Research*, 32(3):213–223, 1995.

[11] R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12:331–370, November 2002.

[12] R. Burnkrant and A. Cousineau. Informational and normative social influence in buyer behavior. *Journal of Consumer research*, pages 206–215, 1975.

[13] L. Chen and P. Pu. Eye-tracking study of user behavior in recommender interfaces. In *User Modeling, Adaptation, and Personalization*, pages 375–380, 2010.

[14] Y. Chen and D. Pennock. A utility framework for bounded-loss market makers. In *Conference on Uncertainty in Artificial Intelligence*, 2007.

[15] Y. Chen and D. Pennock. Designing markets for prediction. *AI Magazine*, 31(4):42–52, 2010.

[16] M.-C. Chiu, S.-P. Chang, Y.-C. Chang, H.-H. Chu, C. C.-H. Chen, F.-H. Hsiao, and J.-C. Ko. Playful bottle: a mobile social persuasion system to motivate healthy water intake. In *International Conference on Ubiquitous Computing*, 2009.

[17] R. Clemen and R. Winkler. Combining probability distributions from experts in risk analysis. *Risk Analysis*, 19:187–203, 1999.

[18] J. Cohen and E. Golden. Informational social influence and product evaluation. *Journal of Applied Psychology*, 56(1):54, 1972.

[19] M. d. Condorcet. *Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix*. Paris: L'Imprimerie Royale, 1785.

[20] V. Conitzer, M. Rognlie, and L. Xia. Preference functions that score rankings and maximum likelihood estimation. In *International Joint Conference on Artificial Intelligence*, pages 109–115, 2009.

[21] V. Conitzer and T. Sandholm. Common voting rules as maximum likelihood estimators. In *Conference on Uncertainty in Artificial Intelligence*, pages 145–152, 2005.

[22] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *International Conference on Recommender Systems*, pages 39–46, 2010.

[23] A. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: scalable online collaborative filtering. In *International Conference on World Wide Web*, pages 271–280, 2007.

[24] M. Deshpande and G. Karypis. Selective markov models for predicting web page accesses. *ACM Transactions on Internet Technology*, 4(2):163–184, 2004.

[25] S. Deterding, R. Khaled, L. E. Nacke, and D. Dixon. Gamification, toward a definition. In *CHI 2011 Gamification Workshop*, 2011.

[26] C. Dimitrakakis. Bayesian Variable Order Markov Models. In *International Conference on Artificial Intelligence and Statistics*, pages 161–168, 2010.

[27] M. Dudik, S. Lahaie, D. M. Pennock, and D. Rothschild. A combinatorial prediction market for the us elections. In *Conference on Electronic Commerce*, pages 341–358, 2013.

[28] M. Ekstrand, M. Ludwig, J. Konstan, and J. Riedl. Rethinking the recommender research ecosystem: reproducibility, openness, and lenskit. In *International Conference on Recommender Systems*, pages 133–140, 2011.

[29] S. Feng, L. Xing, A. Gogar, and Y. Choi. Distributional footprints of deceptive product reviews. In *International Conference on Weblogs and Social Media*, 2012.

[30] F. Galton. Vox populi. *Nature*, 75:450–451, 1907.

[31] Z. Gantner, S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. Mymedialite: A free recommender system library. In *International Conference on Recommender Systems*, pages 305–308, 2011.

[32] F. Garcin, C. Dimitrakakis, and B. Faltings. Personalized news recommendation with context trees. In *International Conference on Recommender Systems*, pages 105–112, 2013.

[33] F. Garcin and B. Faltings. Pen recsys: A personalized news recommender systems framework (extended). In *News Recommender Systems Workshop*, 2013.

[34] F. Garcin and B. Faltings. Pen recsys: A personalized news recommender systems framework (short). In *International Conference on Recommender Systems*, pages 469–470, 2013.

[35] F. Garcin, B. Faltings, and R. Jurca. Aggregating reputation feedback. In *International Conference on Reputation: Theory and Technology (ICORE)*, 2009.

[36] F. Garcin, B. Faltings, R. Jurca, and N. Joswig. Rating aggregation in collaborative filtering systems. In *International Conference on Recommender Systems*, 2009.

[37] F. Garcin, L. Xia, and B. Faltings. How aggregators influence human rater behavior? In *Workshop on Social Computing and User Generated Content*, 2013.

[38] F. Garcin, K. Zhou, B. Faltings, and V. Schickel. Personalized news recommendation based on collaborative filtering. In *International Joint Conferences on Web Intelligence and Intelligent Agent Technology*, pages 437–441, 2012.

[39] E. Giladi and Y. Klar. When standards are wide of the mark: nonselective superiority and inferiority biases in comparative judgments of objects and concepts. *Journal of Experimental Psychology: General*, 131(4):538, 2002.

[40] R. Hanson. Combinatorial information market design. *Information Systems Frontiers*, 5(1):107–119, 2003.

[41] R. Hanson. Logarithmic market scoring rules for modular combinatorial information aggregation. *Journal of Prediction Markets*, 1(1):3–15, 2007.

[42] J. Herlocker, J. Konstan, L. Terveen, and J. Riedl. Evaluating Collaborative Filtering Recommender Systems. *ACM Transactions on Information Systems*, 22:5 – 53, 2004.

[43] J. Herlocker, J. Konstan, L. Terveen, and J. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22:5–53, 2004.

[44] M. Hoffman, F. Bach, and D. Blei. Online learning for latent dirichlet allocation. In *advances in neural information processing systems*, pages 856–864, 2010.

[45] N. Hu, P. A. Pavlou, and J. Zhang. Can online reviews reveal a product's true quality?: empirical findings and analytical modeling of online word-of-mouth communication. In *Conference on Electronic Commerce*, pages 324–330, 2006.

[46] N. Hu, J. Zhang, and P. A. Pavlou. Overcoming the j-shaped distribution of product reviews. *Communications of the ACM*, 52(10):144–147, 2009.

[47] P. Huber. *Robust statistics*. Springer, 2011.

[48] W. IJntema, F. Goossen, F. Frasincar, and F. Hogenboom. Ontology-based news recommendation. In *Workshop on Data Semantics*, 2010.

[49] P. G. Ipeirotis, F. Provost, and J. Wang. Quality management on amazon mechanical turk. In *Human Computation Workshop*, 2010.

[50] H. J. Jung and M. Lease. Improving consensus accuracy via z-score and weighted voting. In *Human Computation Workshop*, 2011.

[51] R. Jurca and B. Faltings. Using chi-scores to reward honest feedback from repeated interactions. In *Conference on Autonomous Agents and Multiagent Systems*, pages 1233–1240, 2006.

[52] R. Jurca and B. Faltings. Mechanisms for making crowds truthful. *Journal of Artificial Intelligence Research*, 34(1):209, 2009.

[53] R. Jurca and B. Faltings. Incentives for answering hypothetical questions. In *Workshop on Social Computing and User Generated Content*, 2011.

[54] R. Jurca, F. Garcin, A. Talwar, and B. Faltings. Reporting incentives and biases in online review forums. *Transactions on the Web*, 4(2):1–27, 2010.

[55] H. Kajino, Y. Tsuboi, I. Sato, and H. Kashima. Learning from crowds and experts. In *Human Computation Workshop*, 2012.

[56] B. Kille, F. Hopfgartner, T. Brodt, and T. Heintz. The plista dataset. In *News Recommender Systems Workshop*, pages 16–23, 2013.

[57] E. Kirshenbaum, G. Forman, and M. Dugan. A live comparison of methods for personalized article recommendation at forbes.com. In *Machine Learning and Knowledge Discovery in Databases*, pages 51–66, 2012.

[58] A. Kumar and M. Lease. Learning to rank from a noisy crowd. In *Human Computation Workshop*, 2011.

[59] K. Lang. Newsweeder: Learning to filter netnews. In *International Conference on Machine Learning*, pages 331–339, 1995.

[60] R. Larrick and J. Soll. Intuitions about combining opinions: Misappreciation of the averaging principle. *Management Science*, 52(1):111–127, 2006.

[61] M. Lease. On quality control and machine learning in crowdsourcing. In *Human Computation Workshop*, pages 97–102, 2011.

[62] C. Leberknight, S. Sen, and M. Chiang. On the volatility of online ratings: An empirical study. In *Workshop on eBusiness*, 2012.

[63] K. Lerman and T. Hogg. Using a model of social dynamics to predict popularity of news. In *International Conference on World Wide Web*, pages 621–630, 2010.

[64] L. Li, W. Chu, J. Langford, and R. Schapire. A contextual-bandit approach to personalized news article recommendation. In *International Conference on World Wide Web*, 2010.

[65] J. Liu, P. Dolan, and E. Pedersen. Personalized news recommendation based on click behavior. In *International Conference on Intelligent User Interfaces*, pages 31–40, 2010.

[66] N. Liu, M. Zhao, E. Xiang, and Q. Yang. Online evolutionary collaborative filtering. In *International Conference on Recommender Systems*, pages 95–102, 2010.

[67] T.-Y. Liu. *Learning to Rank for Information Retrieval*. Springer, 2011.

[68] P. Lops, M. Gemmis, and G. Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender Systems Handbook*, pages 73–105. Springer, 2011.

[69] Y. Low, J. Gonzalez, A. Kyrola, D. Bickson, C. Guestrin, and J. Hellerstein. Graphlab: A new parallel framework for machine learning. In *Conference on Uncertainty in Artificial Intelligence*, 2010.

[70] A. Mao, A. Procaccia, and Y. Chen. Better human computation through principled voting. In *Conference on Artificial Intelligence*, 2013.

[71] M. McGlohon, N. Glance, and Z. Reiter. Star quality: Aggregating reviews to rank products and merchants. In *International Conference on Weblogs and Social Media*, 2010.

[72] N. Miller, P. Resnick, and R. Zeckhauser. Eliciting informative feedback: The peer-prediction method. *Management Science*, 51(9):1359–1373, 2005.

[73] B. Mobasher, R. Burke, R. Bhaumik, and C. Williams. Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness. *ACM Transactions on Internet Technology*, 7(4):23, 2007.

[74] A. Montgomery, S. Li, K. Srinivasan, and J. Liechty. Modeling online browsing and path analysis using clickstream data. *Marketing Science*, 23(4):579–595, 2004.

[75] H. Moulin. On strategy-proofness and single peakedness. *Public Choice*, 35:437–455, 1980.

[76] D. Oleson, A. Sorokin, G. Laughlin, V. Hester, J. Le, and L. Biewald. Programmatic gold: Targeted and scalable quality assurance in crowdsourcing. *Human Computation Workshop*, 2011.

[77] T. Pfeiffer, X. Gao, A. Mao, Y. Chen, and D. Rand. Adaptive Polling and Information Aggregation. In *Conference on Artificial Intelligence*, pages 122–128, 2012.

[78] J. Picault, M. Ribière, D. Bonnefoy, and K. Mercer. How to get the recommender out of the lab? In *Recommender Systems Handbook*, pages 333–365. Springer, 2011.

[79] J. Pitkow and P. Pirolli. Mining longest repeating subsequences to predict world wide web surfing. In *Proc. of USITS*, page 13, 1999.

[80] D. Prelec. A bayesian truth serum for subjective data. *Science*, 306(5695):462–466, 2004.

[81] A. Procaccia, S. Reddi, and N. Shah. A maximum likelihood approach for selecting sets of alternatives. In *Conference on Uncertainty in Artificial Intelligence*, 2012.

[82] P. Pu, L. Chen, and R. Hu. A user-centric evaluation framework for recommender systems. In *International Conference on Recommender Systems*, pages 157–164, 2011.

[83] G. Radanovic and B. Faltings. A robust bayesian truth serum for non-binary signals. In *Conference on Artificial Intelligence*, 2013.

[84] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *International Conference on World Wide Web*, pages 811–820, 2010.

[85] S. Rendle and L. Schmidt-Thieme. Online-updating regularized kernel matrix factorization models for large-scale recommender systems. In *International Conference on Recommender Systems*, pages 251–258, 2008.

[86] Research Studios Austria Forschungsgesellschaft mbH. easyrec. http://www.easyrec.org.

[87] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proc. of CSCW*, pages 175–186, 1994.

[88] J. Rissanen. A universal data compression system. *Transactions on Information Theory*, pages 656–664, 1983.

[89] J. Ross, L. Irani, M. S. Silberman, A. Zaldivar, and B. Tomlinson. Who are the crowdworkers?: shifting demographics in mechanical turk. In *International Conference on Human Factors in Computing Systems*, 2010.

[90] A. Said, A. Bellogin, J. Lin, and A. de Vries. Do recommendations matter?: News recommendation in real life. In *CSCW*, pages 237–240, 2014.

[91] A. Said, J. Lin, A. Bellogin, and A. de Vries. A month in the life of a production news recommender system. In *Workshop on Living Labs for Information Retrieval Evaluation*, pages 7–10, 2013.

[92] R. Sarukkai. Link prediction and path analysis using markov chains. *Computer and Telecommunications Networking*, 33(1-6):377–386, 2000.

[93] L. Savage. Elicitation of personal probabilities and expectations. *Journal of the American Statistical Association*, 66(336):783–801, 1971.

[94] G. Shani and A. Gunawardana. Evaluating recommendation systems. In *Recommender Systems Handbook*, pages 257–297. Springer, 2011.

[95] G. Shani, D. Heckerman, and R. Brafman. An mdp-based recommender system. *Journal of Machine Learning Research*, 6:1265–1295, December 2005.

[96] X. Su and T. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, pages 4:2–4:2, January 2009.

[97] M. Sun. How does the variance of product ratings matter? *Management Science*, 58(4):696–707, 2012.

[98] S. Suri, D. Goldstein, and W. Mason. Honesty in an online labor market. In *Human Computation Workshop*, 2011.

[99] J. Surowiecki. *The Wisdom of Crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations.* Knopf Doubleday Publishing Group, 2004.

[100] G. Tziralis and I. Tatsiopoulos. Prediction markets: An extended literature review. *The journal of prediction markets*, 1(1):75–91, 2012.

[101] D. Vengroff. Reclab: a system for ecommerce recommender research with real data, context and feedback. In *Workshop on Context-awareness in Retrieval and Recommendation*, pages 31–38, 2011.

[102] Y. Wang, L. Zhou, J. Feng, J. Wang, and Z. Liu. Mining complex time-series data by learning markovian models. In *Proc. of ICDM*, pages 1136–1140, 2006.

[103] J. Witkowski and D. Parkes. Peer prediction without a common prior. In *Conference on Electronic Commerce*, pages 964–981, 2012.

[104] J. Witkowski and D. Parkes. A robust bayesian truth serum for small populations. In *Conference on Artificial Intelligence*, 2012.

[105] L. Xia and V. Conitzer. A maximum likelihood approach towards aggregating partial orders. In *International Joint Conference on Artificial Intelligence*, pages 446–451, 2011.

[106] Y. Yan, R. Rosales, G. Fung, M. Schmidt, G. Hermosillo, L. Bogoni, L. Moy, J. Dy, and P. Malvern. Modeling annotator expertise: Learning when everybody knows a bit of something. In *International Conference on Artificial Intelligence and Statistics*, volume 9, pages 932–939, 2010.

[107] M. Zaki, C. Carothers, and B. Szymanski. Vogue: A variable order hidden markov model with duration based on frequent sequence mining. *Transactions on KDD*, 4:1–31, 2010.

[108] A. Zimdars, D. Chickering, and C. Meek. Using temporal data for making recommendations. In *Conference on Uncertainty in Artificial Intelligence*, pages 580–588, 2001.

[109] A. Zohar and J. Rosenschein. Robust mechanisms for information elicitation. In *Conference on Autonomous Agents and Multiagent Systems*, pages 1202–1204, 2006.

Florent Frédéric GARCIN

florent@garcin.ch (as of April 2014) Swiss

EDUCATION

since Jul. 08 **PhD student at the Artificial Intelligence Lab, EPFL, Switzerland**
Advisor: Prof. Boi Faltings

- Information Aggregation / Reputation Systems
  Studied optimal solutions to aggregate information from the crowd, in particular product reviews and ratings.
- Human Computation & Crowdsourcing
  Developed and implemented (Java/Spring/SQL) prediction markets and peer prediction as web platform (http://go.swissnoise.ch) to elicit private information from the crowd in order to predict future events.
- Recommender Systems
  Developed and implemented (Java/Spring/SQL) new algorithms for personalized news recommendation. Deployed on swissinfo.ch.

2012 **Visiting researcher at Harvard University, Cambridge, USA**
Center for research on computing and society. Host: Lirong Xia

2008 **Master of Science in Computer Science, EPFL, Switzerland**
Diploma Supplement in Internet Computing

2007 **Visiting researcher at Tsinghua University, Beijing, China**
Master thesis: "Cooperation in Underwater Sensor Networks"
Full scholarship from the Chinese government, Host: Fengyuan Ren

2006 **Minor in Management of Technology, EPFL, Switzerland**

PROFESSIONAL EXPERIENCE

since Jul. 08 **Research assistant at the Artificial Intelligence Lab, EPFL**
Teaching assistant: Artificial Intelligence, Computational Game Theory and Applications, Managing Multicultural Teams and Negotiation Techniques. Supervised 10+ bachelor/master projects. In charge of the IT infrastructure of the lab, setup new computational-intensive servers.

2006 **Internship in information security management, UBS AG, Zürich**

LANGUAGES

French	native speaker	**German**	intermediate (B1)
English	fluent (C2)	**Mandarin-Chinese**	intermediate (B1, HSKIII)

ACTIVITIES AND INTERESTS

Sports ski (instructor JS-1), telemark, squash, badminton, running, cycling
Travels strong interests in Asian culture
Associative *Balélec Festival:* team leader of transport and traffic group, stalls group.

Conference & Workshop Papers

- **Swissnoise: Online Polls with Game-theoretic Incentives**

  Florent Garcin and Boi Faltings
  Conference on Innovative Applications of Artificial Intelligence, 2014

- **How Aggregators Influence Human Rater Behavior?**

  Florent Garcin, LiRong Xia, and Boi Faltings
  Workshop on Social Computing and User Generated Content, 2013
  Workshop on Crowdsourcing and Online Behavioral Experiments, 2013

- **PEN RecSys: a Personalized News Recommender Systems Framework**

  Florent Garcin and Boi Faltings
  short: Conference on Recommender systems, 2013
  extended: News Recommender Systems Workshop, 2013

- **Personalized News Recommendation with Context Trees**

  Florent Garcin, Christos Dimitrakakis, and Boi Faltings
  Conference on Recommender Systems, 2013

- **Personalized News Recommendation Based on Collaborative Filtering**

  Florent Garcin, Kai Zhou, Boi Faltings, and Vincent Schickel
  Conference on Web Intelligence and Intelligent Agent Technology, 2012

- **Rating Aggregation in Collaborative Filtering Systems**

  Florent Garcin, Boi Faltings, Radu Jurca, and Nadine Joswig
  Conference on Recommender Systems, 2009

- **Aggregating Reputation Feedback**

  Florent Garcin, Boi Faltings, and Radu Jurca
  Conference on Reputation: Theory and Technology, 2009

- **Cooperation in Underwater Sensor Networks**

  Florent Garcin, Mohammad Hossein Manshaei, Jean-Pierre Hubaux
  Conference on Game Theory for Networks, 2009

- **A Study of Forward Error Correction Schemes for Reliable Transport in Underwater Sensor Networks**

  Florent Garcin, Bin Liu, Fengyuan Ren, and Chuang Lin
  Conference on Sensor, Mesh and Ad Hoc Communications and Networks, 2008

Journal Papers

- **Reporting Incentives and Biases in Online Review Forums**

  Radu Jurca, Florent Garcin, Arjun Talwar, and Boi Faltings
  Transactions on the Web, Volume 4 Issue 2, April 2010