PROCEEDINGS

IJCAI 2011 - 9th Workshop on

INTELLIGENT TECHNIQUES FOR WEB PERSONALIZATION

&

RECOMMENDER SYSTEMS (ITWP'11)

BARCELONA, SPAIN, 16 JULY 2011

Editors:
Sarabjot Singh Anand, Dietmar Jannach,
Bamshad Mobasher and Alfred Kobsa

In Conjunction with Twenty-Second International Joint Conference on Artificial Intelligence
July 16 – 22, 2011 Barcelona, Catalonia (Spain) http://ijcai-11.iiia.csic.es/

Copyright and Bibliographical Information

Proceedings of the 9th Workshop on Intelligent Techniques for Web Personalization & Recommender Systems (ITWP 2011), in conjunction with Twenty-Second International Joint Conference on Artificial Intelligence
July 16 – 22, 2011 Barcelona, Catalonia (Spain) http://ijcai-11.iiia.csic.es/

Sarabjot Singh Anand, Dietmar Jannach, Bamshad Mobasher and Alfred Kobsa (editors).

Workshop Web Site

Additional information about the workshop, including the electronic versions of accepted papers and links to related workshops will be maintained at the workshop Web site:

http://ls13-www.cs.uni-dortmund.de/homepage/itwp2011/

Workshop chairs

- **Sarabjot Singh Anand**
  Department of Computer Science, University of Warwick, UK
- **Dietmar Jannach**
  Department of Computer Science, TU Dortmund, Germany
- **Bamshad Mobasher**
  School of Computing, DePaul University, USA
- **Alfred Kobsa**
  University of California, USA

Program committee / Reviewers

- Esma Aïmeur, Université de Montréal, Canada
- Gediminas Adomavicius, CSOM, University of Minnesota
- Liliana Ardissono, University of Torino, Italy
- Bettina Berendt, K.U.Leuven, Belgium
- Shlomo Berkovsky, CSIRO, Australia
- José Luís Borges, University of Porto, Portugal
- Derek Bridge, University College York, Ireland
- Robin Burke, DePaul University, USA
- Alexander Felfernig, Technical University Graz, Austria
- Gerhard Friedrich, University Klagenfurt, Austria
- Rayid Ghani, Accenture, USA
- Andreas Hotho, University of Karlsruhe, Germany
- Alipio Jorge, University of Porto, Portugal
- Mark Levene, University College, London, UK
- Frank Linton, The MITRE Corporation, USA
- Alexandros Nanopulos, Aristotle University of Thessaloniki, Greece
- Olfa Nasraoui, University of Louisville, USA
- Claire Nedellec, Université Paris Sud, Paris, France
- George Paliouras, Demokritos National Centre for Scientific Research, Greece
- Naren Ramakrishnan, Virginia Tech, USA
- Francesco Ricci, Free University of Bozen/Bolzano, Italy
- Lars Schmidt-Thieme, University of Hildesheim, Germany
- Spiros Sirmakessis, University of Patras, Greece
- Barry Smyth, University College Dublin, Ireland
- Markus G. Stolze, IBM Watson Research Center, NY
- Alex Tuzhilin, Stern School of Business, New York University, USA
- Suk-Chung Yoon, Widener University, Pennsylvania, USA
- Markus Zanker, University Klagenfurt, Austria
- Daniel Zeng, University of Arizona

Preface

Web Personalization and recommendation systems have been steadily gaining ground as essential components of today's Web based applications, including in e-commerce and the delivery of business services, and in providing support for Web search and navigation in information rich domains. The proliferation of Web 2.0 applications has allowed users to go beyond simple consumers of information and instead actively participate in shaping collaborative environments in which users, resources, and user provided content are all networked together. There is, therefore, an increased need for more intelligent and personalized tools that help users navigate these complex information spaces. These tools include a new generation of recommender systems that integrate multiple online channels, are more scalable and more adaptive, and can better handle user interactivity. To achieve this, such applications must rely on intelligent techniques from AI, machine learning, Web mining, statistics, and user modeling in order to leverage and mine all available data, including user profiles, the content and meta-data associated with resources, and underlying network structures.

The aim of this workshop is to bring together researchers and practitioners From Web Mining, Web Personalization, Recommender Systems, and User Modeling communities in order to foster an exchange of information and ideas and to facilitate a discussion of current and emerging topics related to the development of intelligent Web personalization and Recommender Systems. This workshop represents the 9th in a successful series of ITWP workshops that have been held at IJCAI, AAAI and UMAP since 2001 and would be – after the successful events at AAAI'07, AAAI'08, IJCAI'09 and UMAP'10 – the 4th combined workshop on ITWP and Recommender Systems.

This year's workshop attracted a number of high-quality contributions of which seven long and two short papers were accepted for presentation at the workshop. These accepted papers span a variety of issues and techniques related to personalization and recommender systems. Specifically, the papers deal with such topics as recommendation in the social web, using social relations/features in recommendation, learning to model preferences from crowdsourced data, the generation of user recommendations based by integrating and aggregating online data sources; context awareness in recommendation, integration of domain ontologies in similarity measurement, content based filtering for streaming broadcast; recommendation in the physical world; adaptation of web site structure based on user navigation patterns and item placement in diverse recommendation lists.

This year's workshop also includes an invited talk: "Personalization and context-awareness in Retrieval and Recommender Systems" by Ernesto de Luca, DAI-Labor, TU Berlin.

ITWP 2011 Organizing Committee
Sarabjot Singh Anand, Dietmar Jannach, Bamshad Mobasher and Alfred Kobsa

Table of Contents

Invited talk

Dr. Ernesto William De Luca

Personalization and context-awareness in Retrieval and Recommender Systems

Context-aware information is widely available in various ways such as interaction patterns, location, devices, annotations, query suggestions and user profiles and is becoming more and more important for enhancing retrieval performance and recommendation results. At the moment, the main issue to cope with is not only recommending or retrieving the most relevant items and content, but defining them ad hoc. Further relevant issues are personalizing and adapting the information and the way it is displayed to the user's current situation (device, location) and interests.

In this talk I will discuss how personalization and contextual information can be integrated in retrieval and recommendation systems. Two main issues are recognized: a general content context and a user-centric content context. A general content context is a common case defined by time, weather, location and many similar other aspects. A user-centric content context is given by the content of user profiles such as language, interests, devices used for interaction, etc. The inclusion of these two contexts in information systems can help in delivering better structured results that can be personalized and better match the user needs and expectation. Results of such integration will be shown and discussed.

Biographical Sketch

Dr. Ernesto William De Luca is Head of the Competence Center for Information Retrieval and Machine Learning at the DAI-Lab, Berlin Institute of Technology, Germany. His research areas include Semantic Web technologies, Recommender Systems and Information Retrieval. He has authored more than 50 papers on national and international conferences, books and journals in these fields. He has organized a large number of workshops and served as programme committee member at top level conferences. At the moment, he is co-organizing the 2nd Semantic Personalized Information Management (SPIM 2011) Workshop at the 10th International Semantic Web Conference (ISWC 2011) and the 2nd Challenge on Context-aware Movie Recommendation (CAMRa2010) at the 5th ACM Recommender Systems 2011 Conference (RecSys2011).

Measuring Semantic Similarity using a Multi-Tree Model

Behnam Hajian and **Tony White**
School of Computer Science Carleton University, Ottawa, Canada
{bhajian,arpwhite}@scs.carleton.ca

Abstract

Recommender systems and search engines are examples of systems that have used techniques such as Pearson's product-momentum correlation coefficient or Cosine similarity for measuring semantic similarity between two entities. These methods relinquish semantic relations between pairs of features in the vector representation of an entity. This paper describes a new technique for calculating semantic similarity between two entities. The proposed method is based upon structured knowledge extracted from an ontology or a taxonomy. A multi-tree concept is defined and a technique described that uses a multi-tree similarity algorithm to measure similarity of two multi-trees constructed from taxonomic relations among entities in an ontology. Unlike conventional linear methods for calculating similarity based on commonality of attributes of two entities, this method is a non-linear technique for measuring similarity based on hierarchical relations which exist between attributes of entities in an ontology. The utility of the proposed model is evaluated by using Wikipedia as a collaborative source of knowledge.

1 Introduction

Similarity refers to psychological nearness between two concepts. Similarity has roots in psychology, social sciences, mathematics, physics and computer science [Larkey and Markman, 2005]. In social psychology, similarity points to how closely attitudes, values, interests and personality match between people which can lead to interpersonal attraction. This can be explained by the fact that similar people tend to place themselves in similar settings and this consequently decreases potential conflicts between them. Furthermore, finding a person with similar tastes helps to validate values or views held in common. With a mental representation, the similarity between two concepts is defined as a function of the distance between two concepts represented as different points in the mental space [Tversky and Shafir, 2004].

Semantic similarity is used to refer to the nearness of two documents or two terms based on likeness of their meaning or their semantic contents [Tversky and Shafir, 2004]. Conventionally, statistical means (e.g., a vector space model) can estimate the distance between two entities by comparing features representing entities [Salton *et al.*, 1975]. For example, in order to compare two documents, the frequency of co-occurrence of words in the text corpus represents the similarity between the two documents. Semantic relatedness is a broader term than semantic similarity, with the former including other concepts such as antonymy and meronymy. However, in certain literature these two terms are used interchangeably .

We can compare the similarity of two concepts by measuring the commonality of their features. Since each concept is represented by the features describing its properties, a similarity comparison involves comparing the feature lists representing that concept. Simply put, concepts which are near to each other are more similar than points which are conceptually distant. There are several mathematical techniques for estimating this distance, such as latent semantic analysis (LSA) [Landauer *et al.*, 1998]. Measuring similarity among entities has applications in many areas such as: recommendation systems, e-commerce, search engines, biomedical informatics and in natural language processing tasks such as word sense disambiguation.

For instance, in user-based collaborative filtering, the system tries to find people with similar tastes and recommend items highly ranked by the people which might be interesting to their peers. Finding people with similar tastes involves processing of their historical transactions (i.e., items viewed and ranked by them in their previous transactions) and calculating similarity between them using one of the methods described above. On the other hand, in content-based recommender systems and search engines, the system finds items which are more similar to show to a user based on his/her query and the similarity of the items (i.e., products). This category of system calculates similarity between products based on the commonality of the features of different products.

In information retrieval (IR) and search engines, words are considered as features in a document or a query. In IR systems, it is conventional to represent a document by a bag-of-words (BOW). A Vector Space Model (VSM) is generally used to estimate the similarity between two documents in classification/clustering tasks or to estimate similarity between a query and documents in keyword-based search engines.

1.1 Contribution, Motivations and Paper Structure

In the Vector Space Model (VSM), a document or a query is represented as a vector of identifiers such as index terms. However, in many cases, conventional methods such as Dices coefficient, Pearson's correlation coefficient, Jaccards index or cosine similarity, which use VSM to represent a document, do not perform well. This is due to a document being represented in a linear form (i.e., a vector of features) in which semantic relations among features are ignored. Examples of such problems are: ignoring polysemy (terms having different sense with a same spelling such as apple as a fruit and apple as a company) and synonymy (terms with different spelling having a same sense such as big and large). An example of the latter problem can be found in recommender systems which find people with similar tastes according to their previous transactions. An example of this problem is demonstrated in the following example in which similarity of tastes for two people are estimated based on their previous transactions:

- T1 (Clothes, Boxspring, Mp3Player, Mattress, LCD TV)
- T2 (Dress, Bed, Mattress, iPod Touch, LED TV)

Using the VSM-based method for computing similarity between the above transactions, these transactions are no longer similar at all. However, intuitively we have a feeling that LED TV and LCD TV are related to each other since both are subclasses of TV. This observation is also true when comparing iPod Touch and Mp3 Player and for the relationship between Clothes and Dress.

In Recommender systems, a basic problem to be solved is that of the similarity of one item or set of items to another item or set of items. Either an individual is to be compared to another or one set of items is to be compared to another set. As the above example demonstrates, the two individuals have some similarity in their purchasing profiles and that the purchasing profile of person P1 (responsible for T1) has some similarity to that of person P2 (responsible for T2). A survey on next generation recommender systems [Adomavicius and Tuzhilin, 2005] proposes ”improved modelling of users and items” that arguably includes the type of semantic extensions proposed in this paper.

This paper proposes replacing the VSM with a non-linear representation for an entity. The proposed representation models an entity by its features in a hierarchical format using an ontology called a semantic multi-tree. Multi-tree similarity considers semantic relations among the features of entities in a hierarchical structure using the ontology classification. This method enhances conventional information retrieval techniques by computing similarity regarding commonality of not only the features but also commonality of semantic relations among features and their parents.

The rest of the paper is organized as follows: The next section provides background information on the VSM including analysis of its limitations as well as related work. In Section 3, we define our model for representing entities called the semantic multi-tree. Section 4 concentrates on the definition of the proposed method for measuring similarity by use of a semantic multi-tree model. In the following section, the technique is validated against human judgment in WordSimilarity-353 and Rubenstein and Goodenough using Wikipedia categories as a taxonomy. A discussion follows, with conclusions and future work provided in the final section.

2 Background and Related Work

The Vector Space Model is defined as an algebraic model in which a document or an entity is represented as a vector of its features; for example, a document which is represented as a vector of index terms [Salton and McGill, 1983]: $d_j = (w_{1j}, ..., w_{nj})$. In these vectors, w_{ij} represents the number of occurrences of the i^{th} term in the j^{th} document. There are two model representation schemes. The superior scheme for representation of vectors in this model is term frequency-inverse document frequency (tf-idf). In the other scheme, w_{ij} is a binary representation of the occurrence of a corresponding term. In order to retrieve a document among a collection of documents, we have to calculate the similarity between our query and all of the documents in the collection and choose the most relevant documents among them. A frequently used method for estimating the similarity is calculating the cosine of the angle between the vectors representing query and a document. The higher the cosine of the angle between two vectors the more similar the vectors and, therefore, the more similar the entities represented by the vectors.

$$cos\theta = sim(d_i, q) = \frac{d_i.q}{|d_i||q|} \tag{1}$$

Another method for calculating similarity is Pearson product-moment correlation coefficient (PMCC). This method calculates the correlation (linear dependence) between two vectors. The PMMC of two vectors is defined as:

$$sim(d_i, q) = \frac{cov(d_i.q)}{\sigma d_i \times \sigma q} \tag{2}$$

Calculation of similarity is straightforward with these methods but a disadvantage is that neither of them considers semantic relations among features.

A conventional method for computing semantic relatedness is the corpus-based technique that relies on the tendency for related words to appear in similar texts called Latent Semantic Analysis (LSA). Unfortunately, LSA is only able to provide accurate results when the corpus is very large.

In recent years, several techniques have been developed – such as the work proposed by Ted Pedersen et. al – for estimating similarity by measuring semantic distance between two words in WordNet [Pedersen *et al.*, 2005]. A limitation of using lexical databases such as WordNet or similar resources is that they have been created by one or a group of linguists rather than experts in different subjects. Furthermore, WordNet is rarely revised when compared to collaborative knowledge sources such as Wikipedia. As a result, WordNet does not include some special proper nouns in different areas of expertise (e.g., Obama, Skyneedle). Recently, Wikipedia has compensated for this lack of knowledge by providing a mechanism for collaboratively creating knowledge. Wikipedia includes a wide range of articles about almost every entity in the world by using human expertise in different areas. In addition, as of May 2004, Wikipedia articles have been categorized by providing a taxonomy; namely, categories. This facility provides hierarchical categorization with multiple parents for a node by means of a multi-tree structure. This observation motivates us to use Wikipedia as a resource of knowledge in this paper.

There are several approaches for measuring semantic relatedness using resources such as WordNet or Wikipedia categories as a graph or network by considering the number or length of paths between concepts. In the WikiRelate project, Ponzetto and Strube used three measures for computing semantic relatedness: First, a path-based measure using the length of the path between two concepts; second, an information content-based measure and third, the overlap-based measure which applies the Lesk algorithm that defines the relatedness between two words as a function of the overlap between two contexts defining the corresponding words. In WikiRelate [Strube and Ponzetto, 2006], a pair of Wikipedia pages is first retrieved, then categories they refer to are extracted and finally, the relatedness between two concepts is computed regarding the paths found between two concepts in the Wikipedia categories. In the last step, Ponzetto and Strube calculate relatedness by selecting the shortest path and the paths which maximize the information content-based measure.

In contrast with statistical methods for computing relatedness such as LSA, Gabrilovich and Markovitch proposed Explicit Semantic Analysis (ESA) using meaning in natural concepts derived from Wikipedia [Gabrilovich and Markovitch, 2007]. In this work, they used Wikipedia articles for augmenting the text representation and constructing a weighted list of concepts. They finally used tf-idf and conventional machine learning methods to calculate relatedness between the weighted vectors constructed in the previous steps.

Milne and Witten used cross referencing in the Wikipedia link database in order to obtain semantic relatedness between two concepts called Wikipedia Link-based Measure (WLM) [Witten and Milne, 2008]. In WLM, they used tf-idf using link counts weighted by the probability of occurrence of a term in an article. Almost all of the previous research has used tf-idf and VSM to calculate the relatedness between two sets of features.

Ruiz-Casado et al. [Ruiz-Casado *et al.*, 2005] proposed a procedure for automating the extension of an existing ontology or lexical semantic network using Wikipedia. In their work, WordNet was used in conjunction with the english Wikipedia to generate very high accuracy for polysemous words. Furthermore, there are several works based on computation of similarity between two ontologies, such as [Rodríguez and Egenhofer, 2003] in which the authors used the set theoretic Tversky similarity measure in the matching process. The model proposed focuses on the matching between classes and entities in two ontologies. [Maguitman *et al.*, 2005] proposes a graph-based algorithm for calculating similarity between entities in a graph. This algorithm is based on the Open Directory Project (ODP) which is a large human-constructed directory of the Web, using portals and search engines. One of the models which proposes a new similarity model based on the structure of an ontology is [Schickel-Zuber and Faltings, 2007]. This paper proposes Ontology Structure based Similarity (OSS) in which an a-priori score is calculated to measure similarity between two concepts in an ontology. [Lee *et al.*, 2008] also concentrated on measuring similarity between two concepts in a single ontology. None of these works tries to model an entity or a text document by its features by a structured dictionary such as Wikipedia categories in order to measure semantic similarity between two documents.

3 The Semantic Multi-Tree Model

Semantic tree is a term with several different meanings in computer science. The term is regularly used as an alternative term for a semantic tableaux which is a very well known method in logic (i.e., a resolution method for mechanized reasoning) [Annates, 2005]. Semantic tree in this paper is interpreted as the taxonomy of entities in an ontology.

Taxonomy in the literature is defined as the practice and science of classification. Almost all objects, places, concepts, events, properties and relations can be classified into a taxonomic hierarchy. In the ontological literature, taxonomy refers to a set of concepts with *is-a* (i.e., SubClassOf/InstanceOf) relations between them. Therefore, we consider taxonomy as a narrower concept than ontology since ontology includes broader relations such as *part-of*, *has-a*, *rules*, *axioms* and *events* as well as *classes*, *hierarchical relations* and *attributes*.

Definition 1: A taxonomy, $\mathcal{O}$, is defined as a set of classes, and *is-a* relations between them, $\mathcal{O} = (\mathcal{C}_\mathcal{O}, \mathcal{R}_\mathcal{O})$. In formal logic, the *is-a* relation is defined as a *subclass/instance-of* relation in an ontology:

Subclass/Instance-of: $\forall x : c_i(x) \rightarrow c_j(x)$. Such that $\forall c_i, c_j \in \mathcal{C}_\mathcal{O}, \textit{is-a}(c_i, c_j) \in \mathcal{R}_\mathcal{O}$.

Definition 2: A Multi-Tree is defined as a tree data structure in which each node may have more than one parent. A multi-tree is often used to describe a partially ordered set. In this paper, a taxonomy of concepts is modelled by a multi-tree structure in which each concept may refer to multiple super-concepts. It should be noted that in taxonomies such as Wikipedia Categories and WordNet cycles do exist; however, we have avoided capturing them by breaking edges creating directed cycles and capturing the rest of the graph by using a multi-tree structure. It should also be noted that the definition used here is more general in that the algorithms used allow for the existence of multiple paths between a leaf and root node; i.e., the diamond-free poset requirement is relaxed.

Formally, in this paper, a multi-tree is a directed acyclic graph, $T = (V, E, C, L, M, W, P)$, with hierarchical categorization of its nodes in different levels such that:

- V is a set of vertices (nodes), $V = \{v_1, ..., v_n\}$. Each vertex corresponds to a concept in the taxonomy.
- E is a set of edges, $E = \{e_1, ..., e_n\}$, (in which $e = \langle v_i, v_j \rangle$ is an ordered set representing an edge from node v_i to node v_j. Each edge represents an *is-a* relation between two concepts c_i, c_j which means (c_i *is-a* c_j). The direction in this digraph is always from a concept (subclass) to its parent (super-class).

- *C* is a set of terms representing concepts which are used as nodes labels.
- *L* is a function mapping V to $\mathbb{R}$ $L : V \to \mathbb{R}$ assigning a real number to each node. This function is recursively defined as being 1 plus the average value of L for the children of the node. Initially, this function assigns 0 to the leaf nodes.
- *M* is a bijective mapping function mapping V to C $(M : V \to C)$ assigning a label (representing a concept) to a node.
- *W* is a function mapping V to $\mathbb{R}$ $(W : V \to \mathbb{R})$ which assigns a real number as a weight to each node. This weight is utilized to calculate the similarity between two entities, which will be discussed in the next section.
- *P* is a function mapping E to $\mathbb{R}$ $(P : E \to \mathbb{R})$ which assigns a real number to each edge as a propagation ratio of each edge. In this paper P was set to 1.

The following functions, properties and operators are defined for a Multi-Tree:

- *leaf(v)* is a function mapping V to $\{true, false\}$ that returns a Boolean value indicating whether a node is a leaf node or not. A leaf node in Multi-Tree does not have any children. A multi-tree may have several leaves.
- *root(v)* is a function mapping V to $\{true, false\}$ that returns a Boolean value indicating whether a node is a root node or not. A Multi-Tree node is a root if it does not have any parents. A multi-tree has only one root node.
- *children(v)* is a function mapping V to P(V) (the power set of V) that returns the set of all the direct children of the node.
- *parents(v)* is a function mapping V to P(V) (the power set of V) that returns the set of all the direct parents of the node.
- $\beta_v = |children(v)|$ is defined as the cardinality of the child set of node v. (count of children of the node v)
- $\gamma_v = |parents(v)|$ is defined as the cardinality of the parent set of node v. (count of parents of the node v)
- The combination operator with the symbol $\uplus$ is defined between two multi-trees T_1, T_2 and returns a multi-tree T_u containing all the vertices and edges that exist in both T_1 and T_2. In other words, this operator returns the combination of two multi-trees. $T_u = T_1 \uplus T_2 \Rightarrow$

$$T_u = \begin{cases} E_u = E_1 \cup E_2 \\ V_u = V_1 \cup V_2 \\ C_u = C_1 \cup C_2 \\ \end{cases} \begin{cases} L^{T_u} \\ M^{T_u} \\ P^{T_u} \\ W^{T_u} \end{cases} \quad (3)$$

- The weights of the vertices in the tree T_u are calculated by a recursive function $W^{T_u} : V \times \mathbb{R} \to \mathbb{R}$ as defined in equation 4. In the proposed algorithm, weight is propagated from the leaves to the root of the multi-tree combined from two multi-trees. In this equation, α is a damping factor (degradation ratio). The damping factor causes the nodes at lower levels of a multi-tree (i.e., nodes near to the leaves) to contribute more to the weight than nodes in higher levels.

$$W^{T_u}(v_i, \alpha) = \begin{cases} \Delta^{T_u}(v_i) & \textit{leaf}(v_i)\textit{=true} \\ \rho^{T_u}(v_i, \alpha) & \textit{root}(v_i)\textit{=true} \\ \Phi^{T_u}(v_i, \alpha) & \text{Otherwise} \end{cases} \quad (4)$$

This function considers nodes in a multi-tree in three categories: leaves, root and nodes situated between leaves and the root whose weights are calculated by functions Δ, ρ and Φ respectively.

- Δ is a function mapping $V \to \{0, 1\}$. This function determines whether a specific node, v_i, in a combined tree T_u exists in both of the trees from which it is constituted (T_1, T_2).

$$\Delta^{T_u}(v_i) = \begin{cases} 1 & \text{if } v_i \in V^{T_1}, v_i \in V^{T_2} \\ 0 & \text{Otherwise} \end{cases} \quad (5)$$

- ρ is a function mapping $V \times \mathbb{R} \to \mathbb{R}$. This function is used to calculate the weights of the nodes in a multi-tree.

$$\rho^{T_u}(v_i, \alpha) = (\frac{1}{\beta_{v_i}})(\sum_{\forall v_x \in children(v_i)} P(v_i, v_x) W^{T_u}(v_x, \alpha)) \quad (6)$$

- Φ is a function mapping $V \times \mathbb{R} \to \mathbb{R}$. This function returns the weight of a node if the node is neither a leaf node nor the root of the multi-tree.

$$\Phi^{T_u}(v_i, \alpha) = (1 - \frac{1}{\alpha^{L(v_i)+1}})\rho^{T_u}(v_i, \alpha) \quad (7)$$
$$+(\frac{1}{\alpha^{L(v_i)+1}})\Delta^{T_u}(v_i)$$

The Φ function calculates the weight of a node by $1 - \frac{1}{\alpha^{L(v_i)}}$ share of the average of the weight of its children which calls the function to calculate the weight of each child recursively and $\frac{1}{\alpha^{L(v_i)}}$ share for the commonality of a node between the multi-trees of two concepts.

The above description is best illustrated by the following example. The example, shown in Figure 1,2 and the calculation using equations 5-7 illustrated in Figure 3, demonstrates how the above functions work for two entities represented by their features (i.e., products appeared in the profiles of two users).

d_1=(Web Cam, Digital Camera, LCD TV, Blender, Mattress)
d_2=(Keyboard, DSLR Camera, LED TV, Mattress, Drawer)

Using a VSM, the similarity between d_1, d_2 is equal to 0.2. However, using the proposed method, although LED and LCD are not equal they have a parent in common which makes the weight non-zero. Considering $\alpha = e = 2.71$ (also used in experiments reported later), the similarity between d_1, d_2 is: 0.444.

W(Keyboard)=0, W(Web Cam)=0, W(Digital Camera)=0, W(DSLR)=0, W(LED)=0, W(LCD)=0, W(Blender)=0, W(Drawer)=0, W(Mattress)=1, W(Kitchen)=0

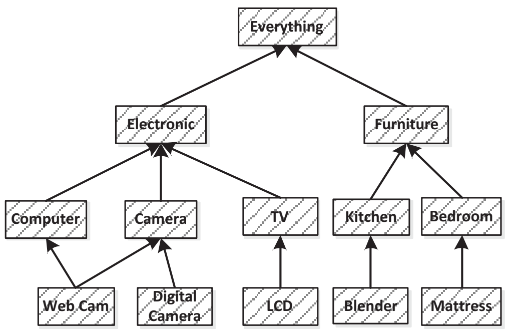


Figure 1: First multi-tree representing transaction d_1

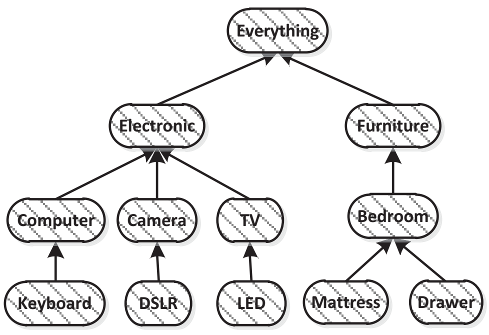


Figure 2: Second multi-tree representing transaction d_2

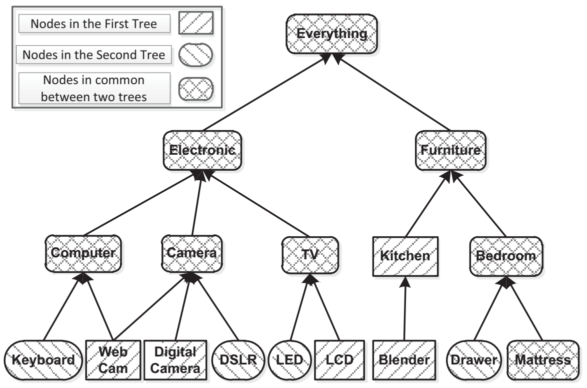


Figure 3: A multi-tree combined from previous two multi-trees

W(Computer)=W(TV)=W(Camera)=$(1-\frac{1}{e})(0)+(\frac{1}{e}) = 0.369$
W(Bed room)=$(\dfrac{1}{2})\times(1-\frac{1}{e})+(\frac{1}{e}) = 0.684$
W(Electronic)=$(\frac{1}{e})\times(1-\frac{1}{e^2})+(\frac{1}{e^2}) = 0.457$
W(Furniture)=$(\dfrac{0+0.684}{2})\times(1-\frac{1}{e^2})+(\frac{1}{e^2}) = 0.431$
W(Everything)=0.444

It is clear from the above example that the proposed method generates a higher similarity value when ontological information is included. From a recommender system perspective this is desirable in that it allows purchasing profiles to be compared in a semantic manner.

4 Similarity using a Semantic Multi-Tree

In order to calculate the similarity between two entities, we construct two multi-trees each of which represents features of the corresponding entity in a hierarchical format according to a specific ontology or taxonomy. In this method, each entity is represented by its features as leaves of a multi-tree. The rest of each multi-tree is constructed according to the domain taxonomy (e.g., WordNet or Wikipedia Categories). Hence, the multi-tree corresponding to each entity is a sub multi-tree of the taxonomy with which the sub-multi-tree is constructed. A multi-tree T_x is said to be a sub multi-tree of $T_\mathcal{O}$ if:

$$T_x \subseteq T_\mathcal{O} \Leftrightarrow \begin{cases} V_x \subseteq V_\mathcal{O} \\ E_x \subseteq E_\mathcal{O} \\ C_x \subseteq C_\mathcal{O} \end{cases} \tag{8}$$

Assume that, $T_\mathcal{O} = (V_\mathcal{O}, E_\mathcal{O}, C_\mathcal{O}, L_\mathcal{O}, M_\mathcal{O}, W_\mathcal{O}, P_\mathcal{O})$, is a multi-tree representing the domain taxonomy (e.g., Wikipedia Categories), $\mathcal{O} = (\mathcal{C}_\mathcal{O}, \mathcal{R}_\mathcal{O})$, in which $\mathcal{C}_\mathcal{O}$ represents set of concepts and $\mathcal{R}_\mathcal{O}$ represents set of relations among concepts in the taxonomy. The transformation function $\mathcal{T}$ is defined as a bijective function $\mathcal{T} : \mathcal{R}_\mathcal{O} \to E$ which maps each relation in the taxonomy $\mathcal{O}$ to an edge in the multi-tree $T_\mathcal{O}$. So, $E_x = \{e_i = \mathcal{T}(R_i) \mid R_i \in \mathcal{R}_\mathcal{O}\}$. ($C_\mathcal{O} \equiv \mathcal{C}_\mathcal{O}$ and $E_\mathcal{O} \equiv \mathcal{R}_\mathcal{O}$).

The multi-tree, $T_x = (V_x, E_x, C_x, L_x, M_x, W_x, P_x) \subseteq T_\mathcal{O}$, corresponds to entity $d_x = (c_1, ..., c_n)$ in which c_i is a term representing a feature of this entity as well as a concept in the taxonomy. We define $C_x \subseteq C_\mathcal{O}$ in multi-tree T_x as a set of terms representing features of the entity d_x.

Hence, $C_x = \{c_1, ..., c_n\} \cup \{c_j \mid \forall c_i \in C_x, \forall c_j \in \mathcal{C}_\mathcal{O}, \textit{is-a}(c_i, c_j) \in \mathcal{R}_\mathcal{O}\}$, $V_x = \{v_i = M(t_i) \mid t_i \in C_x\}$ and $E_x = \{e_i = \mathcal{T}(R_i) \mid \forall c_k, c_l \in C_x, R_i(c_k, c_l) \in R_\mathcal{O}\}$ such that $C_x \subseteq \mathcal{C}_\mathcal{O}$ and $\mathcal{O} = (\mathcal{C}_\mathcal{O}, \mathcal{R}_\mathcal{O})$ is the taxonomy which is used to construct the tree.

In the next step, the combination operator is applied to the two trees whose similarity is being computed. Applying the combination operator to the two trees, the weight of the root of the combined tree represents the similarity of the two trees. The weight of the root is recursively calculated by application of equations 5-7. The following steps demonstrate the process of calculating the similarity between two entities d_1, d_2 represented by sets of features C_1, C_2 respectively:

1. Construct multi-trees T_1 and T_2 from sets of features C_1 and C_2 respectively.
2. Construct $T_{sim} = T_1 \uplus T_2$ as a combination of two multi-trees $T_1, T_2 \subseteq T_\mathcal{O}$
3. Update the weights for the nodes in the combined multi-tree T_{sim} using the recursive equations 5-7.
4. The weight of the root of T_{sim} is the value which represents the similarity of two entities represented by C_1, C_2; i.e., $Sim(d_1, d_2) = W(root(T_{sim}))$.

Algorithm 1 and 2 describes the process by which a multi-tree is constructed from a feature set representing an entity.

Algorithm 1 Constructing a multi-tree.

```
Proc ConstructMulti-Tree(ConceptSet $C_x$)
$T_x \leftarrow null$
for all c in $C_x$ do
    FindPaths($T_{\mathcal{O}}.M^{-1}(c), T_{\mathcal{O}}, T_x$)
end for
return $T_x$
```

Algorithm 2 Finding a path in a multi-tree from a leaf node to root.

```
Proc FindPaths(Node v,Multi-Tree $T_{\mathcal{O}}$,Multi-Tree $T_x$)
if root(v) then
    $T_x.root \leftarrow v$
    return
end if
for all parent in $T_{\mathcal{O}}$.Parents(v) do
    if parent not in $T_x.V$ then
        $T_x.V \leftarrow T_x.V \cup parent$
        $T_x.E \leftarrow T_x.E \cup < parent, v >$
        FindPaths( parent,$T_{\mathcal{O}}, T_x$)
    end if{avoid cycles}
end for
```

Algorithm 3 describes the calculation of similarity using the proposed non-linear method.

Algorithm 3 Calculation of similarity using multi-trees.

```
$T_1 \leftarrow$ConstructMulti-Tree(ConceptSet $C_1$)
$T_2 \leftarrow$ConstructMulti-Tree(ConceptSet $C_2$)
$T_{Sim} \leftarrow T_1 \uplus T_2$
$similarity \leftarrow W^{T_{Sim}}(root, \alpha)$
```

5 Experimental Results

The proposed model is not only useful for measuring similarity between pairs of words but is also useful for information retrieval and recommender systems. In this paper, we evaluated the semantic multi-tree model for the application of measuring similarity between pairs of words, but the domain of the proposed model is not limited just to the application of measuring similarity between pairs of words. Our rationale for doing this is that the concept domain is potentially larger as we are not limited to (say) movies, music or books, domains often used in recommender datasets.

One of the methods for evaluating psycholinguistic systems is comparing the results with human judgement. Among three standard datasets that exist in the domain of measuring semantic relatedness between pairs of words, WordSimilarity-353 is the most comprehensive dataset since this dataset includes all of the 30 nouns of the Miller and Charles dataset and most of the 65 pairs of Rubenstein and Goodenough testset [Rubenstein and Goodenough, 1965]. The WordSimilarity-353 dataset contains 353 pairs of words compared by human agents in terms of similarity [Finkelstein *et al.*, 2002]. This system has been evaluated by both WordSimilarity-353 and Rubenstein and Goodenough testsets. While the Charles and Miller testset is not available on the web and is already included in WordSimilarity-353, we are not able to give statistics about the performance of proposed method on this dataset.

Wikipedia is a resource of concepts linked to each other forming a network, which is collaboratively constructed by human agents around the world. Each page in Wikipedia is linked to a set of categories classifying concepts and Wikipedia pages. Each page in Wikipedia describes one of the concepts associated to a word. A Wikipedia page describes a concept using other concepts described in other pages. In order to evaluate the proposed model to estimate the similarity between two entities, we used Wikipedia as a resource of knowledge and Wikipedia Categories as a taxonomy of concepts with which Wikipedia pages are annotated. The existing links in a Wikipedia page are considered as features describing the associated concept. Figure 4 illustrates the Wikipedia link structure data model. In this figure, each circle represents a wikipedia category and each square represents a Wikipedia page corresponding to a concept. Solid lines represent links between pages and a dotted line represents a link between a page and categories it belongs to.

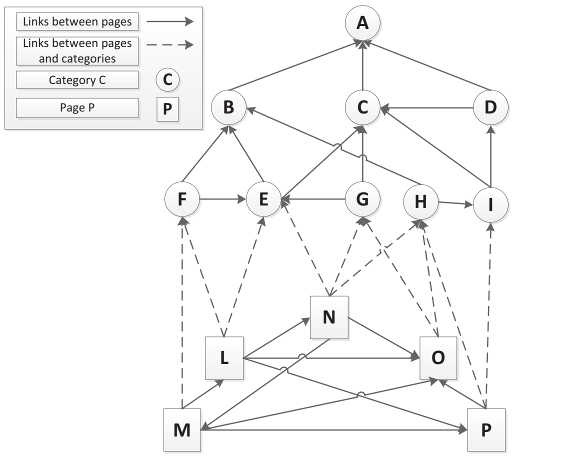


Figure 4: The Wikipedia link structure data model

In this experiment, The VSM is compared to the multi-tree model described previously against human judgement in terms of accuracy and correlation. For this purpose, each word is mapped to a Wikipedia page, and then a vector containing the links of the pages to which the corresponding page is linked is constructed and is referred to as a *link vector*. In Figure 4, the page L is linked to $\{N, O, P\}$ which are considered as features of the link vector $C_L = (N, O, P)$. Each link in the link vector represents another page in Wikipedia which is linked to Wikipedia categories. In the next step, another vector is constructed according to the categories of a link vector's elements (i.e., leaf nodes in Categories) called a first order category vector. In Figure 4, $\{N, O, P\}$ are linked to categories $\{E, G, H, I\}$. Then, the categories of the main page, L, (i.e., $\{E, F\}$) are added to the first order category vector and then the multi-tree representing the concept L is constructed from the first order category vector ($C_L^{1st} = (E, F, G, H, I)$). The rest of the process for construction of the multi-tree model is the same as described in Sections 3 and 4. This process is pictorially represented in Figure 5.

In VSM, the link vectors are compared using cosine similarity as described in equation 1. Both VSM schemes have been evaluated against human judgement with the same dataset and the results are compared in Table 1. Since, in this paper, we are not using a corpus-based approach, and in each experiment only two vectors representing two concepts are compared, the inverse document frequency of each link can not be calculated. Therefore, tf was used instead of tf-idf to implement the second VSM scheme.

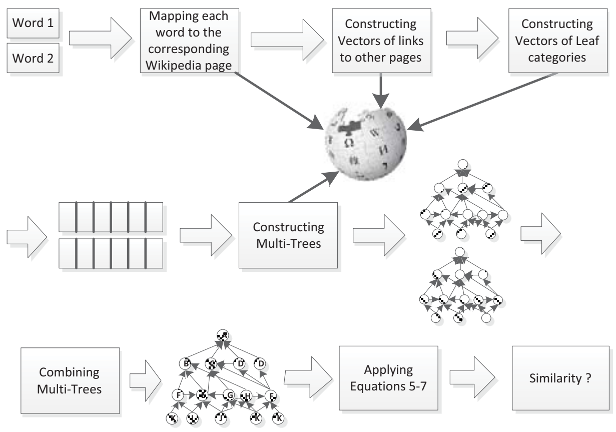

Word 1
Word 2
Mapping each word to the corresponding Wikipedia page
Constructing Vectors of links to other pages
Constructing Vectors of Leaf categories
Constructing Multi-Trees
Combining Multi-Trees
Applying Equations 5-7
Similarity ?

Figure 5: The Architecture of the proposed system

The average accuracy in Table 1 is measured regarding the difference between the value of similarity measured by the techniques tested above and that of human judgement.

$Accuracy = 1 - Average(error_i)$
$error_i = Sim_{Human}(w_{i1}, w_{i2}) - Sim_{Computer}(w_{i1}, w_{i2})$

The correlation was estimated by Spearman's rank correlation coefficient. The correlation between human judgement and the multi-tree model is demonstrated in Figure 6. Table 1 demonstrates that the proposed model achieved better results than both VSM schemes in terms of accuracy and correlation with human judgment in estimating similarity between entities.

6 Discussion, Conclusions and Future Work

In this paper we observed that techniques such as linear VSM ignore the semantic relationships among features. VSM calculates the similarity between two documents regarding the commonality of their features. However, in some cases, two documents may not be equal but may refer to the same entity. This limits the capability of a VSM to retrieve related

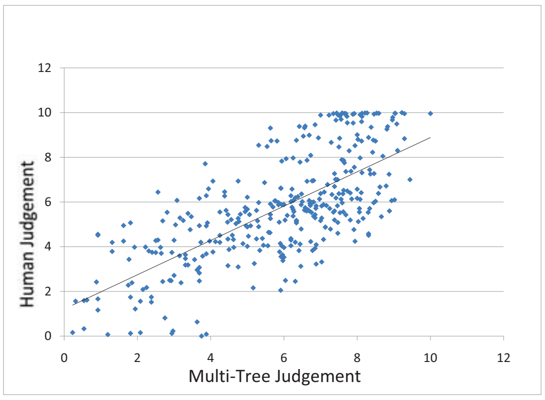


Figure 6: Human Judgement vs. Multi-Tree similarity algorithm on WordSimilarity-353 dataset

	Average Accuracy compared to human judgement	Correlation with Human judgement
WordSim-353		
VSM Boolean	57.1	54
VSM Frequency of terms	59.2	56.9
Multi-Tree	84.7	70.6
Rubenstein and Goodenough		
VSM Boolean	61.9	57.1
VSM Frequency of terms	62.2	60
Multi-Tree	80	74

Table 1: The comparison between VSM and Multi-Tree model using two testsets.

documents. The multi-tree model compensates for the lack of semantic relatedness among features using taxonomic relations that exist among the features of two entities. In this model the similarity weight is propagated from leaf nodes to the root of the multi-tree. The multi-tree model was evaluated by using the WordSimilarity-353 and Rubenstein and Goodenough datasets against human judgement and the results show that the multi-tree method outperforms VSM in terms of correlation and the average accuracy against human judgement for similarity of pairs of words. The results for WikiRelate and WLM, two previous systems which used Wikipedia Categories or WordNet to perform the task of similarity between two words using the same dataset, are shown in Table 2. WikiRelate and WLM were briefly described in Section 2.

The results in Table 2 show that the method proposed in this paper outperforms two of the other competitors namely WikiRelate and WLM by more than 20% and 3% respectively. However, ESA is the first ranked system using machine learning techniques as the basis of a semantic interpreter that is part of a system that maps fragments of natural language text into a weighted sequence of Wikipedia concepts ordered by their relevance to the input. Regarding ESA, it is clear that augmenting the text representation and constructing a weighted list of concepts provides benefits that link analysis alone does not completely replace.

Dataset	WikiRelate	ESA	WLM	Multi-Tree
Goodenough	52	82	64	74
WordSim-353	49	75	69	71
W-average	49	76	68	71

Table 2: Performance of semantic relatedness measures for two standard datasets for three popular systems vs. Multi-tree model.

Another potential model extension is in using a more sophisticated function such as Pearsons product-moment correlation or cosine similarity instead of the simple average function in equation 6.

A potential application of this model is in recommender systems, which concentrate on similarity between two products or two people. Referring once again to the example described in Section 3 and shown graphically in Figures 1, 2 and 3, the similarity of buying patterns can be established using a semantic multi-tree approach. For the evaluation of such systems, we need to construct a handcrafted taxonomy of products plus annotation of the product dataset to the taxonomy of products. Keyword search engines are also another potential application of such systems instead of linear VSM.

References

[Adomavicius and Tuzhilin, 2005] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, pages 734–749, 2005.

[Annates, 2005] U. Annates. Semantic tree method-historical perspective and applications Izabela Bondecka-Krzykowska. *Annales Universitatis Mariae Curie-Skłodowska: Informatica*, page 15, 2005.

[Finkelstein *et al.*, 2002] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin. Placing search in context: The concept revisited. *ACM Transactions on Information Systems (TOIS)*, 20(1):116–131, 2002.

[Gabrilovich and Markovitch, 2007] E. Gabrilovich and S. Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 6–12, 2007.

[Landauer *et al.*, 1998] T.K. Landauer, P.W. Foltz, and D. Laham. An introduction to latent semantic analysis. *Discourse processes*, 25(2):259–284, 1998.

[Larkey and Markman, 2005] L.B. Larkey and A.B. Markman. Processes of similarity judgment. *Cognitive Science: A Multidisciplinary Journal*, 29(6):1061–1076, 2005.

[Lee *et al.*, 2008] W.N. Lee, N. Shah, K. Sundlass, and M. Musen. Comparison of ontology-based semantic-similarity measures. In *AMIA Annual Symposium Proceedings*, volume 2008, page 384. American Medical Informatics Association, 2008.

[Maguitman *et al.*, 2005] A.G. Maguitman, F. Menczer, H. Roinestad, and A. Vespignani. Algorithmic detection of semantic similarity. In *Proceedings of the 14th international conference on World Wide Web*, pages 107–116. ACM, 2005.

[Pedersen *et al.*, 2005] S.P.T. Pedersen, S. Banerjee, and S. Patwardhan. Maximizing semantic relatedness to perform word sense disambiguation, 2005. *University of Minnesota Supercomputing Institute Research Report UMSI*, 25, 2005.

[Pesquita *et al.*, 2009] C. Pesquita, D. Faria, A.O. Falcão, P. Lord, and F.M. Couto. Semantic similarity in biomedical ontologies. *PLoS Comput Biol*, 5(7):e1000443, 2009.

[Rodríguez and Egenhofer, 2003] M.A. Rodríguez and M.J. Egenhofer. Determining semantic similarity among entity classes from different ontologies. *IEEE transactions on knowledge and data engineering*, pages 442–456, 2003.

[Rubenstein and Goodenough, 1965] H. Rubenstein and J.B. Goodenough. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633, 1965.

[Ruiz-Casado *et al.*, 2005] M. Ruiz-Casado, E. Alfonseca, and P. Castells. Automatic assignment of wikipedia encyclopedic entries to wordnet synsets. *Advances in Web Intelligence*, pages 380–386, 2005.

[Salton and McGill, 1983] G. Salton and M.J. McGill. Introduction to modern information retrieval. *New York*, 1983.

[Salton *et al.*, 1975] G. Salton, A. Wong, and C.S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[Schickel-Zuber and Faltings, 2007] V. Schickel-Zuber and B. Faltings. Oss: a semantic similarity function based on hierarchical ontologies. In *Proceedings of the 20th international joint conference on Artifical intelligence*, pages 551–556. Morgan Kaufmann Publishers Inc., 2007.

[Strube and Ponzetto, 2006] M. Strube and S.P. Ponzetto. WikiRelate! Computing semantic relatedness using Wikipedia. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, page 1419. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.

[Tversky and Shafir, 2004] A. Tversky and E. Shafir. *Preference, belief, and similarity: selected writings*. The MIT Press, 2004.

[Witten and Milne, 2008] I.H. Witten and D. Milne. An effective, low-cost measure of semantic relatedness obtained from Wikipedia links. In *Proceeding of AAAI Workshop on Wikipedia and Artificial Intelligence: an Evolving Synergy, AAAI Press, Chicago, USA*, pages 25–30, 2008.

Keyword-Based TV Program Recommendation

Christian Wartena, Wout Slakhorst, Martin Wibbels, Zeno Gantner, Christoph Freudenthaler, Chris Newell, Lars Schmidt-Thieme
Novay, Enschede, the Netherlands; {christian.wartena,wout.slakhorst,martin.wibbels}@novay.nl
University of Hildesheim, Hildesheim, Germany {zeno.gantner,freudenthaler,schmidt-thieme}@ismll.de
BBC R&D, London, UK, chris.newell@rd.bbc.co.uk

Abstract

Notwithstanding the success of collaborative filtering algorithms for item recommendation there are still situations in which there is a need for content-based recommendation, especially in new-item scenarios, e.g. in streaming broadcasting. Since video content is hard to analyze we use documents describing the videos to compute item similarities. We do not use the descriptions directly, but use their keywords as an intermediate level of representation. We argue that a nearest-neighbor approach relying on unrestricted keywords deserves a special definition of similarity that also takes word similarities into account. We define such a similarity measure as a divergence measure of smoothed keyword distributions. The smoothing is done on the basis of co-occurrence probabilities of the present keywords. Thus co-occurrence similarity of words is also taken into account. We have evaluated keyboard-based recommendations with a dataset collected by the BBC and on a subset of the MovieLens dataset augmented with plot descriptions from IMDB. Our main conclusions are (1) that keyword-based rating predictions can be very effective for some types of items, and (2) that rating predictions are significantly better if we do not only take into account the overlap of keywords between two documents, but also the mutual similarities between keywords.

1 Introduction

Notwithstanding the success of collaborative filtering algorithms for item recommendation there is still situations in which there is a need for content-based recommendation, especially in new-item scenarios, e.g. in streaming broadcasting. Since video content is hard to analyze we use context documents to compute item similarities. We do not use the documents directly, but use keywords as an intermediate level of representation. The representation by keywords has the advantage that the two tasks of text analysis and recommendation are clearly separated. Moreover, this offers the possibility to integrate information from different sources, including human classification and allows correction of faulty analyses, which might be important for many organizations.

Content-based recommendation relies on the ability to compute similarities between items based on their content. Classical methods use the overlap of words (either keywords are all words in the documents/descriptions), expressed by a correlation coefficient, like the Jaccard coefficient, or by the cosine similarity, to define the similarity between items. However, two items might have very similar content but use a different vocabulary to describe it. If we restrict the description of an item to a few keywords, the problem will become even more severe. Especially when keywords are not restricted to a set of standardized terms, it might be the case that two items have a considerable overlap in content but are described by completely disjoint sets of keywords. Thus we expect that recommendations could be improved if we are able to include keyword similarities in the definition of item similarities.

We compute similarities between keywords by comparing their co-occurrence distributions. For words in texts it is a well studied phenomenon that semantic and syntactic similarities can be computed by comparing the contexts in which they appear. Stated in other words: appearing in a similar context is a better indication for similarity than direct co-occurrence. For keywords we expect the same behavior since they are extracted from the (rather short) texts. In each text one synonym of a word is likely to be dominant and selected as a keyword. In other documents different synonyms of the keyword will appear in similar contexts.

Since we can use the same collection of keyword annotated items as we use for recommendation, the keyword-to-keyword similarities can be integrated easily into the item-item similarities. We consider a Markov chain on items and keywords, with transitions from items to keywords, representing the probabilities of terms to be a keyword for a given item and transitions from keywords to items, representing the probabilities for each document to be annotated with a given tag. Now the co-occurrence distribution of a keyword is obtained by a two-step Markov chain evolution starting with a keyword. Keyword similarities are determined by comparing their co-occurrence distributions. Item similarities are obtained by comparing the keyword distribution that arises from a one-step Markov chain evolution. By a three-step evolution starting with a document we incorporate the co-occurrence distributions of the keywords into a kind of smoothed keyword distribution of the item. When these smoothed distributions are compared, the co-occurrence similarity of keywords is included in the item-item similarity.

We have evaluated recommendations based on the keywords with a dataset collected by the BBC and with viewing data from MovieLens combined with plot descriptions from IMDB. For the BBC dataset we have the original editorial synopsis and a collection of related web pages. From both sets of texts we have extracted keywords by two different methods. For all set of keywords in the BBC dataset we see a clear improvement of recommendation results when keyword similarities are included in the computation of item-item similarities. Moreover, we see that keyword-based recommendation gives very good results, comparable or slightly better than those obtained by state-of-the-art collaborative filtering recommenders. Further observations from the experiments with this dataset are that the keywords extracted using a co-occurrence-based technique introduced in [20] give better results than the keywords extracted on the basis of their tf.idf value and that the related websites give rise to better keywords than the original descriptions.

In contrast to the BBC data, for the MovieLens dataset keyword-based recommendation is not able to predict useful ratings at all. This might be explained by the fact that keywords try to define the topic of an item. In a homogeneous database of movies it is likely that topic is not a key factor determining the users appreciation of the movie.

Our main conclusions are that it matters how the keywords are extracted and which texts are used and in the second place that the similarity measure is very important: recommendation results are significantly better if we do not only take into account the overlap of keywords between two documents, but also the mutual similarities between keywords.

2 Related Work

2.1 Co-occurrence-Based Similarity

The idea that words can be described in terms of the context in which they appear and hence the idea that word similarities can be derived by comparing these contexts has a long tradition in linguistics and is stated e.g. by Zelig Harris [5]. The concept has become known as the distributional hypothesis. Various formalizations of the idea differ considerably in the way a context of a word is defined. Co-occurrence distributions arise from approaches that do not use grammatical structure. Schütze and Pederson [16] suggest that one could construct a vector of co-occurrence probabilities from a complete word co-occurrence matrix, where co-occurrences are counted in a fixed size window. The cosine similarity of these vectors then provides a similarity measure. However, they do not pursue this approach because it was computationally too expensive. The approach that is most similar to the approach we will use is that of Linden and Piitulainen [10], who take all words in any dependency relation to the word under consideration as its context. Then the probability distribution over the words in the context is computed. Finally, the Jensen-Shannon divergence is used to compare these distributions.

This approach is very much the same as the query language models used in pseudo-relevance methods in information retrieval as formulated e.g. by [8] and [21]. In these approaches first, all documents containing the query term are retrieved. Then the average distribution of words in the documents is computed which in this approach is called the query language model. Finally, documents are ranked according to the similarity of the document distribution to the query language model.

2.2 Keyword Extraction

Extracting keywords from a text is closely related to ranking words in the text by their relevance for the text. To a first approximation, the best keywords are the most relevant words in the text. Determining the right weight structure for words in a text is a central area of research since the late 1960's ([15]). In 1972 Spärck Jones (reprinted as [17]) proposed a weighting for specificity of a term that has become known as *tf.idf.* This measure is still dominant in determining the relevance of potential keywords for a text. However, keywords are not simply the most specific words of a text and other factors may also play a role in keyword selection. Frank et al. [4] and Turney [19] and subsequently many others have used machine learning approaches to keyword extraction to integrate other features.

The relevance measure used below was introduced by Wartena et al. [20] and it was shown there that this measure gives good results for keyword extraction.

2.3 Keyword-Based Recommendation

As noted e.g. by [2] popular collaborative filtering algorithms are not suited for TV Program Recommendation, as the new item problem here is very prevalent. For new items content based recommendation has to be used. In content based recommendation approaches it is common to base recommendations on the words found in textual descriptions of the items. Here usually tf.idf weights or information gain is used ([12]) to determine the relevance of words. Words with low weights are usually removed, but still a relatively large number of words (100 or more [12]) is used for representation of the text. Furthermore, not all highly relevant words usually can serve as keywords that often are required to be noun phrases. Thus this approach differs significantly from a keyword-based approach.

Recently, there is a considerable interest in using social tags for recommendation. Tags are in many respects similar to keywords, but also have a lot of different characteristics. In most tagged collections the assigners of the tags are the same people that we want to compute recommendations for. Thus most approaches try to capture the tagging behavior of users to improve recommendations. One of the first papers that integrates tag-based similarities in a nearest-neighbors recommender is by Tso-Sutter et al. [18]. Liang et al. [9] also use a nearest-neighbor approach for tag-based recommendation. Most other approaches like the one of Firan et al. [3] build user profiles from tags and base recommendations on these profiles.

3 Markov Chains on Items and (Key)words

We use the distributions of terms over items for two different purposes: first we consider the distribution of all terms occurring in the texts to select a few key terms to represent each document. In a second stage we consider the distribution of keywords over items. We have to keep in mind that we talk about different sets of terms in both cases. The concepts and techniques used are however the same.

Consider a set of n term occurrences (words) $\mathcal{W}$ each being an instance of a term t in $\mathcal{T} = \{t_1, \dots t_m\}$, and each occurring in a source document d in a corpus $\mathcal{C} = \{d_1, \dots d_M\}$. Let $n(d,t)$ be the number of occurrences of term t in d, $n(t) = \sum_d n(d,t)$ be the number of occurrences of term t, $N(d) = \sum_t n(d,t)$ the number of term occurrences in d and n the total number of term occurrences in the entire collection.

We define three (conditional) probability distributions

$$\begin{aligned} q(t) &= n(t)/n && \text{on } \mathcal{T} && (1)\\ Q(d|t) &= n(d,t)/n(t) && \text{on } \mathcal{C} && (2)\\ q(t|d) &= n(d,t)/N(d) && \text{on } \mathcal{T}. && (3) \end{aligned}$$

Probability distributions on $\mathcal{C}$ and $\mathcal{T}$ will be denoted by P, p with various sub- and superscripts.

Consider a Markov chain on $\mathcal{T} \cup \mathcal{C}$ having transitions $\mathcal{T} \to \mathcal{C}$ with transition probabilities $Q(d|t)$ and transitions $\mathcal{C} \to \mathcal{T}$ with transition probabilities $q(t|d)$ only. Given a term distribution $p(t)$ we compute the one-step Markov chain evolution. This gives us a document distribution $P_p(d)$:

$$P_p(d) = \sum_t Q(d|t)p(t). \qquad (4)$$

Likewise given a document distribution $P(d)$, the one-step Markov chain evolution is the term distribution

$$p_P(t) = \sum_d q(t|d)P(d). \qquad (5)$$

Since $P(d)$ gives the probability to find a term occurrence in document d, p_P is the weighted average of the term distributions in the documents. Combining these, i.e. running the Markov chain twice, every term distribution gives rise to a new term distribution

$$\bar{p}(t) = p_{P_p}(t) = \sum_{t',d} q(t|d)Q(d|t')p(t'). \qquad (6)$$

For some term z, starting from the degenerate term distribution $p_z(t) = p(t|z) = \delta_{tz}$ (1 if $t = z$ and 0 otherwise), we get the *distribution of co-occurring terms* or *co-occurrence distribution* $\bar{p}_z$

$$\bar{p}_z(t) = \sum_{d,t'} q(t|d)Q(d|t')p_z(t') = \sum_d q(t|d)Q(d|z). \qquad (7)$$

This distribution is the weighted average of the term distributions of documents containing z where the weight is the probability $Q(d|z)$ that an instance of term z has source d. If we compute term similarities by comparing their co-occurrence distribution (rather than the source distributions $Q(d|z)$) we base the similarity on the context in which a word occurs as intended in the distributional hypothesis.

Likewise we obtain a term distribution if we run a Markov chain three times starting from the degenerated document distribution $P_d(i) = P(i|d) = \delta_{id}$:

$$\bar{P}_d(t) = p_{P_{p_{P_d}}}(t) = \sum_{d',t',d''} q(t|d')Q(d'|t')q(t'|d'')P(d''|d) \qquad (8)$$

$$= \sum_{d',t'} q(t|d')Q(d'|t')q(t'|d) = \sum_z q(z|d)\bar{p}_z(t). \qquad (9)$$

The distribution $\bar{P}_d$ can be seen as a smoothed version of the document distribution P_d in which co-occurrence information of the words is integrated. Thus, if we compare documents using these smoothed distributions we also take into account co-occurrence-based word similarities.

4 Keyword Extraction

For all items in our datasets a short textual description is available. We extract words from these texts to represent them as a vector in a word space. We can either use all words (after removing stop words) or only a small selection.

For keyword extraction we compare two different extraction methods. Both methods are based on ranking words and selecting the top n ranked words. The first method uses standard tf.idf ranking. The tf.idf value of a term t in a document d is defined as

$$tf.idf(t,d) = n(d,t)\frac{1}{\log df(t)}, \qquad (10)$$

where $n(d,t)$ is the number of occurrences of w in d, and df is the number of documents d' for which $n(d',t) > 0$.

The second method uses the hypothesis that the co-occurrence distribution of a good keyword is a good estimator of the term distribution of the document. Thus the suitability of a word as a keyword can be predicted by comparing the co-occurrence distribution of the word and the term distribution. There are various options to compute the similarity between two distributions. In [20] it was shown that the following correlation coefficient gives the best results:

$$r(z,d) = \frac{\sum_t(\bar{Q}_d(t) - q(t))(\bar{p}_z(t) - q(t))}{\sqrt{\sum_t(\bar{Q}_d(t) - q(t))^2}\sqrt{\sum_t(\bar{p}_z(t) - q(t))^2}}. \tag{11}$$

This coefficient captures the idea that two distributions are similar if they diverge in the same way from the background distribution q. The coefficient is in fact the cosine of the residual co-occurrence distribution of the term and the smoothed term distribution of the document after subtracting the background term distribution. Note that the "residual" probabilities can be negative and hence $r(z,d)$ also can become negative. For keyword extraction we will not only use the coefficient for ranking, but we will also require that the correlation coefficient defined in 11 is positive.

The different keyword extraction strategies are implemented in a UIMA[1] text analysis pipeline. All words in the text are stemmed using the tagger/lemmatizer from [6] and annotated by the Stanford part of speech tagger ([1]). To compute co-occurrence distributions all open class words are taken into account.

5 Keyword-Based Recommendation

The recommendation strategy we use is a straightforward nearest-neighbor approach for recommendation ([13]). Content-based nearest-neighbor approaches are similar to classical nearest-neighbor or collaborative filtering algorithms, but the similarity measure between items is based on the content of the items and not on the ratings. The rating we predict for a user and an item is the weighted average of all items rated by the user, where more similar items get greater weights. To be precise, let I_u be the set of all items rated by user u, then the predicted rating $R(u,i)$ of u for item i is defined by

$$R(u,i) = \frac{\Sigma_{j\in I_u}\mathrm{sim}(i,j)R(u,j)}{\Sigma_{j\in I_u}\mathrm{sim}(i,j)}. \tag{12}$$

We use two different keyword based similarity measures for items. The first measure is the Jaccard coefficient:

$$\mathrm{sim}(i,j) = \alpha + \frac{|K_i \cap K_j|}{|K_i \cup K_j|}, \tag{13}$$

where K_i is the set of keywords of item i. The additional parameter α ensures that each item is taken into account, even if the set of keywords is disjoint from the item for which a rating has to be predicted. Thus, items which do not overlap with any other items rated by the user get the user average as the prediction. If a very large value is taken for α, the predicted rating will always be the user average. Some initial experiments suggest that a value of about 0.1 yields the best results.

[1]`http://incubator.apache.org/uima/`

Table 1: Characteristics of the two datasets

	BBC Data	MovieLens subset
Items	2 487	704
Users	84 581	4 805
Ratings	130 262	361 961
Viewings	820 295	–

Since all keywords are drawn from an unrestricted vocabulary it might be the case that two texts are tagged with similar or strongly related words but not with exactly the same words. Thus we should not only check whether the same keywords are used, but also how strongly the keywords are related. As argued before, this can be done by comparing co-occurrence distributions: the co-occurrence distribution can be seen as a proxy for the semantics of a word. The whole text now has to be represented by the average of all co-occurrence distributions of all its keywords. This new distribution is in fact a smoothed version of the original keyword distribution of the document. The similarity between two items i and j is now given by

$$\mathrm{sim}(i,j) = \alpha + 1 - \mathrm{JSD}(\bar{p}_i|\bar{p}_j). \tag{14}$$

Again we use $\alpha = 0.1$, and JSD is the Jensen-Shannon divergence.

6 Evaluation

6.1 Data Sets

BBC Broadcast Data

As a first dataset to test our hypothesis that kNN-based rating prediction will benefit from including co-occurrence into the computation of item similarity was collected in a user study at the BBC. BBC programming provides a very interesting use case for keyword based recommendation. Since the BBC does not have a static database of items, like the movie databases on which much of the research on recommendation was done, but a stream of items. Here in fact each item that we want to predict ratings for is a new item. Content-based recommendation might be very useful in this situation. For all items an editorial description and one or more web pages are available.

The BBC data was collected during field trials of the MyMedia project[2] concerning recommender systems. An audience research panel was asked to rate all content items they watched during the field trial. In parallel, media server logs were analyzed to determine the viewing behavior of a larger superset of users. The characteristics of the dataset are described in Table 1.

Every content item in the BBC dataset has a related web page or website. This meant that two descriptions were available for each item:

[2]`http://www.mymediaproject.org`

Table 2: Number of unique keywords and average keywords per item

	Unique KWs		KWs per item	
	tf.idf	co-occ.	tf.idf	co-occ.
BBC Original descr.	7 136	5 631	9.39	8.44
BBC Web descr.	3 950	2 770	10.00	9.98
IMDB Plots	6 651	4 827	10.00	10.00
IMDB Keywords		14 177		73,23

1. Original editorial descriptions typically 30 to 200 words in length.
2. Website text typically 200 to 4000 words in length.

The website text was obtained automatically using some knowledge about the rough HTML structure of the web sites. Note that some content items have very brief descriptions and a simple, single web page associated with them whereas other items have longer descriptions and a substantial website. Where items were part of an ongoing series the web site frequently includes information about the complete series, rather than information about an individual episode.

We have extracted keywords from all texts by stemming and the two weighting schemes discussed above. Since we only extract nouns and verbs as keywords and we also exclude person names, as far as properly identified, less than ten keywords were found for a number of items. For all texts that are long enough 10 keywords were extracted. When extracting keywords using the correlation defined in 11 we also restrict the set of possible keywords to those term that have a positive correlation. Thus the number of keywords extracted here sometimes is lower than 10 even if 10 nouns are present in the text. The average number of keywords assigned and the total number of unique keywords used are given in Table 2.

MovieLens Dataset

The second dataset we have used is derived from the 10 Million rating dataset from MovieLens ([11]). We have augmented this dataset with the plot descriptions of the movies from IMDB ([7]). For a lot of movies the available plots are very short and uninformative. Thus we restricted the dataset to the movies having plots of at least 200 words. The characteristics of the dataset are described in Table 1. The number of keywords per item and the total number of unique keywords are given in Table 2.

As compared to the BBC dataset we see that the dataset is much denser: the number of users and items is smaller whereas there are many more ratings.

6.2 Experimental Setup

The goal of the experiment is twofold. First we want to know whether extracted keywords provide a viable resource on which to base recommendations. In the second place we want to test whether the similarity measure defined in (14) gives better rating predictions than the Jaccard coefficient (13). To test the latter hypothesis for each set of keywords we compute predictions using both measures. In order to test the first hypothesis we compare the keyword-based rating predictions to predictions from other algorithms. We use the following baselines:

1. Item average.
2. User average.
3. Collaborative filtering.
4. Genre- and series-based.

Item average (i.e. for a user-item pair we predict the average rating other users have assigned to that item) provides a nice baseline in the experiment but is not an alternative to content-based recommendations in real scenarios, since it cannot be applied for new items. User average (i.e. for a item user pair we predict the average rating the user has given to other items) also is a good baseline but not useful in real life since it does not help a user to make any choices. Collaborative filtering provides a very strong baseline and is some sense gives the limit we want to reach. However, it is only applicable in the static experiment and not in the streaming broadcast scenario as discussed above. For collaborative filtering we have used a state-of-the-art matrix factorization implementation.[3] For the genre-based recommendation we use the same algorithm as for the keyword-based recommendation. To do so we simply treat the genre labels as keywords. In the experiment with the BBC dataset there are a lot of series. We expect that series-based recommendation might give very good results, since it is likely that someone who likes some episodes of a series will also like the remaining episodes. Series can easily be identified, since in almost all cases all items of a series have the same title. By using the title of each item as a keyword we get a series-based recommender. Since we use $\alpha = 1$ for all items that do not belong to a series already rated by the user we predict the user average. Given the good results of genre-based recommendation in earlier experiments we also use genres and the combination of genres and title for content-based recommendation.

For evaluation we have done a leave-one-out experiment: each rating is predicted using all ratings except the one that has to be predicted. Since the recommender does not need any training of a model (except the co-occurrence distributions of the keywords) this is a very feasible approach. For the collaborative filtering we use a different protocol, since for each split a new model has to be trained. The result given here is obtained using a 10-fold cross-validation. Interpreting the results requires some caution because the matrix factorization models were trained using roughly 10 % smaller datasets.

[3] Biased matrix factorization from the MyMediaLite package: `http://ismll.de/mymedialite` [14]

Table 3: Results of content-based recommendation on BBC dataset

Data	Distance	RMSE
Web – tf.idf	Jaccard	0.302
Web – co-occ	Jaccard	0.290
Original – tf.idf	Jaccard	0.335
Original – co-occ	Jaccard	0.329
Genres	Jaccard	0.312
Title	Jaccard	0.287
Genres + title	Jaccard	0.293
Web – tf.idf	JSD	0.285
Web – co-occ	JSD	0.283
Original – tf.idf	JSD	0.332
Original – co-occ	JSD	0.312
Genres	JSD	0.339
Genres + Title	JSD	0.285
User average		0.348
Item average		0.464
MF		0.291

Table 4: Results of content-based recommendation on MovieLens/IMDB dataset

Data	Distance	RMSE
Plot - tf.idf	Jaccard	0,414
Plot - co-occ	Jaccard	0,412
Original Keywords	Jaccard	0,409
Genres	Jaccard	0,406
Plot - tf.idf	JSD	0,414
Plot - co-occ	JSD	0,413
Original Keywords	JSD	0,413
Genres	JSD	0,410
User average		0,415

6.3 Results

As an evaluation measure we use the Root Mean Square Error (RMSE) as is common for rating prediction. The results, in terms of RMSE are given in Table 3 and Table 4 for the BBC and MovieLens datasets, respectively.

The first remarkable fact is that keyword-based rating prediction gives very good results on the BBC dataset but cannot improve on the item average baseline in the case of the MovieLens/IMDB data. This result is not very surprising. Keywords mainly give the topic of the program or the movie plot. Whether someone likes a movie might depend on the genre, the director, the actors, etc. but probably not on the topic of the plot. Nevertheless we see that keyword-based recommendation indeed can be very useful since it clearly outperforms simple baselines like user or item average. As expected the series (title) and genre-based recommenders perform very well. However, the best keyword-based recommenders perform equally well. Surprisingly, the content-based recommenders perform equal well as the matrix factorization. The conclusion for our first hypothesis therefore is that keyword-based recommendation can be very useful for a dataset in which the topic of the item matters and for which no other suitable metadata, such as genre or series information is available.

With regard to our second question, whether the inclusion of keyword co-occurrence information in the definition of item similarity is useful, we see that in almost all cases our new distance measure gives better results than the standard measure. Only the genre-based results are poorer. We have however to say that the measure was not intended for use with such clearly defined concepts such as genres. It should solve problems with (near) synonyms in a set of freely selected keywords.

Furthermore we observe that the co-occurrence-based keywords perform better than tf.idf-based keywords. Thus the results also provide more evidence to support the conclusions of a comparison between the two methods in previous work ([20]). Finally, we see that the keywords extracted from the related material perform better than the keywords extracted from the original descriptions. When we look into more detail, on the contrary one gets the impression that the keywords extracted from the original descriptions contain less mistakes and noise. However, the main effect seems to be, that there are a lot of items for which the original descriptions are too short and give too few keywords.

7 Conclusion

In this paper we have investigated keyword-based rating prediction. Keywords constitute a useful level of description of an item since keywords can be assigned by humans or extracted automatically from one or more texts. We have shown that for some datasets keyword-based rating predictions give very good results, comparable to state-of-the art collaborative filtering methods. We have hypothesized that the reason lies in the nature of the dataset and the relevance of the topic of the item for the appreciation of the item. It remains a question for future research to apply keyword-based rating prediction to more datasets to verify this hypothesis.

We have argued that a nearest-neighbor approach relying on unrestricted keywords deserves a special definition of nearness taking word similarities also into account. We have defined such a similarity measure as a divergence measure of smoothed keyword distributions where the smoothing is done on the basis of the co-occurrence probabilities of the keywords. In the experiments we see that for various sets of keywords this measure always gives better results than the Jaccard coefficient.

Other findings are that the keywords extracted from the related web pages lead to better recommendation results than the keywords extracted from the original abstracts. The main reason seems to be that the abstracts are in many cases too short to extract an optimal number of relevant keywords. Finally we see that the keywords obtained by comparison of co-occurrence distributions lead to better recommendation results than the keywords extracted using a standard tf.idf relevance measure.

8 Acknowledgments

This work was funded by the European Commission FP7 project MyMedia[4] under the grant agreement no. 215006. We thank the anonymous reviewers for their valuable feedback.

References

[1] Stanford part of speech tagger. http://nlp.stanford.edu/software/tagger.shtml.

[2] P. Cotter and B. Smyth. Ptv: Intelligent personalised tv guides. In *AAAI/IAAI*, pages 957–964. AAAI Press / The MIT Press, 2000.

[3] C. S. Firan, W. Nejdl, and R. Paiu. The benefit of using tag-based profiles. In V. A. F. Almeida and R. A. Baeza-Yates, editors, *LA-WEB*, pages 32–41. IEEE Computer Society, 2007.

[4] E. Frank, G. W. Paynter, I. H. Witten, C. Gutwin, and C. G. Nevill-Manning. Domain-specific keyphrase extraction. In T. Dean, editor, *IJCAI*, pages 668–673. Morgan Kaufmann, 1999.

[5] Z. Harris. Distributional structure. *Word*, 10(23):146–162, 1954.

[6] M. Hepple. Independence and commitment: Assumptions for rapid training and execution of rule-based pos taggers. In *ACL*, 2000.

[7] http://www.imdb.com.

[8] J. D. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In W. B. Croft, D. J. Harper, D. H. Kraft, and J. Zobel, editors, *SIGIR*, pages 111–119. ACM, 2001.

[9] H. Liang, Y. Xu, Y. Li, R. Nayak, and L.-T. Weng. Personalized recommender systems integrating social tags and item taxonomy. In *Web Intelligence*, pages 540–547. IEEE, 2009.

[10] K. Lindén and J. Piitulainen. Discovering synonyms and other related words. *CompuTerm 2004*, pages 63–70, 2004.

[11] http://www.grouplens.org/system/files/-README_10M100K.html.

[12] M. J. Pazzani. A framework for collaborative, content-based and demographic filtering. *Artif. Intell. Rev.*, 13(5-6):393–408, 1999.

[13] M. J. Pazzani and D. Billsus. Content-based recommendation systems. In *The Adaptive Web: Methods and strategies of web personalization. Volume 4321 oF Lecture Notes in Computer Science*, pages 325–341. Springer-Verlag, 2007.

[14] S. Rendle and L. Schmidt-Thieme. Online-updating regularized kernel matrix factorization models for large-scale recommender systems. In *RecSys '08: Proceedings of the 2008 ACM Conference on Recommender Systems*. ACM, 2008.

[15] G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. Technical report, Cornell University, 1987.

[16] H. Schütze and J. Pederson. A cooccurrence-based thesaurus and two applications to information retrieval. In *Proceedings of RIA Conference*, pages 266–274, 1994.

[17] K. Spärck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 60:493–502, 2004.

[18] K. H. L. Tso-Sutter, L. Balby Marinho, and L. Schmidt-Thieme. Tag-aware recommender systems by fusion of collaborative filtering algorithms. In R. L. Wainwright and H. Haddad, editors, *SAC*, pages 1995–1999. ACM, 2008.

[19] P. D. Turney. Learning algorithms for keyphrase extraction. *Inf. Retr.*, 2(4):303–336, 2000.

[20] C. Wartena, R. Brussee, and W. Slakhorst. Keyword extraction using word co-occurrence. In *DEXA Workshops*, pages 54–58. IEEE Computer Society, 2010.

[21] C. Zhai and J. D. Lafferty. Model-based feedback in the language modeling approach to information retrieval. In *CIKM*, pages 403–410. ACM, 2001.

[4] `http://www.mymediaproject.org/`

A topology-based approach for followees recommendation in Twitter

Marcelo G. Armentano, Daniela L. Godoy and Analía A. Amandi
Universidad Nacional del Centro de la Provincia de Buenos Aires
CONICET, Consejo Nacional de Investigaciones Científicas y Técnicas
Argentina
{marmenta,dgodoy,amandi}@exa.unicen.edu.ar

Abstract

Nowadays, more and more users keep up with news through information streams coming from real-time micro-blogging activity offered by services such as Twitter. In these sites, information is shared via a followers/followees social network structure in which a follower will receive all the micro-blogs from the users he follows, named followees. Recent research efforts on understanding micro-blogging as a novel form of communication and news spreading medium have identified different categories of users in Twitter: information sources, information seekers and friends. Users acting as information sources are characterized for having a larger number of followers than followees, information seekers subscribe to this kind of users but rarely post tweets and, finally, friends are users exhibiting reciprocal relationships. With information seekers being an important portion of registered users in the system, finding relevant and reliable sources becomes essential. To address this problem, we propose a followee recommender system based on an algorithm that explores the topology of followers/followees network of Twitter considering different factors that allow us to identify users as good information sources. Experimental evaluation conducted with a group of users is reported, demonstrating the potential of the approach.

1 Introduction

Micro-blogging activity taking place in sites such as Twitter is becoming every day more important as real-time information source and news spreading medium. In the followers/followees social structure defined in Twitter a follower will receive all the micro-blogs from the users he follows, known as followees, even though they do not necessarily follow him back. In turn, re-tweeting allows users to spread information beyond the followers of the user that post the tweet in the first place

Studies conducted to understand Twitter usage [Java *et al.*, 2007; Krishnamurthy *et al.*, 2008] revealed that few users maintain reciprocal relationships with other users, which can be regarded as friends or acquaintances, while most of them behave either as information sources or information seekers. Users behaving as information sources tend to collect a large amount of followers as they are actually posting useful information or news. In turn, information seekers follow several users to obtain the information they are looking for and rarely post any tweet themselves.

Finding high quality sources among the expanding micro-blogging community using Twitter becomes essential for information seekers in order to cope with information overload. In this paper we present a topology-based followee recommendation algorithm aiming at identifying potentially interesting users to follow in the Twitter network. This algorithm explores the graph of connections starting at the target user (the user to whom we wish to recommend previously unknown followees), selects a set of candidate users to recommend and ranks them according to a scoring function that favors those users exhibiting the distinctive behavior of information sources.

Unlike other works that focus on ranking users according to their influence in the entire network [Weng *et al.*, 2010; Yamaguchi *et al.*, 2010], the algorithm we propose explores the follower/following relationships of the user up to a certain level, so that more personalized factors are considered in the selection of candidates for recommendation, such as the number of friends in common with the target user. Since only the topology of the social structure is used but not the content of tweets, this algorithm also differs from works exploiting user-generated content in Twitter to filter information streams [Chen *et al.*, 2010; Phelan *et al.*, 2009; Esparza *et al.*, 2010] or to extract topic-based preferences for recommendation [Hannon *et al.*, 2010].

The rest of this paper is organized as follows. Section 2 reviews related research in the area. Section 3 describes our approach to the problem of followee recommendation in Twitter. In Section 4 we present the experiments we performed to validate our proposal and in Section 5 we present and discuss the results obtained and in Section 6 we compared our results with a related approach. Finally, in Section 7, we discuss some aspects of our proposal and present our conclusions.

2 Related Work

The problem of helping users to find and to connect with people on-line to take advantage of their friend relationships has been studied in the context of traditional human social networks. For example, SONAR [Guy *et al.*, 2009] recommends related people in the context of enterprises by aggregating information about relationships as reflected in different sources within an organization, such as organizational chart relationships, co-authorship of papers, patents, projects and others. Chen et al. [Chen *et al.*, 2009] compared relationship-based and content-based algorithms in making people recommendations, finding that the first ones are better at finding known contacts whereas the second ones are stronger at discovering new friends. Weighted minimum-message ratio (WMR) [Lo and Lin, 2006] is a graph-based algorithm which generates a personalized list of friends in social network build according to the observed interaction among members. Unlike these algorithms that gathered social networks in enclosed domains, mainly starting from structured data (such as interactions, co-authorship relations, etc.), we propose a people recommendation algorithms that take advantage of Twitter social structure populated by massive, unstructured and user-generated content.

Understanding micro-blogging as a novel form of communication and news spreading medium has been one of the primary concerns of recent research efforts. Kwak *et al.* [2010] analyzed the topological characteristics of Twitter and its power for information sharing, finding some divergences between this follower/followees network and traditional human social networks: follower distribution exhibit a non-power-law (users have more followers than predicted by power-law), the degree of separation is shorter than expected and there is a low reciprocity (most followers in Twitter do not follow their followers back). Other works addressed the problem of detecting influential users as a method of ranking people for recommendation. In the previous study it was found that ranking users by the number of followers and by PageRank give similar results. However, ranking users by the number of re-tweets indicates a gap between influence inferred from the number of followers and that inferred from the popularity of user tweets. Coincidently, a comparison of in-degree, re-tweets and mentions as influence indicators carried out in [Cha *et al.*, 2010] concluded that the first is more related to user popularity, whereas influence is gained only through a concentrated effort in spawning re-tweets and mentions and can be hold over a variety of topics. TwitterRank [Weng *et al.*, 2010] tries to find influential twitterers by taking into account the topical similarity between users as well as the link structure, TU-Rank [Yamaguchi *et al.*, 2010] considers the social graph and the actual tweet flow and Garcia and Amatriain [2010] propose a method to weight popularity and activity of links for ranking users.

The influence rankings presented by studies on the complete Twittersphere have not direct utility for followee recommendation since people get connected for multiple reasons. We demonstrated with our experiments that indegree, which has proven to be a good representation of a user's influence in Twitter using only its topology (see for example [Kwak et al., 2010]) gives the worst results for followee recommendation since people that are popular in Twitter would not necessarily match a particular user interests (if a user follows accounts talking about technology, he/she would not be interest in Ashton Kutcher, one of the most influential Twitter accounts according to Kwak et al. [2010])

Recommendation technologies applied to Twitter have mainly focused on taking advantage of the massive amount of user-generated content as a novel source of preference and profiling information [Chen *et al.*, 2010, Phelan *et al.*, 2009, Esparza *et al.*, 2010]. In contrast, we concentrate in recommending interesting people to follow. In this direction, Sun *et al.* [2009] proposes a diffusion-based micro-blogging recommendation framework which identifies a small number of users playing the role of news reporters and recommends them to information seekers during emergency events. Closest to our work are the algorithms for recommending followees in Twitter evaluated and compared using a subset of users in [Hannon *et al.*, 2010]. Multiple profiling strategies were considered according to how users are represented in a content-based approach (by their own tweets, by the tweets of their followees, by the tweets of their followers, by the combination of the three), a collaborative filtering approach (by the IDs of their followees, by the IDs of their followers or a combination of the two) and two hybrid algorithms. User profiles are indexed and recommendations generated using a search engine, receiving a ranked-list of relevant Twitter users based on a target user profile or a specific set of query terms. Our work differs from this approach in that we do not require indexing profiles from Twitter users; instead a topology-based algorithm explored the follower/followee network in order to find candidate users to recommend.

The main difference between existent work and our work is that the mentioned approaches for followee recommendations, except for the approach presented in [Hannon *et al.*, 2010], were evaluated using datasets gathered from Twitter, with no assessment about the target user interest in the recommendations. In other words, the target user interest in a followee recommended that is not in the current list of the target user's followees cannot be assessed within these datasets in order to determinate the correctness of the recommendation. For this reason, the approach proposed in this work was evaluated with a controlled experiment with real users.

3 Followees Recommendations on Twitter

The algorithm we propose for recommending followees on Twitter consists in two steps: (1) we explore the target user's neighborhood in search of candidates and (2) we rank candidates according to different weighting features. These steps are detailed in Sections 3.1 and 3.2, respectively.

3.1 Finding candidates

The general idea of the algorithm we implemented is to suggest users that are in the neighborhood of the target user, where the neighborhood of a user is determined from the follower/followee relations in the social network.

In order to find candidate followees to recommend to a target user U, we based our search algorithm on the following hypothesis: The users followed by the followers of U's followees are possible candidates to recommend to U. In other words, if a user F follows a user that is also followed by U, then other people followed by F can be interesting to U.

The rationale behind this hypothesis is that the target user is an information seeker that has already identified some interesting users acting as information sources, which are his/her current followees. Other people that also follows some of the users in this group (i.e. is subscribe to some of the same information sources) have interests in common with the target user and might have discover other relevant information sources in the same topics, which are in turn their followees.

This scheme is outlined in Figure 1 and can be resumed in the following steps:

1. Starting with the target user, we first obtain the list of users he/she follows, let's call this list S.
$$S = \bigcup_{s \in followees(U)} s$$
2. From each element in S we get its followers, let's call the union of all these lists L
$$L = \bigcup_{s \in S} followers(s)$$
3. Finally, from each element in L, we get its followees to obtain the list of possible candidates to recommend. Let's call the union of all these lists T.
$$T = \bigcup_{l \in L} followees(l)$$
4. Exclude from T those users that the target user already follows. Let's call the resulting list R.
$$R = T - S$$

Each element in R is a possible user to recommend to the target user. Notice that each element can appear more than once in R, depending on the number of times that each user appears in the followees or followers lists obtained at steps 2 and 3 above.

3.2 Weighting features

Once we find the list R of candidate recommendations for the target user, we explored different features to give a score to each unique user $x \in R$.

The first feature explored is the relation between the number of followers a user has with respect to the number of users the given user follows, as shown in Equation 1.

$$w_f(x) = \frac{|followers(x)|}{|followees(x)|} \qquad (1)$$

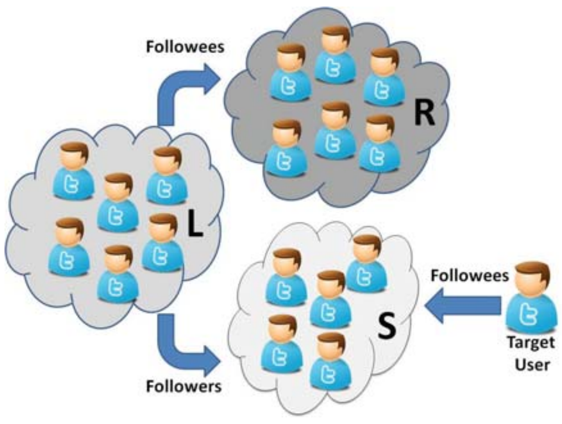


Figure 1: Scheme for finding candidate recommendations

Since we seek for sources of information to recommend, we assume that this kind of users will have a lot of followers and that they will follow few people. If user *x* has no followees, then only the number of followers is considered without changing the significance of the weighting feature.

We use this metric as a baseline for comparison with other metrics. Our aim is to demonstrate that metrics for ranking popular users on Twitter are not good for ranking recommendations of users that a target user might be interest in following. In [Kwak *et al*., 2010] it has been shown that the rankings of users that can be obtained by number of followers and by PageRank [Brin and Page, 1998] are very similar. We opted to use this factor as an estimator of the "importance" of a given user because the number of followers is a metric by far more easily to obtain that the user PageRank in a network with an order of almost 2 billion social relations.

The second feature explored corresponds to the number of occurrences of the candidate user in the final list of |R| candidates for recommendations, as shown in Equation 2.

$$w_o(x) = \frac{|\{i \in R / i = x\}|}{|R|} \qquad (2)$$

The number of occurrences of a given user *x* in this final list is an indicator of the amount of (indirect) neighbors that also have *x* as a (direct) connection itself.

The third feature we considered is the number of friends in common between the target user U and the candidate recommendation x:

$$w_c(x) = |followees(x) \cap followees(U)| \qquad (3)$$

Finally, we considered two combinations of these features: the average of the three features, and their product:

$$w_s(x) = \frac{1}{3}[w_o(x) + w_f(x) + w_c(x)] \qquad (4)$$

$$w_p(x) = w_o(x) * w_f(x) * w_c(x) \qquad (5)$$

It is worth noticing that the selection of these weighting features was not arbitrary. Our choice was based on a deep analysis of previous studies about Twitter and particular properties of this specific network that makes general link prediction approaches unsuitable. All the studies about the properties of the Twitter network agree in that there is a minimal overlap with the features available on other online social networks (OSNs).

4 Experiment setting

To evaluate the proposed algorithms, we have carried out a preliminary experiment using a group of 14 users. These users, 8 males and 6 females, were in the last years of their course of studies and were students of a Recommender Systems related course dictated at our University as an elective course during 2010. The students selected for the experiment were volunteers familiarized with Twitter.

During the first part of the course, we asked these users to create a Twitter account and to follow at least 20 Twitter users who publish information or news about a set of particular subjects of their interest. The general interests expressed by users ranged between diverse subjects such as technology, software, math, science, football, tennis, basket, religion, movies, journalists, government, music, cooking, shoes, TV programs and even other students in their faculty. Some users only concentrated on one particular subject while others distributed their followees among several topics.

Then, we used the user IDs of the user accounts created by the students as seeds to crawl a sub-graph of the Twitter network corresponding to three levels of both followee and follower relations, centered on each seed. The resulting dataset consisted on 1,443,111 Twitter users and 3,462,179 *following* relations already existing among them.

During the second part of the course, we provided these users with a desktop tool that allowed them to login to Twitter and ask for followees recommendations. Since the users who participated in the experiment were students of a "recommender systems" course, all of them had knowledge about concepts such as rankings and metrics. As part of a not compulsory practical exercise of the course they were motivated to discover which metric better ranked recommendation results and to write a brief report about the results they obtained for their particular case. The desktop application provided for this exercise allowed students to select the weighting feature by which they liked to rank recommendations, with no predefined order.

In all cases, 20 recommendations were presented to the users. Then, we asked the students to explicitly evaluate whether the recommendations were relevant or not according to the same topical criteria they have chosen to select their followees as information sources in the first place. For each recommendation in the resulting ranking the application showed the user name, description, profile picture and a link to the home page of the corresponding account. This link could be used to read the tweets published by the recommended user in the case that the information provided by the application was not enough to determine the student's interest in the recommendation. The question we asked students to ask themselves to determine whether a recommendation was relevant or not was "Would you have followed this recommended user in the first place (when selecting which users to follow in the first part of the experiment), if you had know this account?" For example, if a given student was interest in technology and he/she had not discovered the account @TechCrunch during his/her first selection of followees, that would be an interest recommendation because @TechCrunch tweets about news on technology.

5 Results

We first evaluated the performance of the proposed algorithm in terms of their overall precision in followees recommendation. *Precision* can be defined as the number of relevant recommendations over the number of recommendations presented to the user and it can be also computed at different positions in the ranking. For example, P@5 ("precision at five") is defined as the percentage of relevant recommendations among the first five, averaged over all runs. Figure 2 shows the precision achieved by the algorithm, averaged between all users, for each weighting feature at four different positions of the ranking: P@1, P@5, P@10 and P@20. The results of considering each feature separately and the two aggregations functions are showed in this figure.

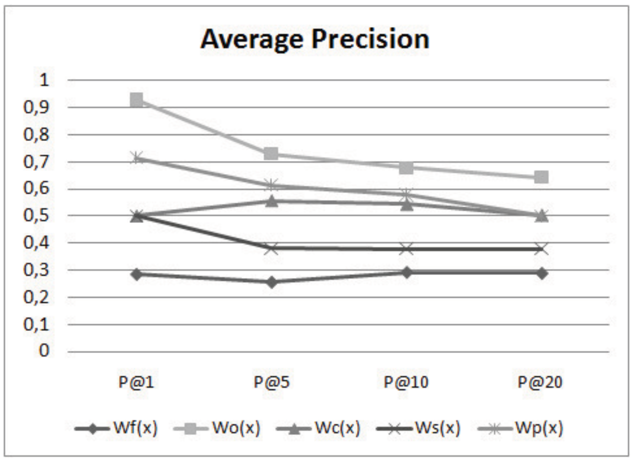


Figure 2: Average precision for each weighting feature

We can observe several interesting facts in the results presented in Figure 2. First, it results that $w_o(x)$, the weighting feature considering the number of occurrences of a user in the list of recommendations as gathered by the algorithm proposed, generates better precision scores than any other weighting feature explored. For this weighting feature we obtained a good recommendation in the first position of the ranking for 93% of the users. For longer ranking lists, precision decrease from 0.73 for P@5 to 0.64 for P@20, which we believe are all good results.

It is worth noticing that although we reported results up to P@20, recommendations lists tend to be shorter (frequently 5) in order to help the user to focus on the most relevant results. In these small lists the algorithm reached good levels of precision, recommending mostly relevant users.

The weighting feature considering the number of followers, $w_f(x)$, got the worst precision scores, with values under 30%. This fact reveals that this metric, although widely used in other approaches as mentioned in Section 2, is only good at measuring a user's general popularity in the entire Twitter network, but popularity does not necessarily translate into relevance for a particular user. Celebrities and politicians, such as Barack Obama (@barackobama), Lady Gaga (@ladygaga), Yoko Ono (@yokoono), and Tom Cruise (@tomcruise) were a common factor in the rankings of many users regardless their particular interests. Among other popular users suggested that in some cases met the user's interests were popular blogs and news media such as Mundo Geek (@mundo_geek), C5N (@C5N), El Pais, (@el_pais), Mundo Deportivo (@mundodeportivo), Red Hat News (@redhatnews) and Fox Sport LA (@foxsportslat).

A similar situation occurs with $w_c(x)$, the weighting feature considering the number of friends in common between the target user U and the candidate user to recommend to U. Although precision is better than $w_f(x)$ for every size of the recommendation lists, this weighting feature does not reach the performances obtained with $w_o(x)$. This result is expected since the fact that two users U and X share a friend Y does not necessarily means that X is a good information source.

We also found that $w_s(x)$ tends to perform poorly. This score is affected by the term corresponding to the relation between the number of followers and the number of followees, which in most cases is higher than the other terms involved. This factor highly affects the overall average among the three weighting features, causing a decrease in precision.

The second score which combines the three weighting features, $w_p(x)$, seems to overcome this problem since in this case each weighting feature is multiplied to obtain the final score. Nevertheless, celebrities and very popular Twitter user accounts also tend to appear at the top positions of the ranking diminishing the general precision again. However, the factor corresponding to $w_o(x)$ also makes good recommendations to appear interleaved with some popular users on Twitter.

Another interesting issue observed in the results presented in Figure 2 is that for both $w_f(x)$ and $w_c(x)$ precision tend to keep almost constant across different sizes in the list of recommendations and even with a slightly increment as the size of the recommendation set increases. This fact seems to contradict the definition of precision in the information retrieval sense which, by principle, should decrease as the number of recommendations increases. However, this behavior occurs because all $w_f(x)$, $w_s(x)$ and $w_c(x)$ does not concentrate relevant recommendations in the top positions of the ranking. On contrary, we can observe that for $w_o(x)$ and $w_p(x)$ relevant recommendations tend to be clustered towards the top of the ranking.

Although precision measure gives a general idea of the overall performance of the presented weighting features, it is also very important to consider the position of relevant recommendations in the ranking presented to the user. Since it is known that users focus their attention on items at the top of a list of recommendations [Joachims, 2005], if relevant recommendations appear at the top of the ranking using one algorithm and at the bottom of the ranking using the other, the first algorithm will be perceived as better performing by users even though their general precision might be similar.

Discounted cumulative gain (DCG) is a measure of effectiveness used to evaluate ranked lists of recommendations. DCG measures the usefulness, or gain, of a document based on its position in the result list using a graded relevance scale of documents in a list of recommendations. The gain is accumulated from the top of the result list to the bottom with the gain of each result discounted at lower ranks. The premise of DCG is that highly relevant documents appearing lower in a list should be penalized as the graded relevance value is reduced logarithmically proportional to the position of the result. The DCG accumulated at a particular rank position k is defined as shown in Equation 6:

$$DCG@k = rel_1 + \sum_{i=2}^{k} \frac{rel_i}{\log_2 i} \qquad (6)$$

DCG is often normalized using an ideal DCG vector that has value 1 at all ranks. Figure 3 shows the normalized DCG obtained for both algorithms at four different positions of the ranking: nDCG@1, nDCG@5, nDCG@10 and nDCG@20.

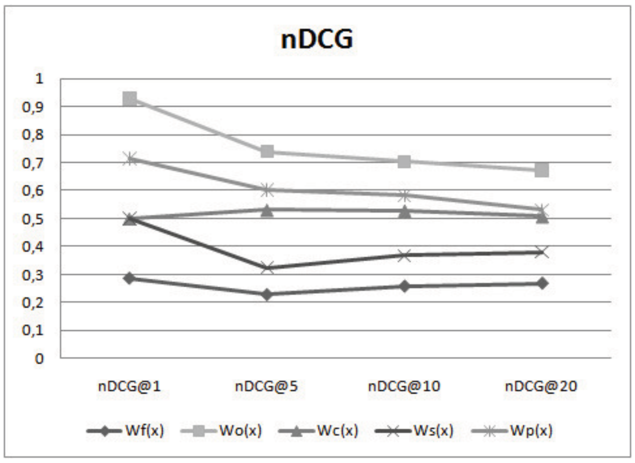


Figure 3: Normalized Discounted Cumulative Gain at rank k for each weighting feature

nDCG@1 is equivalent to P@1 by definition. Then, we can see that scoring users with $w_o(x)$ always positions relevant users above in the ranking than other weighting features, seconded by $w_p(x)$.

Success at rank k (S@k) is another metric commonly used for ranked lists of recommendations. The success at rank k is defined as the probability of finding a good recommendation among the top k recommended users. In other words, S@k is the percentage of runs in which there was at least one relevant user among the first k recommended users. Figure 4 shows the results we obtained for this metric with values of k ranging from 1 to 10.

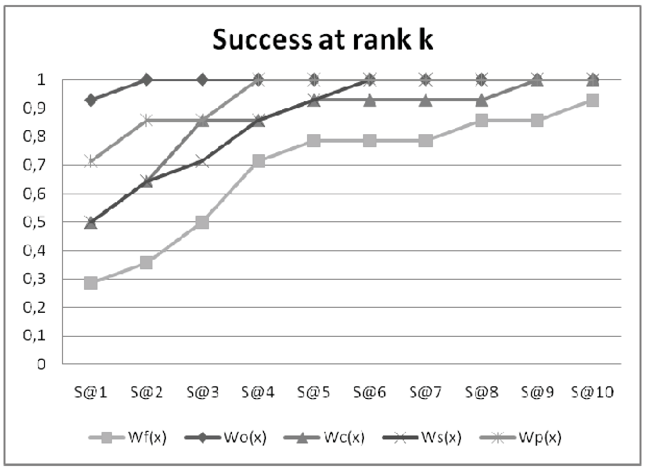


Figure 4: Success at rank k for each weighting feature

For S@k we can observe results equivalent to nDCG@k. Again, scoring users with $w_o(x)$ always positions relevant users above in the ranking than the other weighting features. The ranking according $w_p(x)$ allowed users to find a relevant recommendation always at the most at position 4 in the ranking, while for $w_s(x)$ we obtain success 1 at position 6. With this metric we can confirm that $w_f(x)$ and $w_c(x)$ are not good weighting factors by their own.

To study further the algorithm ability to rank followees for recommendation, we used *Mean Reciprocal Rank* (MRR), a metric that measures where in the ranking is the first relevant recommendation. If the first relevant recommendation is at rank r, then the MRR is 1/r. This measure averaged over all runs provides insight in the ability of the system to recommend a relevant user to follow in Twitter at the top of the ranking. Figure 5 plots the MMR measure for both proposed algorithms.

This metric gives us another view of which weighting feature generates better ranking of recommendations. We confirm that $w_o(x)$ always ranks users better than the other proposed weighting features, while ranking users by their “popularity” does not generate good recommendations.

The experiments presented make us believe that there is reason to be optimistic about the potential for a followee recommender for Twitter using the method described in Section 3.1 to obtain a list of candidates and simple ranking them by the number of occurrences of each candidate in the list generated by this method. Among the advantages of this method when compared with content-based alternatives is that recommendations can be found quickly based on a simple analysis of the network structure, without considering the content of the tweets posted by the candidate user. Nevertheless, we also believe that combining the proposed method with an analysis of the content of the tweets posted by a user in the list of candidates can improve the precision of a followee recommender system, at the expense of computational performance.

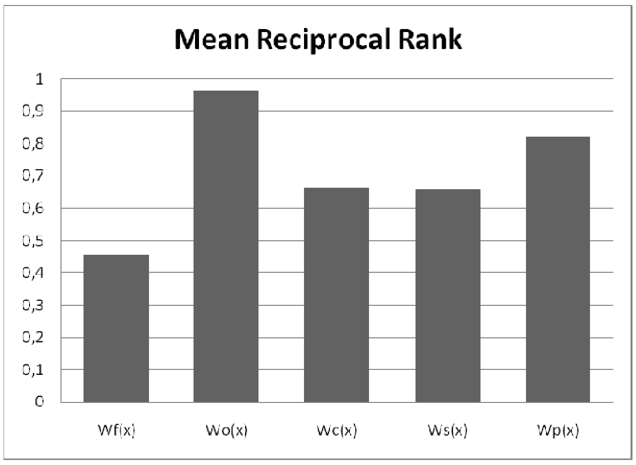


Figure 5: Mean Reciprocal Rank for each weighting feature

6 Comparison with related work

From the related work, the approach that we find more similar to ours (and the only one, up to our knowledge that experimented with real users in a controlled experiment) is *Twittomender*, proposed by Hannon et al. [2010]. Although the results presented in [Hannon *et al.*, 2010] are not fully comparable to the results presented in this article since different datasets were used, in this section we present a comparison about the precision reported for *Twittomender* and the precision obtained with our approach.

Twittomender create different indexes for all users in the dataset generated from different sources of profile information. Four of these indexes are content-based, modeling users by their own tweets, by the tweets of their followers, by the tweets of their followees and by a combination of the three. The three remaining strategies are topology-based and model users by the IDs of their followees, by the IDs of their followees and by a combination of both.

The strategy used for ranking users in the online experiment presented in [Hannon *et al.*, 2010] generates the seven rankings according to the different approaches described above and then generate a single ranking by merging those seven rankings. When merging the rankings they use a scoring function that is based on the position of each user in the recommendation lists. In this way users that are frequently present in high positions are preferred over users that are recommended less frequent or in lower positions.

Hannon et al. performed a live user trial with 34 users, reporting a precision of about 38.2% for k=5 and 33.8% for k=10. Table 1 summarizes the comparison between *Twittomender* and our system. Notice that precision values for *Twittomender* system are approximate because they were taken (and in some cases computed) from the graphics presented in the article.

It is worth noticing that although the number of volunteers who participated in *Twittomender* experiment is more than twice the number of volunteers who participated in our experiment, the number of Twitter users involved in our experiment is by far higher than the number of users in their database. Furthermore, *Twittomender* can only recommend users that are previously indexed. When a user is registered into the system, all his/her followees and followers profiles along with his/her own profile are indexed. Our work differs from this approach in that we do not require indexing profiles from Twitter users; instead a topology-based algorithm explores three levels of the follower/followee network in order to find candidate users to recommend.

7 Discussion and Conclusion

In this article we presented a simple but effective algorithm for recommending followees in the Twitter social network. This algorithm first explores the target user neighborhood in search of candidate recommendations and then sorts these candidates according to different weighting features: the relation between the number of followers and the number of followees, the number of occurrences of each candidate in the final list, the number of friends in common, and two combinations of the three features.

We evaluated the proposed algorithm with real users and we obtained satisfactory results in finding good followee recommendations. We found that considering just the overlapping users among the different lists of follower and followees explored by our crawling method gives better results than the other features considered. As expected, the indegree of a user is not a good feature for ranking followee recommendations. Considering the number of followers for ranking users put celebrities and popular Twitter accounts at the top of the list, but these recommendations are not necessarily interesting for a particular user. However, there are some interesting recommendations discovered by this feature, such as top bloggers who write about a particular subject or news media accounts.

Although the results reported seems promising, we are planning to repeat the experiment this year in order to involve more users in the experiment and obtain more statistical support for the results reported. Moreover, we are very optimistic about the potential improvements that we can obtain by extending the presented approach with content-based techniques. A natural extension of our approach in which we are currently working on is a hybrid algorithm that filters the candidate recommendations found with the topology-based method with a content-based analysis of the tweets posted by the users. In this new approach, a target user U is modeled with a vector of terms built from a content analysis of the tweets posted by U's followees. This vector is then compared with the vector of terms corresponding to each candidate recommendation and the similarity obtained is considered in the generation of the ranking.

The results reported in this article make us feel really enthusiastic about the potentials of Twitter for building recommender systems of sources of information.

	Twittomender "live-user" trial	Our approach
#seeds	34	14
#users	100,000	1,443,111
P@5	~38.2%	72.9%
P@10	~33.8%	67.9%
P@20	~26.9%	64.3%

Table 1: Comparative table between *Twittomender* and our approach

References

[Brin and Page, 1998] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*. Volume 30, Issue 1-7, pages 107-117. 1998.

[Cha *et al.*, 2010] M. Cha, H. Haddadi, F. Benevenuto, and K. Gummadi. Measuring user influence in Twitter: The million follower fallacy. *In Proceedings of the 4th International Conference on Weblogs and Social Media (ICWSM'10)*, Washington DC, USA, 2010.

[Chen *et al.*, 2009] J. Chen, W. Geyer, C. Dugan, M. Muller, and I. Guy. Make new friends, but keep the old: recommending people on social networking sites. *In Proceedings of the 27th International Conference on Human Factors in Computing Systems*, pages 201–210, 2009.

[Chen *et al.*, 2010] J. Chen, R. Nairn, L. Nelson, M. Bernstein, and E. Chi. Short and tweet: experiments on recommending content from information streams. *In Proceedings of the 28th International Conference on Human Factors in Computing Systems (CHI'10),* pages 1185–1194, 2010.

[Esparza *et al.*, 2010] S. Garcia Esparza, M. P. O'Mahony, and B. Smyth. On the real-time web as a source of recommendation knowledge. *In Proceedings of the 4th ACM Conference on Recommender Systems (RecSys'10)*, pages 305–308, 2010.

[Garcia and Amatriain, 2010] R. Garcia and X. Amatriain. Weighted content based methods for recommending connections in online social networks. *In Workshop on Recommender Systems and the Social Web*, pages 68–71, Barcelona, Spain, 2010.

[Guy *et al.*, 2009] I. Guy, I. Ronen, and E. Wilcox. Do you know?: recommending people to invite into your social network. *In Proceedings of the 13th International Conference on Intelligent User Interfaces (IUI'09)*, pages 77–86, 2009.

[Hannon *et al.*, 2010] J. Hannon, M. Bennett, and B. Smyth. Recommending Twitter users to follow using content and collaborative filtering approaches. *In Proceedings of the 4th ACM Conference on Recommender Systems (RecSys'10),* pages 199–206, 2010.

[Java *et al*., 2007] A. Java, X. Song, T. Finin, and B. Tseng. Why we twitter: understanding micrblogging usage and communities. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis (WebKDD/SNA-KDD '07).* ACM, New York, NY, USA, pages 56-65. 2007.

[Joachims *et al*., 2005] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '05).* ACM, New York, NY, USA, 154-161. 2005.

[Krishnamurthy *et al*., 2008] B. Krishnamurthy, P. Gill, and M. Arlitt. A few chirps about twitter. *In Proceedings of the first workshop on Online social networks (WOSP '08).* ACM, New York, NY, USA, pages 19-24. 2008.

[Kwak *et al.*, 2010] H. Kwak, C. Lee, H. Park, and S. Moon. What is Twitter, a social network or a news media? *In Proceedings of the 19th International Conference on World Wide Web (WWW'10)*, pages 591–600, 2010.

[Lo and Lin, 2006] S. Lo and C. Lin. WMR–A graph-based algorithm for friend recommendation. *In Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI'06)*, pages 121–128, Washington, DC, USA, 2006.

[Phelan *et al.*, 2009] O. Phelan, K. McCarthy, and B. Smyth. Using Twitter to recommend real-time topical news. *In Proceedings of the 3rd ACM Conference on Recommender Systems (RecSys'09),* pages 385–388, 2009.

[Sun *et al.*, 2009]A. R. Sun, J. Cheng, and D. D. Zeng. A novel recommendation framework for micro-blogging based on information diffusion. *In Proceedings of the 19th Workshop on Information Technologies and Systems,* 2009.

[Weng *et al.*, 2010] J. Weng, E-P. Lim, J. Jiang, and Q. He. TwitterRank: finding topic-sensitive influential twitterers. *In Proceedings of the 3rd ACM International Conference on Web Search and Data Mining (WSDM'10),* pages 261–270, New York, NY, USA, 2010.

[Yamaguchi *et al.*, 2010]Y. Yamaguchi, T. Takahashi, T. Amagasa, and H. Kitagawa. TURank: Twitter user ranking based on user-tweet graph analysis. *In Web Information Systems Engineering*, volume 6488 of LNCS, pages 240–253, Hong Kong, China, 2010.

Context-Aware Recommendation Based On Review Mining

Negar Hariri, Bamshad Mobasher, Robin Burke and Yong Zheng
DePaul University, College of Computing and Digital Media
243 S. Wabash Ave, Chicago, IL 60604, USA
{nhariri, mobasher, burke, yzheng8}@cs.depaul.edu

Abstract

Recommender systems are important building blocks in many of today's e-commerce applications including targeted advertising, personalized marketing and information retrieval. In recent years, the importance of contextual information has motivated many researchers to focus on designing systems that produce personalized recommendations in accordance with the available contextual information of users. Compared to the traditional systems that mainly utilize users' preference history, context-aware recommender systems provide more relevant results to users. We introduce a context-aware recommender system that obtains contextual information by mining user reviews and combining them with user rating history to compute a utility function over a set of items. An item utility is a measure that shows how much it is preferred according to user's current context. In our system, the context inference is modeled as a supervised topic-modeling problem in which a set of categories for a contextual attribute constitute the topic set. As an example application, we used our method to mine hidden contextual data from customers' reviews of hotels and use it to produce context-aware recommendations. Our evaluations suggest that our system can help produce better recommendations in comparison to a standard kNN recommender system.

1 Introduction

In recent years, recommender systems (RS) have been extensively used in various domains to recommend items of interest to users based on their profiles. A user's profile is a reflection of the user's previous selections and preferences that can be captured as rating scores given to different items in the system. Using preference data, different systems have been developed to produce personalized recommendations based on collaborative filtering, content-based or a hybrid approach.

Despite the broad usage of such recommender systems, failure to consider the users' current situations may result in considerable performance degradation in recommendations. For example, a customer who has once bought a toy for his friend's child may repeatedly receive suggestions to buy items related to kids as the recommendation algorithm decides based on the whole history in user's profile without prioritizing his current interests. To address this issue, the notion of context and context-aware recommender systems (CARS) has been introduced.

Contextual information can be explicit or implicit and can be inferred in different ways such as using GPS sensor data, clickstream analysis or monitoring user rating behavior. In this paper, we concentrate on deriving context from a textual description of a user's current state and the item features in which he/she is interested. This data can be in different forms such as tweets, blog posts, review texts or it can be given directly to the system as part of a query.

As an example application of our approach, we have used our method to mine hidden contextual data from customers' reviews of hotels. The reason behind the selection of this dataset is that users usually provide some contextual cues in their comments. For example, they may mention that they are with family or on a business trip, or they may express their opinions about the hotel services that are important to them such as having wireless internet, conference rooms, etc. In order to evaluate our method, we have used "trip Advisor" hotel reviews dataset where each review contains an overall rating, an optional review comment and also a "trip type" attribute that shows the types of trips user suggest for this hotel. For this attribute, the user can select a subset of five possible values: Family, Couples, Solo travel, Business, and Friends' getaway. The "trip type" attribute is not a feature of user or hotel (as different users may assign different values), it is rather related to the interaction and it is assumed to be an indication of context in our system.

Our approach in inferring context is based on using a classifier that is trained by the samples of descriptions and their corresponding contexts. Usually the trip type that a customer picks for a hotel is related to his review. Having this assumption, a set of review texts and their associated trip types are selected as the training set for the context classifier. After training, for a given description (as the user context) the classifier computes the probability of each the trip category. This probability distribution is used to infer context. Since we are dealing with a multi-class supervised classification problem, we chose to use Labeled-LDA [1] as our categorization method as based on our experiments it performs better in our dataset in comparison to other similar methods.

We propose a method to use this inferred context to produce context-aware recommendations. While most of the existing approaches assume that a user's rating behavior depends on the current context and predict a rating function, we differentiate between the "rating" that a user gives to an item and the "utility" gains from choosing it. The inferred context is used to define a utility function for the items reflecting how much each item is preferred by a user given his current context. More specifically, the utility value depends on two factors: the predicted rating and the "*context score*" where context score represents the suitability of an item for a user in a given context. Rating can be predicted based on any conventional recommendation algorithms such as kNN.

Through the rest of this paper, we will first review some of the related work. Section 3 describes our proposed context-aware recommendation process. Finally, section 4 includes the evaluation of the proposed method and its comparison with traditional recommender.

2 Related Work

Several researchers have previously investigated the use of contextual information in various applications of recommender systems. Although there is no clear-cut definition of context, one of the most commonly used definitions was suggested by Abowd et al. [2] as follows: "Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves." This is a general definition that limits the context only to the information that could be used to characterize the situation or the circumstance. Another similar definition by H.Lieberman et al. [3] is: "context can be considered to be everything that affects computation except the explicit input and output". In addition to these general definitions, a number of more specific definitions of context have been recently provided. For example, "Context can be described by a vector of context attributes, e.g. time, location or currently available network bandwidth in a mobile scenario". [4].

Capturing and representation of context in a system depends on the way context is defined in that system. Dourish et al. [5] presented two different views in modeling context: The representational view and the interactional view. In representational view, context is defined as *a form of information* that is *stable*, *delineable* and is *separate from activity*. Having this view, context can be defined and represented as a specific set of attributes of the environment within which the user's interaction with the system has taken place. For example, time and location can be considered as contextual attributes. In interactional view, it is assumed that *contextuality is a rational property* that holds between objects and activities rather than to be information (as in the case of representational view). Also, the contextual features are not definable and static but their *scope is defined dynamically*. Furthermore, rather than assuming that context defines the situation within which an activity occurs, it is assumed that *context arises from activity* and activity is induced by context. Therefore, even though context is not observable itself, the activity that arise from the context can be observed.

Adomavicius et al. [6] suggest three different architectures for context-aware recommender systems: In Contextual pre-filtering approach, the dataset is first filtered, the recommendations will be then provided based on the contextualized dataset. On the other hand, contextual post-filtering approach generates recommendations similar to traditional recommender systems. It will then filter and re-rank these recommendations to provide contextual recommendations. In contextual modeling, context is added to the problem as an additional dimension; meaning that in contrast to traditional recommender systems that estimate the rating function in two dimensional space of $User \times Item$, the context-aware recommender system is defined over the space of $User \times Item \times Context$. The representation of context and the way it should be captured and integrated into the recommendation algorithm depend on available contextual information as well as the definition of context in the system.

An interesting application of context-aware recommender systems is in mobile devices that are equipped with GPS or have internet access. In this case, different contextual information can be captured in real-time in order to be used in the recommendation process. For example PioApp Recommender [4] produces recommendations based on points of interest (such as restaurants, museum and train stations) in the neighborhood of the mobile user. Social camera introduced in [7] assists users in picking photo compositions given their current location and scene context. Many mobile travel applications such as [8–10] have also took advantage of context in order to make better suggestions. Numerous algorithms have also been suggested for music and movie recommendation (as well as many other domains). Micro-profiling introduced in [11], splits each single user profile into several possibly overlapping sub-profiles where each of them represent user's preference in a particular context. A context random walk algorithm was proposed in [12] to model the user's movie browsing behavior and then use it to make context-aware recommendations.

Some of the above mentioned approaches such as [8, 9] use a simple representational view of context where context is shown as a set of attributes (such as time, location, weather conditions) that is given to the system as input; while some other systems try to infer the contextual attributes from the user behavior. Instead of using a representational model, the context-aware recommender in [13] uses an interaction model. The proposed system was inspired by human memory model in psychology where the short term and long term memories are separately modeled. The short term memory contains the user preferences derived from his active interaction with the system while the long term memory stores the preference models related to his previous interactions with the system. They introduced three types of contextual cues including collaborative, semantic and behavioral cues in order to retrieve relevant preference models from the long term memory. The retrieved memory objects will then be combined with user's current preference model to generate and aggregate a final preference model that is used to produce recommendations.

In this paper we propose a method for mining contextual data from texual reviews. The importance of the hidden data in review comments has been the subject of many researches in the area of opinion mining and sentiment analysis. In opinion analysis, various Natural Language processing techniques and text analysis methods are applied to a set of review to extract attributes of the object that are referred to in the review text and to discover polarity (positive, negative or neutral) of the expressed opinions.

The problem of extracting contextual information from unstructured text is fairly new and has not been extensively addressed in prior researches. Aciar [14] introduces a method to identify review sentences which contain contextual information. In their approach, rule sets were created to classify sentences into contextual and preference categories where the preference category groups sentences including user's evaluation of the features. The approach presented in [14] does not discuss the use of the retrieved information in the recommendation process while we will provide a way of incorporating the contextual knowledge in producing the recommendations.

3 Context-Aware Recommendation Process

Our context-Aware recommender system (CARS), includes several components. The first component is the context miner that is responsible for determining a user's current context. Context is represented as a distribution function over the set of trip types and can be mined from a textual description of user's current situation and the features that are important to him. The main part of the context inference module consists of a multi-class supervised classifier. After training the classifier, context can be inferred for a given query. An example query is shown in Table 1. Based on the underlined words, it seems that the user is most probably looking for "couples" or "family" type trip rather than a "business" one.

I'm planning a <u>romantic</u> trip for my <u>anniversary</u>. I'm looking for an all inclusive resort <u>near a beach</u>. I expect the hotel room to be spacious, have a <u>nice view</u> over the sea and to be nicely <u>decorated</u>.

Table 1: A sample query

The second component of our system is the rating predictor that is a simple collaborative filtering recommender which predicts ratings of items. This component can be replaced with other types of rating prediction algorithms. The third component calculates the utility function based on user's current context and the predicted rating and presents a set of suggestions according to the order of the utility values.

3.1 Context Representation

Contextual recommender systems can either have an interactional or a representational view of the context. In this paper, we assume there are explicit labels representing context and the contextual information is obtained for each textual review by mapping it to this label set.

In our experiments, a dataset containing a set of hotel reviews from Trip Advisor website has been used. In this dataset, the "trip type" attribute assigned to a hotel review shows the types of trips that the user suggest for the hotel. The assigned attribute can be selected by the user from a set of five possible choices: Family, couples, Solo travel, business, and friends' getaway. We assume that this element is the representation of context in our system. A sample review from this dataset is depicted in Table 1. This sample shows the relationship between the trip type attribute and the review comment. For example "budget accommodation", "twin bedroom", "small" and "shared bathroom" can be more related to Friends getaway trip than to a business trip or a family travel.

Producing context aware recommendations requires mining the user's current context. If the user explicitly specifies his context, then it can be easily used in the recommendation algorithm. On the other hand, if he implies his context in a set of sentences describing his current state or his desired features for the hotel, then an inference method is required to determine the probability of each trip type. In this way, the context is shown as a distribution over the set of trip categories. In both cases, let $Context_u^i$ denote the context of user u when using item i. For example, if the reviewer u indicates the trip type for hotel i as business and solo travel, then the context represenation is $Context_u^i = \{P(\text{family}) = 0,$ $P(\text{couples}) = 0$, $P(\text{solo travel}) = 0.5$, $P(\text{business}) = 0.5$, $P(\text{friends' get away}) = 0\}$.

The context inference problem just described is similar to a multi-labeled text classification problem in which documents can be a classified into one or more categories. The general solution is to provide a training set, build the model and use the model to categorize the new documents. If the trip type categories assigned to each review is assumed to be related to the review comment (and we will show they are related in our dataset), then we can use a set of review comments and their corresponding trip type values as our training set for training the classifier.

Trip type	Friends' getaway
Review Comment	This is an excellent option for budget accommodation in a hostel type establishment in a top class location, very close to central station and quick bus journey to circular quay. Stayed in twin bedroom which was very small but did the trick. If all you want is a clean bed in a clean room then this is grand. Shared bathroom and showering facilities were kept clean too.
Summary Quote	Excellent hostel accommodation in great location

Table 2: A sample review comment and the associated trip type

3.2 Inferring the Context

Different techniques have been used in text categorization such as probabilistic methods, regression modeling and SVM

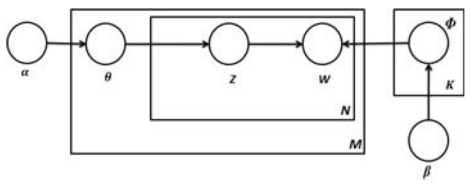


Figure 1: Graphical Representation of LDA [16]

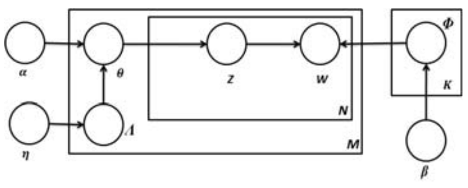


Figure 2: Graphical Representation of Labeled-LDA [1]

classification. In this article, we have used Labeled Latent Drichlet Allocation [1] (shown as L-LDA) for our dataset as it has shown to perform relatively well on our dataset. This method is a supervised classification algorithm for multi-labeled text corpora and is based on topic modeling.

Topic modeling and Labeled-LDA

Topic modeling deals with statistical modeling of documents in order to discover the latent topics behind them. Probabilistic latent semantic analysis (PLSA) [15] is one of the early approaches in this area which models a document as a probability distribution over the set of topics.

Later, the Latent Drichlet Allocation [16], known as LDA, was proposed as an extension of PLSA. LDA specifies a generative process for creating documents. The document generation is based on the idea that documents are mixture of topics, where a topic is a probability distribution over words. To generate a new document d, first the distribution over topics denoted by $\theta^{(d)}$ should be specified. For each word in the document a topic t is selected based on $\theta^{(d)}$. Let $\phi^{(t)}$ denote the multinomial distribution over words for topic t; According to this distribution a word is picked and is added to the document. It should be noted that this is similar to the general procedure followed by most of the existing topic models except that the statistical assumptions differs based on the model. The LDA model assumes that the topic mixture θ is a k-dimensional random variable as follows [16].

$$P(\theta|\alpha) = \frac{\Gamma(\sum_{i=1}^{k}\alpha_i)}{\prod_{i=1}^{k}\Gamma(\alpha_i)}\theta^{\alpha_1-1}\dots\theta_k^{\alpha_k-1} \quad (1)$$

Where α is k-vector with elements $\alpha_i > 0$ and $\Gamma(x)$ is the gamma function. Figure 1 describes the graphical representation of LDA where the rectangles show replicates. The outer rectangle represents M documents while the inner rectangle illustrates the process of sampling words for a document of size N. In the LDA model, the document size follows a Poisson distribution. In corpora with a large vocabulary, it is likely that some of the words do not appear in the training examples. In order to cope with problem, a smoothing strategy is used by placing a Drichlet prior on ϕ with parameter β as shown in the figure.

In our problem, the user reviews are assumed to be documents where the topics behind these documents are the set of possible values for a trip type. As the topics are predefined, we need to adopt a supervised topic modeling approach. Some variations of LDA have been proposed to support supervised learning such as [1, 17, 18] among which we chose to use Labeled-LDA [1] as the other methods limit each document to be associated with only one topic while in our case, reviews can have multiple labels. Similar to LDA, in Labeled-LDA modeling, each word in the document is assigned a single topic. However, in order to incorporate supervision, the topic should belong to the label set of the document. In other words, there is a one-to-one relationship between the set of labels assigned to the documents and the topics and the topic mixture of each document is formed according to its label set. Figure 2 shows the graphical presentation of Labeled-LDA. Having k unique labels in all documents, the parameter Λ for each document is a k dimensional binary vector that shows the presents or absence of each topic in the document label set. For each document, Λ is generated using a Bernoulli coin toss with a prior probability vector η

As in [1], we used Gibbs sampling [19] for training. Let C^{WT} and C^{DT} represent two matrices which contain word-topic counts and document-topic counts respectively. Gibbs sampling begins with randomly assigning words to topics and filling the two matrices accordingly. Then iteratively updates them to finally converge to estimations of θ and ϕ. At each iteration, a word token is selected and its current topic is removed and C^{WT} and C^{DT} are updated by decrementing the corresponding entries to the removed topic assignment. Then, a new topic is sampled based on the topic assignments to all other words and the count matrices are incremented accordingly. After convergence, estimates of θ and ϕ can be obtained using equations 2 and 3 respectively.

$$\theta_j^{(d)} = \frac{C_{dj}^{DT} + \alpha}{\sum_{k=1}^{T} C_{dk}^{DT} + T\alpha} \quad (2)$$

$$\phi_i^{(j)} = \frac{C_{ij}^{WT} + \beta}{\sum_{k=1}^{W} C_{kj}^{WT} + W\beta} \quad (3)$$

3.3 Predicting item utility

As noted earlier, we make a distinction between predicting *rating* and *utility*. We assume that the utility of an item for a user may differ among different contexts, even though the user has rated the same item equally in those contexts. For instance, in the hotel review dataset it is possible that the rating given by a customer to hotel on a business trip does not change if he visits the same hotel one more time with his family while the utility of selecting that hotel changes from one trip type to the other. When he is on a business trip, business services of the hotel are more important while in a family trip some other characteristics of the hotel (such as having a pool, distance to beach etc.) gain more priority.

We define *context score* as a measure of suitability of an item for a user in a given context. To calculate the context score for user u and item i, we need to predict the context that u would assign to i that is denoted by $predictedContext(u, i)$. The predicted context will then be compared to current context of u (that can be inferred). We use a collaborative approach for calculating context of a $(user, item)$ pair. The similarity between two items i and j is computed using the cosine similarity as follows:

$$\text{contexualSimilarity}(i, j) = \frac{\sum_u \text{commonLabels}(i, j)}{\sqrt{\sum_u |\text{labels}(i)| \times |\text{labels}(i)|}} \tag{4}$$

Where $\text{commonLabels}(i, j)$ is the number of times users assign the same trip type category to both i and j and $\text{labels}(i)$ counts the number of trip type labels given to i by all users. This similarity is used to obtain a neighborhood for item i by selecting the top N most similar items. Then, the predicted context can be computed as in equation 5. In the predicted context, the probability of each trip category is calculated by taking the weighted average of its probabilities in the neighbors' contexts.

$$\text{predictedContext}(u, i) = \frac{\sum_{k \in \text{Neighbors}(i)} \text{context}_u^k \cdot \text{contexualSimilarity}(k, i)}{\sum_{k \in \text{Neighbors}(i)} |\text{contextualSimilarity}(k, i)|} \tag{5}$$

Where context_u^k stands for the neighbor k context given by user u.

Our notion of predicted context for a $(user, item)$ pair is somehow similar to the idea of "best context" introduced in [20] for music recommendation. The authors have defined this concept as the contextual information most suited for a particular item. They have used a vector representation of context where each dimension corresponds to a contextual attribute. if the user believes that context is suitable for that specific item, the value of the corresponding dimension is set to one. They have proposed four different approaches for the prediction of the best context. The first method is based on averaging the context vectors of the item across all users. Another technique is to find the K-nearest neighbors of the user (based on rating history) and compute the predicted context as the weighted average of the contexts assigned to that item by his neighbor. The other two methods follow the same approach except that the similarity of users are computed based on the context vectors and independent of their rating history. Our method is different from the previously mentioned approaches in various aspects: The above methods focus on predicting the suitable context for a $(user, item)$ pair while we address the whole process of context-aware recommendation; In other words, predicting the best context is just a part of our context-aware algorithm. Moreover, our method for calculation of contextual similarity and also prediction of the best context is different from the previous techniques.

Context score of item i for user u can be estimated by comparing the distribution of inferred context of u and predicted context for this item. We used three different methods namely Chebyshev Similarity [21], Kullback-Leibler Similarity [22] and a simple cosine similarity. We have chosen to use cosine similarity in our evaluations as it performs better on our dataset.

Let IC_u denote the inferred context for user u and PC_u^i indicate the predicted context (calculated based on equation 5). The context Score for item i and user u is computed as follows:

$$\text{contextScore}(u, i) = \frac{IC_u \cdot PC_u^i}{||IC_u|| ||PC_u^i||} \tag{6}$$

The utility score of item i and user u is calculated as a function of both the context score of i and also the predicted rating of the item. In our experiments, standard item-based kNN was used to calculate the predicted ratings.

$$\begin{aligned} \text{utility}(i, j) = \alpha \cdot \text{predictedRating}(u, i) + \\ (1 - \alpha) \cdot \text{contextScore}(u, i) \end{aligned} \tag{7}$$

In relation 7, α is a constant representing the weight of the predicted rating in the utility function. The items are sorted based on utility values and the top N items are suggested to the user.

4 Evaluation

The evaluations presented in this paper were performed on the Trip Advisor dataset that contains 12558 reviews for 8941 hotels made by 1071 reviewers. About 9500 of the reviews has the "trip type" label which has been used as an indication of context.

Our system consists of two main parts and the experiments have been designed accordingly. The first experiment focuses on assessing the accuracy of the context inference module on our dataset. In the second experiment, the performance of the recommender system is compared with the standard kNN recommender.

4.1 Context Inference Evaluation

The accuracy of the context inference algorithm plays a significant role in the performance of the system. As previously explained, we used Labeled-LDA as it has shown to perform relatively better than other multi-Labeled text classification method. In this experiment we will assess its performance on our dataset. The experiment was set up as a five-fold cross validation. In each of the five runs, one of the folds was used for testing while the topic model was built based on the remaining four folds. For every test case (i.e. the review text), the probability distribution over the trip type categories were predicted. A category is assigned to a test case if the predicted probability for that category exceeds a certain threshold.

The results are evaluated by measuring both precision and recall where precision is computed as the fraction of correct categorical labels, and recall is computed as the ratio of correct labels to total number of labels. Figures 3 and 4 depict recall and precision values for different categories. As

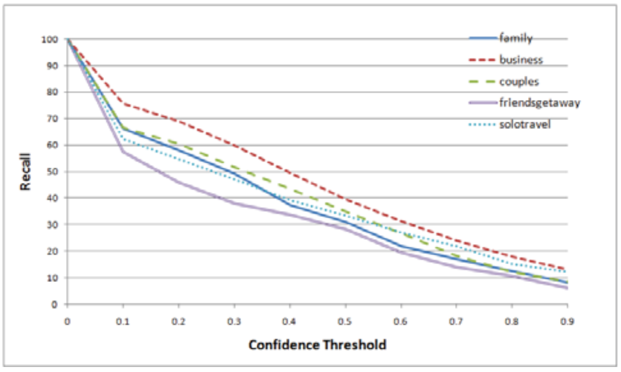


Figure 3: Recall values for different categories

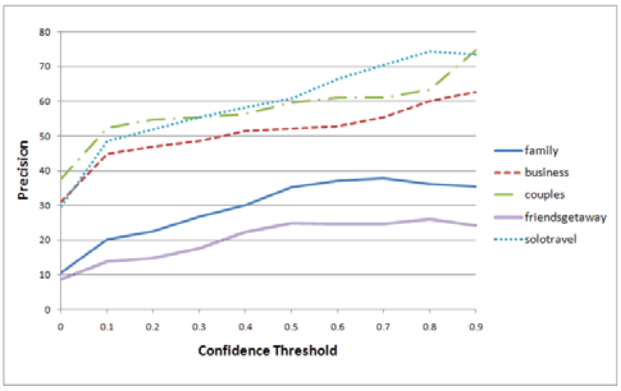


Figure 4: Precision values for different categories

it is shown, the precision tends to be higher as the threshold increases. Also, as expected, by increasing the confidence threshold, recall is likely to decrease.

4.2 Evaluation of Recommendations

As we are working with a sparse dataset, a preprocessing phase has been added to the procedure in order to prune the matrix by removing all those items that have less than 5 ratings.

In previous sections, we introduced a context-aware recommender that produce recommendations for a user based on a utility function that depends both the user's current context and also the predicted rating for that item. As recommendations are based on utility function (and not ratings alone), it is not logical to use metrics such as MAE and other metrics that compare the predicted rating with the actual ones. Instead, hit ratio was chosen as our performance measure and we performed a leave-one-out cross validation experiment on those reviews that have ratings greater than the reviewer's average rating. Having the recommendation size of k, the hit ratio is calculated as the probability that the left-out item is included in the list of N recommendations. The standard item-based kNN algorithm has also been run on the same dataset and under the same condition as our recommender method. Figure 5 shows the hit ratio having different sizes of recommendation list for standard kNN and context aware recommender system where the user's context is inferred and also when it is explicitly expressed. The results suggest that an increase in hit ratio is expected when the contextual information is involved in producing the recommendations.

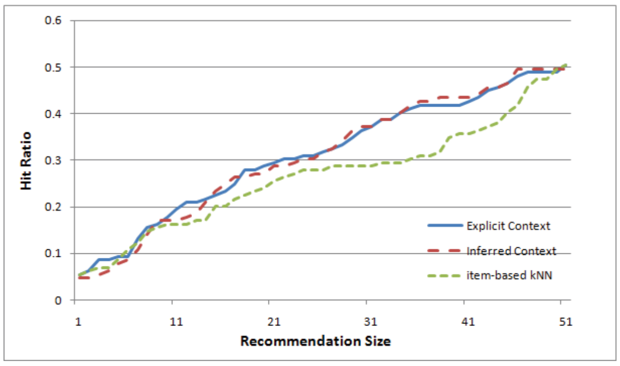


Figure 5: Hit Ratio comparison for item-based kNN and context-aware recommender

5 Conclusions

This paper has presented a novel approach for mining context from unstructured text and using it to produce context-aware recommendations. In our system, the context inference is modeled as a supervised topic-modeling problem for which we used Labeled-LDA to build the context classifier. The inferred context is used to define a utility function for the items reflecting how much each item is preferred by a user given his current context. The utility value for each item depends on two factors: the predicted rating and the "*context score*" where context score represents the suitability of the item for a user in a given context. Rating can be predicted based on any conventional recommendation algorithms such as kNN.

As an example application, we have used our method to mine hidden contextual data from customers' reviews of hotels in "Trip Advisor" dataset and used it to produce context-aware recommendations. Our evaluations indicate that using the contextual information can improve the performance of the recommender system in terms of hit ratio.

References

[1] D. Ramge, D. Hall, R. Nallapati, and C. Manning, "Labeled lda: a supervised topic model for credit attribution in multi-labeled corpora," in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, 2009.

[2] G. Abowd, A. Dey, N. D. P.J. Brown, M. Smith, and P. Steggles, "Towards a better understanding of context and context-awareness," *Handheld and Ubiquitous Computing*, vol. 1707, no. 2, pp. 304–307, 1999.

[3] H. Lieberman and T.Selker, "Out of context: Computer systems that adapt to, and lean from, context," *IBM Systems Journal*, vol. 39, no. 3, pp. 617–632, 2000.

[4] W. Woerndl and J. Schlichter, "Introducing context into recommender systems," in *Proceedings of AAAI Workshop on Recommender Systems in E-Commerce*, 2007, pp. 138–140.

[5] P. Dourish, "What do we talk about when we talk about context," *Personal and Ubiquitous Computing*, vol. 8, no. 1, pp. 19–30, 2004.

[6] G. Adomavicius and A. Tuzhilin, "Context-aware recommender systems," in *Proceedings of the 2008 ACM conference on Recommender Systems*. ACM, 2008.

[7] S. Bourke, K. McCarthy, and B. Smyth, "The social camera: Recommending photo composition using contextual features," in *Proceedings of Workshop on Context-Aware Recommender System*. ACM, 2010.

[8] K. Cheverst, N. Davies, K. Mitchell, A. Friday, and C. Efstratiou, "Developing a context-aware electronic tourist guide: some isues and experiences," in *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, p. 17.

[9] L. Ardissono, A. Goy, G. Petrone, M. Segnan, and P. Torasso, "Intrigue: personalized recommendation of tourist attractions for desktop and hand held devices," *Applied Artificial Intelligence*, vol. 17, no. 8, pp. 678–714, 2003.

[10] M. V. Setten, S. Pokraev, and J. Koolwaaij, "Context-aware recommendations in the mobile tourist application compass," in *Proceedings of Third International Conference In Adaptive Hypermedia and Adaptive Web-Based Systems*. Springer, August 2004.

[11] L. Baltrunas and X. Amatriain, "Towards time-dependant recommendation based on implicit feedback," in *Proceedings of Workshop on Context-Aware Recommender System*. ACM, 2009.

[12] T. Bogers, "Movie recommendation using random walks over the contextual graph," in *Proceedings of Workshop on Context-Aware Recommender System*. ACM, 2010.

[13] S. Anand and B. Mobasher, "Contextual recommendation," *From Web to Social Web: Discovering and Deploying User and Content Profiles*, 2007.

[14] S. Aciar, "Mining context information from consumers reviews," in *Proceedings of Workshop on Context-Aware Recommender System*. ACM, 2010.

[15] T. Hoffman, "Probabilistic latent semantic indexing," in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR99)*, 1999.

[16] D. Blei, A. Ng, and M. Jordan, "Latent dirichlet allocation," *The Journal of Machine Learning Research*, vol. 3, 2003.

[17] D. Blei and J. McAuliffe, "Supervised topic models," *Neural Information Processing Systems*, vol. 21, 2007.

[18] L. Julien, F. Sha, and M. I. Jordan, "Disclda: Discriminative learning for dimensionality reduction and classification," *Neural Information Processing Systems*, vol. 22, 2008.

[19] D. j. S. W. R. Gilks, S. Richardson, *Markov chain Monte Carlo in practice*. London: Chapman & Hall, 1996.

[20] L. Baltrunas, M. Kaminskas, F. Ricci, L. Rokach, B. Shapira, and K. Luke, "Best usage context prediction for music tracks," in *Proceedings of the 2nd Workshop on Context Aware Recommender Systems*, September 2010.

[21] C. Cantrell, *Modern Mathematical Methods for Physicists and Engineers*. Cambridge University Press, 2000.

[22] R. L. S. Kullback, "On information and sufficiency," *Annals of Mathematical Statistics*, vol. 22, pp. 79–86, 1951.

A Bayesian Concept Learning Approach to Crowdsourcing

Paolo Viappiani
Dept. of Computer Science
Aalborg University

Sandra Zilles, Howard J. Hamilton
Dept. of Computer Science
University of Regina

Craig Boutilier
Dept. of Computer Science
University of Toronto

Abstract

We develop a Bayesian approach to concept learning for crowdsourcing applications. A probabilistic belief over possible concept definitions is maintained and updated according to (noisy) observations from experts, whose behaviors are modeled using discrete types. We propose recommendation techniques, inference methods, and query selection strategies to assist a user charged with choosing a configuration that satisfies some (partially known) concept. Our model is able to simultaneously learn the concept definition and the types of the experts. We evaluate our model with simulations, showing that our Bayesian strategies are effective even in large concept spaces with many uninformative experts.

1 Introduction

Crowdsourcing is the act of outsourcing a problem to a group or a community. It is often referred to as *human computation*, as human "experts" are used to solve problems present difficulty for algorithmic methods; examples include Amazon's Mechanical Turk, the ESP game (for image labeling), and *reCaptcha* (for book digitization). Multiple human teachers, or experts, give feedback about (label) a particular problem instance. For instance, users refer to sites such as Yahoo! Answers to ask questions about everything from cooking recipes to bureaucratic instructions and health suggestions (e.g., which ingredients do I need to make tiramisu? how do I apply for a Chinese visa? how do I lose 20 pounds?).

As the information obtained with crowdsourcing is inherently noisy, effective strategies for aggregating multiple sources of information are critical. Aggregating noisy labels and controlling workflows are two problems in crowdsourcing that have recently been addressed with principled techniques [Dai *et al.*, 2010; Shahaf and Horvitz, 2010; Chen *et al.*, 2010]. In this work, we address the problem of generating recommendation for a user, where recommendation quality depends on some latent concept. The knowledge of the concept can only be refined by aggregating information from noisy information sources (e.g., human experts), and the user's objective is to maximize the quality of her choice as measured by satisfaction of the unknown latent concept. Achieving *complete* knowledge of the concept may be infeasible due to the quality of information provided by the experts, but also unnecessary. For instance, to successfully make tiramisu (a type of cake), certain ingredients might be necessary, while others may be optional. The concept c represents all possible "correct" recipes A configuration or instance x is a candidate recipe, and it satisfies c iff it can be used to make the cake (i.e., is correct). By asking various, possibly noisy, experts about particular ingredients, the user may "learn" a recipe satisfying c without ever learning *all* recipes satisfying c.

Following [Boutilier *et al.*, 2009], our aim is not to learn the concept definition *per se*; rather we want to learn just enough about it to make a (near-)optimal decision on the user's behalf. By exploiting the structure of the concept, a recommender system can adopt a strategy that queries only concept information that is relevant to the task at hand. For instance, if the system knows that an ingredient is extremely unlikely to be used in tiramisu, or is unlikely to be available, querying about this ingredient is unlikely to be helpful. Finally, the system needs to select the experts whose answers are (predicted to be) as informative as possible.

Our main contributions are 1) computational procedures to aggregate concept information (originating from noisy experts) into a probabilistic belief, 2) algorithms to generate recommendations that maximize the likelihood of concept satisfaction and 3) strategies to interactively select queries and experts to pose them to.

Our work is related to the model of Boutilier et al. [Boutilier *et al.*, 2009; 2010], who present a regret-based framework for learning subjective features in the context of preference elicitation. Our approach can be seen both as a Bayesian counterpart of that model, and as an extension to the case of multiple experts.

2 Bayesian Concept Learning Approach

We consider the problem of learning a latent concept by aggregating information from several sources called *experts*. Each expert may have a partial and incorrect definition of the concept. As in traditional concept learning [Mitchell0, 1977; Kearns and Li, 1993], we assume an abstract concept c is drawn from a concept class $\mathcal{C}$. However, instead of trying to identify the concept explicitly, we maintain a distribution

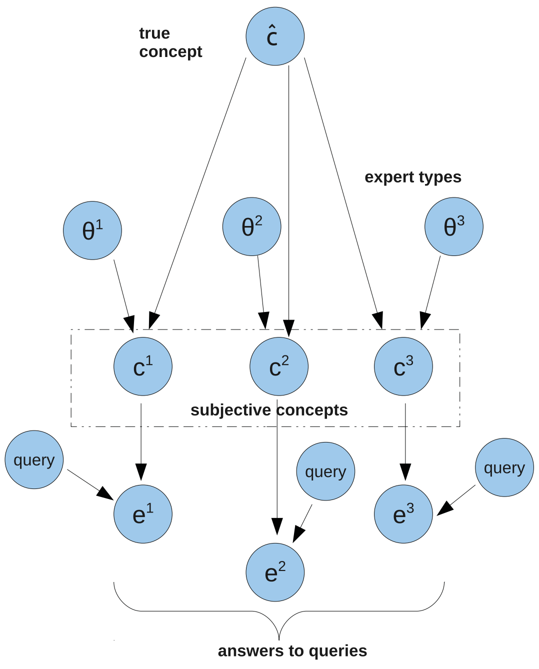


Figure 1: Abstract model of the problem of learning an unknown concept from mutiple noisy experts.

over possible concept definitions, and update the distribution according to the information acquired from the experts, in order to recommend an instance that is highly likely to satisfy the concept.

2.1 Concepts

We consider the problem of learning an abstract boolean concept drawn from a fixed concept class. A boolean concept c is a function $\{0,1\}^n \to \{0,1\}$, where $\{\mathcal{X}_1, \ldots, \mathcal{X}_n\}$ is a set of n boolean features. A *solution* (goal of the learning problem) is any boolean vector (configuration) $(x_1, \ldots, x_n) \in \{0,1\}^n$ for which $c(x_1, \ldots, x_n) = 1$. We allow the solution space to be restricted by feasibility constraints; below we assume linear constraints of the type $A{\cdot}x \leq B$ (with matrix A and vector B of the right dimensions). For example, *budget constraints* associate a vector of costs $(a_1, \ldots, a_n)$ with each feature and require total cost not to exceed the available budget b.

Throughout the paper, we restrict our focus to conjunctions [Haussler, 1989] as latent concepts, although our abstract model can be extended to boolean functions in general. A conjunctive concept c is a conjunction of literals over (some of) the atoms $\mathcal{X}_1, \ldots, \mathcal{X}_n$, e.g., $c = \mathcal{X}_2 \wedge \neg\mathcal{X}_4 \wedge \mathcal{X}_7$. A conjunction c can be equivalently represented as an assignment $(X_1^c, \ldots, X_n^c)$ of features $\mathcal{X}_1, \ldots, \mathcal{X}_n$ to the domain $\{T, F, DC\}$; in other words X_i^c can have one of the values T (true; the literal $\mathcal{X}_i$ occurs in c), F (false; the literal $\neg\mathcal{X}_i$ occurs in c), or DC (don't care; the atom $\mathcal{X}_i$ does not occur in c). In the above example, $X_2^c = X_7^c = T$, $X_4^c = F$, and $X_i^c = DC$ for $i \in \{2, 4, 7\}$.

Since the latter representation is used throughout the text, we write $c = (X_1^c, \ldots, X_n^c)$ and, with a slight abuse of notation, we will sometime refer to X_i^c as the "value" of feature i in concept c; we will also drop the superscript when c is clear from context. A configuration $x = (x_1, \ldots, x_n)$ yields $c(x) = 1$ (we say x satisfies c) iff (i) $x_i = 1$ for each i such that the literal $\mathcal{X}_i$ occurs in c ($X_i^c = T$), and (ii) $x_i = 0$ for each i such that the literal $\neg\mathcal{X}_i$ occurs in c ($X_i^c = F$).

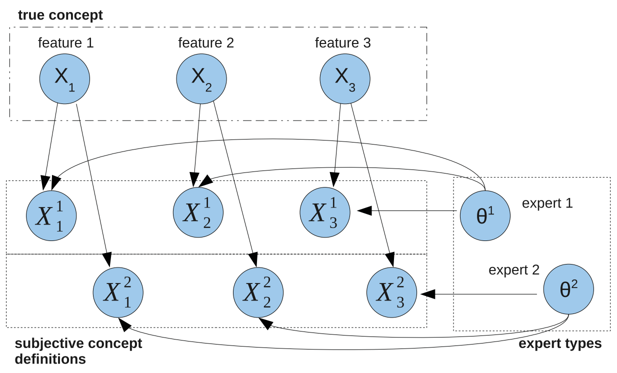


Figure 2: Graphical model for Bayesian learning of conjunctions (3 features, 2 experts).

Because the concept is unknown, the system maintains a belief $P(c) = P(X_1^c, \ldots, X_n^c)$. We assume some prior distribution over concepts. It is sometimes convenient to reason with the marginal probabilities, $P(X_i)$, representing the distribution over feature i, i.e., $P(X_i = T)$, $P(X_i = F)$, and $P(X_i = DC)$; for convenience, we write these terms as $P(T_i)$, $P(F_i)$, and $P(DC_i)$, respectively.

2.2 Query Types

The system acquires information about the concept by posing queries to a set of experts. These concept queries can be of different forms (e.g., membership, equivalence, superset, or subset queries [Angluin, 1988]) and their answers partition the hypothesis space. For instance, a membership query asks whether a given configuration x satisfies the concept (e.g., *"Is this a valid recipe for tiramisu?"*). Membership queries can be too cognitively demanding for a crowd-sourcing domain, as an expert would have to verify every problem feature to check whether the provided instance is satisfied. Thus, in this work we focus on *literal* queries, a special form of *superset queries*. A literal query q_i on feature i asks for the value of X_i; possible answers to the query are T, F, or DC.[1] Literal queries can be thought of as requests for a piece of information such as *"Are eggs needed for tiramisu?"*. Query strategies for selecting literal queries are discussed in Section 4.[2]

2.3 Expert Types

In practice, experts do not always provide correct answers. Hence we assume that experts belong to different populations

[1]Alternatively, one could ask queries such as "Is X_i positive in the concept definition?" Adapting our model to such queries is straightforward.

[2]Notice that literal queries cannot be answered unambiguously in general since dependencies may exist; but the value of a literal in a conjuctive concept is independent of the value of any other literal.

or *types* from a set $\mathcal{T} = \{t_1, \ldots, t_k\}$. The type of an expert represents the expert's capacity and commitment to correctly answering queries about the concept (or aspects thereof). For instance, as in [Chen *et al.*, 2010], types might discriminate "good" or knowledgeable experts, whose answers are likely to be correct, from "bad" experts, whose answers are drawn randomly. Our model generalizes to any number of types.

We indicate the assignments of types to experts with a vector $\theta = (\theta^1, \ldots, \theta^m)$, where $\theta^j \in \mathcal{T}$ is the type of expert j. A further natural assumption is that experts are noisy and provide feedback with respect to their subjective definition of the concept. In other words, we assume that there exists one underlying (true) concept definition $\hat{c} = (X_1, \ldots, X_n)$, but each expert's response is based on its own subjective concept $c^j = (X_1^j, \ldots, X_n^j)$. When a query q_i^j on feature i is posed to expert j, the expert reveals its subjective value X_i^j for that feature (either T, F or DC). Subjective concepts are distributed, in turn, according to a generative model $P(c^j|\hat{c}, \theta^j)$, given expert type θ^j and true concept $\hat{c}$. For example, an "uninformed" expert may have a subjective concept that is probabilistically independent of $\hat{c}$, while an "informed" expert may have a concept that is much more closely aligned with $\hat{c}$ with high probability. In our experiments below, we assume a factored model $P(X_i^j|X_i, \theta^j)$. Moreover, since we always ask about a specific literal, we call this distribution the *response model*, as it specifies the probability of expert responses as function of their type. This supports Bayesian inference about the concept given expert answers to queries (note that we do not assume expert types are themselves observed; inference is also used to estimate a distribution over types).

The graphical model for the general case is shown in Figure 1. In Figure 2 we show the model for conjunctions with 3 features and 2 experts; the subjective concept c^j of expert $j \in \{1, 2\}$ is composed by X_1^j, X_2^j and X_3^j.

As queries provide only "noisy" information about the true concept $\hat{c}$, the system cannot fully *eliminate* hypotheses from the version space given expert responses. To handle concept uncertainty, the system maintains a distribution or *belief* $P(c)$ over concept definitions, as well as a distribution over expert types $P(\theta)$. Both distributions are updated whenever queries are answered.

Beliefs about the true concept and expert subjective concepts will generally be correlated, as will beliefs about the types of different experts. Intuitively, if two experts consistently give similar answers, we expect them to be of the same type. When we acquire additional evidence about the type of one expert, this evidence affects our belief about the type of the other expert as well. Thus, when new evidence e is acquired, the joint posterior $P(c, \theta|e)$ cannot be decomposed into independent marginals over c and the θ^j, since c and θ are not generally independent. Similarly, new evidence about feature X_i might change one's beliefs about types, and therefore influence beliefs about another feature X_j. We discuss the impact of such dependence on inference below.

2.4 Decision-making

The system needs to recommend a configuration $x = (x_1, \ldots, x_n) \subseteq \{0, 1\}^n$ that is likely to satisfy the concept (e.g., a recipe for tiramisu), based on the current belief $P(c)$. A natural approach is to choose a configuration x^* that maximizes the *a posteriori* probability of concept satisfaction (MAPSAT) according to the current belief: $x^* \in \arg\max_x P(c(x))$.

Exact maximization typically requires enumerating all possible configurations and concept definitions. Since this is not feasible, we consider the marginalized belief over concept features and optimize, as a surrogate, the product of probabilities of the individual features satisfying the configuration: $P(c(x)) \approx \tilde{P}(c(x)) = \prod_i P(c_i(x_i))$, where c_i is the restriction of concept c to feature i. In this way, optimization without feasibility or budget constraints can be easily handled. For each feature i, we choose $x_i = 1$ whenever $P(T_i) \geq P(F_i)$, and choose $x_i = 0$ otherwise.

However, in the presence of feasibility constraints, we cannot freely choose to set attributes in order to maximize the probability of concept satisfaction. We show how, using a simple reformulation, this can be solved as an integer program. Let $p_i^+ = P(T_i) + P(DC_i)$ be the probability that setting $x_i = 1$ is consistent with the concept definition for the i-th feature; similarly let $p_i^- = P(F_i) + P(DC_i)$ be the probability that setting $x_i = 0$ is consistent. Then the probability of satisfying the i-th feature is $P(c_i(x_i)) = p_i^+ x_i + p_i^-(1 - x_i)$. The overall (approximated) probability of concept satisfaction can be written as:

$$P(c(x)) \approx \prod_{1 \leq i \leq n} p_i^+ x_i + p_i^-(1-x_i) = \prod_{1 \leq i \leq n} (p_i^+)^{x_i} \prod_{1 \leq i \leq n} (p_i^-)^{(1-x_i)} \tag{1}$$

The latter form is convenient because we can linearize the expression by applying logarithms. To obtain the feasible configuration x^* maximizing the probability of satisfaction, we solve the following integer program (the known term has been simplified):

$$\max_{x_1, \ldots, x_n} \sum_{1 \leq i \leq n} [log(p_i^+) - log(p_i^-)] \cdot x_i \tag{2}$$

$$s.t. \quad A \cdot x \leq B \tag{3}$$

$$x \in \{0, 1\}^n \tag{4}$$

3 Inference

When a query is answered by some expert, the system needs to update its beliefs.Let e_i^j represent the evidence (query response) that expert j offers about feature i. Using Bayes' rule, we update the probability of the concept: $P(c|e_i^j) \propto P(e_i^j|c)P(c)$. Since the type θ^j of expert j is also uncertain, inference requires particular care. We consider below several strategies for inference. When discussing their complexity, we let n denote the number of features, m the number of experts, and k the number of types.

Exact Inference Exact inference is intractable for all but the simplest concepts. A naive implementation of exact inference would be exponential in both the number of features and the number of experts. However, inference can be made more efficient by exploiting the independence in the graphical model. Expert types are mutually independent given concept c: $P(\theta|c) = \prod_{1 \leq j \leq m} P(\theta^j|c)$. This means that each concept can be "safely" associated with a vector of m probabilities $P(\theta^1|c), \ldots, P(\theta^m|c)$, one for each expert. For a concept space defined over n features, we explicitly represent the 3^n possible concept definitions, each associated with a matrix (of dimension m by k) representing $P(\theta|c)$. The probability of a concept is updated by multiplying the likelihood of the evidence and renormalizing: $P(c|e_i^j) \propto P(e_i^j|c)P(c)$. As the queries we consider are *local* (i.e., only refer to a single feature), the likelihood of c is

$$P(e_i^j|c) = \sum_{t \in \mathcal{T}} P(e_i^j|\theta^j = t, X_i^c)P(\theta^j = t|c), \tag{5}$$

where X_i^c is the value of c for feature $\mathcal{X}_i$. The vector $(P(\theta^1|c, e_i^j), \ldots, P(\theta^m|c, e_i^j))$ is updated similarly. Overall complexity is $O(3^n mk)$. Since the number of experts m is usually much larger than the number of features n, exact inference is possible for small concept spaces. In practice, it is only feasible for up to 5–10 features; in our implementation, exact inference with $n = 7$ and $m = 100$ requires 3–4 seconds per query.

Naive Bayes This approach to inference makes the strong assumption that X_i and θ^j are mutually conditionally independent. This allows us to factor the concept distribution into marginals over features: $P(X_1), \ldots, P(X_n)$; similarly beliefs about experts are represented as $P(\theta^1), \ldots, P(\theta^m)$. The likelihood $P(e_i^j|X_i)$ of an answer to a query can be related to $P(e_i^j|\theta^j, X_i)$ (the response model) by marginalization over the possible types of expert j: $P(e_i^j|X_i) = \sum_{v \in \{t_1, t_2, \ldots\}} P(e_i^j|\theta^j = v, X_i)P(\theta^j = v|X_i)$. We write the expression for the updated belief about X_i given evidence:[3]

$$P(X_i|e_i^j) = \frac{P(e_i^j|X_i)P(X_i)}{P(e_i^j)} \tag{6}$$

$$= \frac{\sum_{t \in \mathcal{T}} P(e_i^j|X_i, \theta^j{=}t)P(\theta^j, X_i)}{\sum_{z \in \{T,F,DC\}} \sum_{t \in \mathcal{T}} P(e_i^j|X_i{=}z, \theta^j{=}t)P(\theta^j, X_i)} \tag{7}$$

We update belief $P(X_i)$ using current type beliefs $P(\theta^1), \ldots, P(\theta^m)$. Our strong independence assumption allows simplification of Eq. 7:

$$P(X_i|e_i^j) = \frac{\sum_{t \in \mathcal{T}} P(e_i^j|X_i, \theta^j{=}t)P(\theta^j{=}t)}{\sum_z \sum_{t'} P(e_i^j|X_i{=}z, \theta^j = t')\, P(\theta^j{=}t')\, P(X_i{=}z)} P(X_i) \tag{8}$$

Similarly, for beliefs about types we have:

$$P(\theta^j|e_i^j) = \frac{\sum_z P(e_i^j|X_k = z, \theta^j)\, P(X_i = z)}{\sum_{z'} \sum_t P(e_i^j|X_i = z', \theta^j = t)P(\theta^j)P(X_i = z')} P(\theta^j) \tag{9}$$

This approximation is crude, but performs well in some settings. Moreover, with space complexity $O(n + m)$ and time complexity $O(nm)$, it is very efficient.

[3]Using Naive Bayes, we only update concept beliefs about X_i, the feature we asked about. Similarly, for types, we only update relative to θ^j, the expert that answered the query.

Monte Carlo This approximate inference technique maintains a set of l particles, each representing a specific concept definition, using importance sampling. As with exact inference, we can factor beliefs about types. The marginal probability $P(X_i)$ that a given feature is true in the concept is approximated by the fraction of the particles in which X_i is true (marginalization over types is analogous). Whenever queries are answered, the set of particles is updated recursively with a resampling scheme. Each particle is weighted by the likelihood of the concept definition associated with the particle when evidence e_k^u is observed (the higher the likelihood, the higher the chance of resampling). Formally, the expression of the likelihood of a particle is analogous to the case of exact inference, but we only consider a limited number of possible concepts. Monte Carlo has $O(lmk)$ complexity; hence, it is more expensive than Naive Bayes but less expensive than exact inference.

4 Query Strategies

We now present elicitation strategies for selecting queries. Each strategy is a combination of methods that, given the current beliefs about the concept and the types, i) selects a feature to ask about, and ii) selects the expert to ask. Expert selection depends on the semantics of the types; here, as in [Chen *et al.*, 2010], we assume experts are either "knowledgeable" (type t_1) or "ignorant" (type t_2). As baseline, we consider two inefficient strategies for comparison purposes: (i) *broadcast* iterates over the features and, for each, asks the same query to a fixed number of experts, and (ii) *dummy* asks random queries of random experts and recommends the most frequent answers.

Feature Selection We consider three strategies aimed at directly reducing concept uncertainty. The *maximum entropy* (or *maxent*) strategy selects the feature whose probability distribution over $\{T, F, DC\}$ has the greatest entropy. Unfortunately, this measure treats being uncertain between a T and F as the same as being uncertain between T and DC. The *minval* strategy selects the feature $\mathcal{X}_f$ with the lowest probability of "getting it right:" that is, $f = \arg\min_i\{\max(p_i^+, p_i^-)\}$ is viewed as the feature with the greatest potential for improvement. Each feature is "scored" using the probability, given our current beliefs, that the best guess for its feature value will match the true concept. The intention is to reduce the uncertainty that most hinders the chance of satisfying the concept. Finally, queries can be evaluated with respect to their capacity to improve decision quality using value of information [Howard, 1966]. We optimize *expected value of perfect information (EVPI)*; as shown below, this criterion can be computed using the current belief without expensive Bayesian updates. In this setting, *EVPI* measures the expected gain in the quality of a decision should we have access to perfect information about a particular feature. In other words, given an oracle able to provide the actual value (T, F or DC) of a feature, which should we ask about? The value of querying feature X_i is:[4]

$$EVPI_i = \sum_{z\in\{T,F,DC\}} P(X_i = z) \max_x P(c(x)|X_i = z). \tag{10}$$

Since we aim to select queries quickly, we also consider *Naive EVPI*, where $P(c(x)|X_i)$ is approximated by the product of satisfying each feature.

Observation 1 *In unconstrained problems, the feature selected with the minval heuristic strategy is associated with maximum Naive EVPI.*

The proof is provided in the Appendix. It relies on the fact that, without feasibility constraints, one can optimize features independently. For the more general case, given feature i, we define $x^{+i} = \arg\max_{x\in X: x_i=1} P(c(x))$ to be the optimal configuration among those where feature i is true; we define x^{-i} analogously. We write the approximated satisfaction probabilities as $\tilde{P}(c(x^{+i})) = p_i^+ \cdot p_{\neq i}^{+i}$, where $p_{\neq i}^{+i} = \prod_{j\neq i} P(c_j(x^{+i}))$, and $\tilde{P}(c(x^{-i})) = p_i^- \cdot p_{\neq i}^{-i}$.

Observation 2 *Naive* EVPI *can readily be computed using the current belief:*

$$EVPI_i = P(T_i)p_{\neq i}^{+i} + P(F_i)p_{\neq i}^{-i} + P(DC_i)\cdot\max\{p_{\neq i}^{+i}, p_{\neq i}^{-i}\}$$

From this Observation it follows that, if $P(DC_i) = 0$ (we know that a feature is either true or false in the concept definition), then $EVPI_i = \tilde{P}(c(x^{+i})) + \tilde{P}(c(x^{-i}))$. The most informative feature is the feature i that maximizes the sum of the probability of concept satisfaction of x^{+i} and x^{-i}. This, in particular, is true when one considers a concept space where "don't care" is not allowed.

Naive *EVPI* query maximization is in general very efficient. As the current best configuration x^* will coincide with either x^{i+} or x^{i-} for any feature i, it requires only $n+1$ MAPSAT-optimizations and n evaluations of *EVPI* using Observation 2. Its computational complexity is not affected by the number of experts m.

Expert Selection For a given feature, the *greedy* strategy selects the expert with the highest probability of giving an informative answer (i.e., one of type t_1). It is restricted to never ask the the same expert about the same feature, which would be useless in our model. However, there can be value in posing a query to an expert other than that predicted to be most "knowledgeable" because we may learn more about the types of other experts. The *soft-max* heuristic accomplishes this by selecting an expert j according to a Boltzmann distribution $\frac{e^{P(\theta^j = t_1)/\tau}}{\sum_r e^{P(\theta^r = t_1)/\tau}}$ with "temperature" τ, so that experts that are more likely to be of type t_1 are queried more often.

[4]We consider each possible response (T, F or DC) by the oracle, the recommended configuration conditioned to the oracle's answer, and weight the results using the probability of the oracle's response.

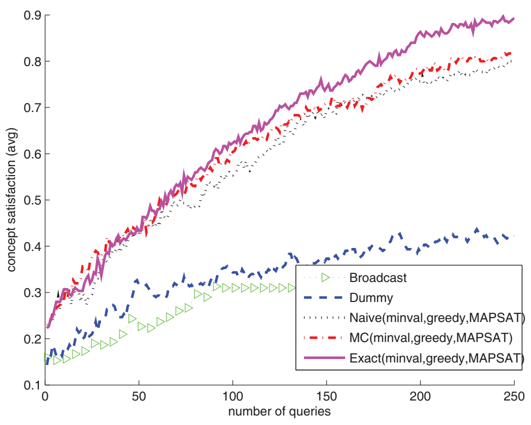


Figure 3: Simulation with 5 features, 100 experts (20% knowledgeable experts); 300 runs

Combined Selection There can be value in choosing the feature and expert to ask in combination. We consider strategies inspired by work on multi-armed bandit problems [Sutton and Barto, 1998], aimed at resolving the tradeoff between *exploration* and *exploitation*. In this setting, exploitation means using a strategy such as *EVPI* to learn more about the concept; in this case, we select experts greedily. On the other hand, exploration in this context means using a strategy such as *soft-max* to *learn more* about expert types; in this case, we select the feature we are most certain about because it will provide the most information about an expert's type. The *explore-exploit* strategy embodies this tradeoff: we generate the pair (i, j), where $\mathcal{X}_i$ is the feature that maximizes *EVPI* and j is the expert chosen greedily as above. We then consider our current belief $P(\theta^j)$ about its type and use this to switch between exploitation and exploration. We sample a value from $P(\theta^j)$; if we obtain t_1, we query q_i^j (exploitation), otherwise, we generate (i', j'), where i' is the index of the feature we are most certain about and j' is chosen with *soft-max* (exploration). In practice this method is more effective using a Boltzmann distribution over types; in the experiments below we "exploit" with probability $0.5 + 0.5 * \frac{e^{P(\theta^j = t_1)/\tau}}{e^{P(\theta^j = t_1)/\tau} + e^{P(\theta^j = t_2)/\tau}}$.

5 Experiments

We experimented with the query strategies described in Section 4 by comparing their effectiveness on randomly generated configuration problems and concepts. Queries are asked of simulated experts, each with a type and a subjective concept drawn from a prior distribution.[5] At any stage, each strategy recommends a configuration (decision) based on the

[5]The type is either "knowledgeable" or "ignorant." We define probabilities for subjective concept definitions such that 70% of the time, knowledgeable experts reveal the true value of a particular feature (i.i.d. over different features), and a true T value is reported to be DC with higher probability than is F. Ignorant experts are uninformative (in expectation) with each feature value T, F, and DC sampled

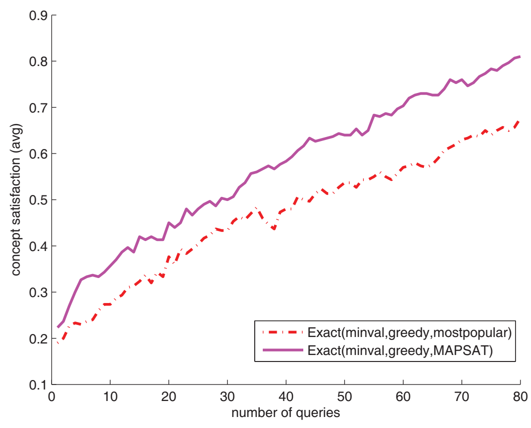


Figure 4: MAPSAT vs *mostpopular* (5 features, 100 experts, 30% knowledgeable, 300 runs)

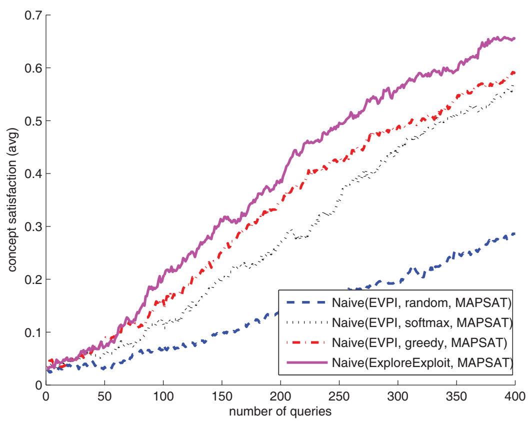


Figure 6: Evaluation of expert selection methods (20 features; 20% of experts are knowledgeable; 500 runs)

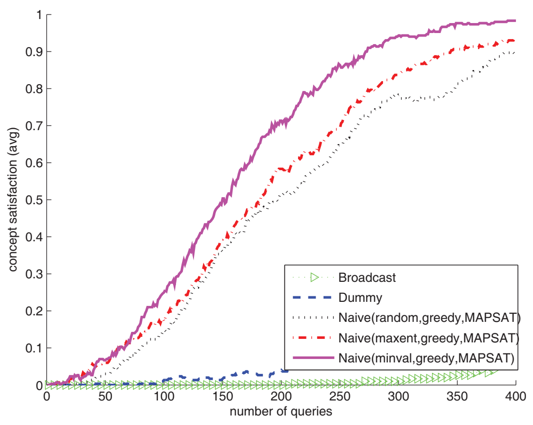


Figure 5: Evaluation of feature selection methods in a larger concept space (30 features; 50% knowledgeable; 500 runs)

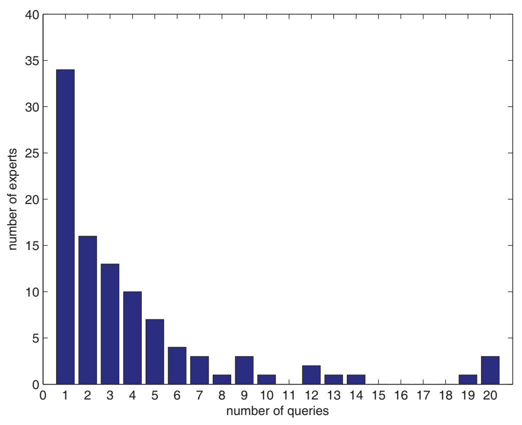


Figure 7: Distribution of the number of queries posed to experts

current belief and selects the next query to ask; we record whether the current configuration satisfies the true concept.

The concept prior (which is available to the recommender system) is sampled using independent Dirichlet priors for each feature; this represents cases where prior knowledge is available about which features are most likely to be involved (either positively or negatively) in the concept. A *strategy* is a combination of: an inference method; a heuristic for selecting queries (feature and expert); and a method for making recommendations (either MAPSAT or *mostpopular*, the latter a heuristic that recommends each configuration feature based on the most common response from the experts).

Our results below show that good recommendations can be offered with very limited concept information. Furthermore, our decision-theoretic heuristics generate queries that allow a concept-satisfying recommendation to be found quickly (i.e., with relatively few expert queries). In the first experiment (see Figure 3), we consider a setting with 5 features and 100 experts, and compare all methods for Bayesian inference (Exact, Naive and Monte Carlo with 100 particles). All three methods generate queries using *minval* (to select features) and *greedy* (to select experts). We also include *broadcast* and *dummy*. Only 20% of experts are knowledgeable, which makes the setting very challenging, but potentially realistic in certain crowdsourcing domains. Nonetheless our Bayesian methods identify a satisfactory configuration relatively quickly. While the exact method performs best, naive inference is roughly as effective as the more computationally demanding Monte Carlo strategy, and both provide good approximations to Exact in terms of recommendation quality. *Dummy* and *broadcast* perform poorly; one cannot expect to make good recommendations by using a simple “majority rule” based on answers to poorly selected queries. In a similar setting, we show that MAPSAT outperforms *mostpopular* (assign features based on the most frequent answers) for

from a random multinomial, drawn from a Dirichlet prior *Dir(4,4,4)*. Since an expert’s answers are consistent with its subjective concept, repeating a query to some expert has no value.

choosing the current recommendation also when used with exact inference (Figure 4).[6]

In the next experiment, we consider a much larger concept space with 30 boolean variables (Figure 5). In this more challenging setting, exact inference is intractable; so we use Naive Bayes for inference and compare heuristics for selecting features for queries. *Minval* is most effective, though maxent and random perform reasonably well.

Finally we evaluate heuristics for selecting experts (*random*, *greedy* and *softmax*) and the combined strategy (*explore-exploit*) in presence of budgeted constraints. Each feature is associated with a cost a_i uniformly distributed between 1 and 10; this cost is only incurred when setting a feature as positive (e.g. when buying an ingredient); the available budget b is set to $0.8 \cdot \sum_i a_i$.

Figure 6 shows that the *explore-exploit* is effective and outperforms the other strategies. This suggests that our combined method balances exploration (asking queries in order to know more about the type of the experts) and exploitation (asking the query to the must knowledgeable expert given our belief) in an effective way. It is interesting to observe that *Naive(EVPI,greedy,MAPSAT)*, while using the same underlying heuristic for selecting features as *Naive(explore-exploit,MAPSAT)*, is very effective at the beginning but becomes outperformed after approximately 50-60 queries, as it never explicitly tries to ask queries aimed at improving knowledge about the expert types.

Although the number of queries may seem large, they are asked of different experts; a single expert is asked at most n queries, and most experts are asked only 1 or 2 queries. Figure 7 shows a histogram about the number of queries asked to experts by *explore-exploit* in the last setting: 3 experts are asked 20 queries, while 34 experts are asked only one.

6 Discussion and Future Work

We have presented a probabilistic framework for learning concepts from noisy experts in a crowdsourcing setting, with an emphasis on learning just enough about the concept to identify a concept instance with high probability. We described methods for making recommendations given uncertain concept information and how to determine the most "relevant" queries. Since experts are noisy, our methods acquire indirect information about their reliability by aggregating their responses to form a a distribution over expert types. Our experiments showed the effectiveness of our query strategies and our methods for approximate inference, even in large concept spaces, with many uninformative experts, and even when "good" experts are noisy.

The are many interesting future directions. Development of practical applications and validation with user studies is of critical importance. While we have focused on conjunctive concepts in this paper, we believe our model can be extended to more general concept classes. Special care, however, must be taken in several aspects of an extended model: the exact semantics of queries; the representation of the concept distribution; and inference over types and concepts. We are also interested in a game-theoretic extension of the model that allow (some or all) experts to provide responses that reflect their self-interest (e.g., by guiding a recommender system to specific products).

[6]As our heuristics only ask queries that are relevant, recommendations made by the *mostpopular* strategy are relatively good in this case.

Further investigation of query selection strategies is important; our strategies adopt ideas from multi-armed bandit and we are interested in exploring this connection in more details. Principled methods for query optimization in preference elicitation [Viappiani and Boutilier, 2010] could also provide additional insights.

Our model values configurations based on their probability of satisfying the concept (i.e., assuming binary utility for concept satisfaction). Several other utility models can be considered. For instance, we might define utility as a sum of some concept-independent reward for a configuration—reflecting user preferences over features that are independent of the latent concept—plus an additional reward for concept satisfaction (as in [Boutilier *et al.*, 2009; 2010]). One could also consider cases in which it is not known with certainty which features are available: the problem of generating recommendations under both concept and availability uncertainty would be of tremendous interest.

References

[Angluin, 1988] D. Angluin. Queries and concept learning. *Mach. Learn.*, 2:319–342, 1988.

[Boutilier *et al.*, 2009] C. Boutilier, K. Regan, and P. Viappiani. Online feature elicitation in interactive optimization. In *ICML 2009*, pages 73–81, 2009.

[Boutilier *et al.*, 2010] C. Boutilier, K. Regan, and P. Viappiani. Simultaneous elicitation of preference features and utility. In *AAAI 2010*, pages 1160–1197, 2010.

[Chen *et al.*, 2010] S. Chen, J. Zhang, G. Chen, and C. Zhang. What if the irresponsible teachers are dominating? In *AAAI 2010*, 2010.

[Dai *et al.*, 2010] P. Dai, Mausam, and D.S. Weld. Decision-theoretic control of crowd-sourced workflows. In *AAAI 2010*, 2010.

[Haussler, 1989] D. Haussler. Learning conjunctive concepts in structural domains. *Mach. Learn.*, 4:7–40, 1989.

[Howard, 1966] Ronald Howard. Information value theory. *IEEE Trans. on Systems Science and Cybernetics*, 2(1):22–26, 1966.

[Kearns and Li, 1993] M.J. Kearns and M. Li. Learning in the presence of malicious errors. *SIAM J. Comput.*, 22:807–837, 1993.

[Mitchell0, 1977] T.M. Mitchell0. Version spaces: A candidate elimination approach to rule learning. In *IJCAI 1977*, pages 305–310, 1977.

[Shahaf and Horvitz, 2010] D. Shahaf and E. Horvitz. Generalized task markets for human and machine computation. In *AAAI 2010*, 2010.

[Sutton and Barto, 1998] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. The MIT Press, 1998.

[Viappiani and Boutilier, 2010] Paolo Viappiani and Craig Boutilier. Optimal bayesian recommendation sets and myopically optimal choice query sets. In *Advances in Neural Information Processing Systems 23 (NIPS)*, pages 2352–2360. MIT press, 2010.

Proof of Observation 1: Assume we ask the oracle about feature i. Let $p_i^* = \max(p_i^+, p_i^-)$. The optimal configuration x^* *in the updated belief* given the oracle's response is such that $x^* = \arg\max_x \prod_i P(c_i(x)|X_i = v)$, where v (either T,F or DC) is the oracle's response. Since there are no constraints, it can be optimized independently for the different features. Feature i of the optimal configuration x_i^* will necessarily be set to 1 or 0 in a way consistent with v (in case of DC, either is equivalent) and we are sure that x_i^* satisfies feature i; all other features will be set according to p_i^*. The (approximated) probability of concept satisfaction is:

$$\max_x \prod_j P(c_j(x)|X_i = v) = \prod_{j \neq i} \max(p_j^+, p_j^-) = \prod_{j \neq i} p_i^* = p_{\neq i}^*. \tag{11}$$

Therefore, $EVPI_i = \sum_{v=T,F,DC} P(X_i = v) \cdot p_{\neq i}^* = p_{\neq i}^*$. The argument follows from observing that $i = \arg\max p_{\neq i}^*$ iff $i = \arg\min p_i^*$. ■

Proof of Observation 2: Note that x^{+i} and x^{-i} are the optimal configurations in the posterior beliefs $P(c|X_i = T)$ and $P(c|X_i = F)$ respectively. In the case that the oracle's answer is DC ("don't care") then the optimal configuration is either x^{+i} or x^{-i} depending on which of the two gives higher probability of satisfying all features beside i. The argument follows from Equation 10. ■

Using Social and Pseudo-Social Networks for Improved Recommendation Quality

Alan Said, Ernesto W. De Luca, Sahin Albayrak
DAI-Lab
TU-Berlin
{alan.said,ernesto.deluca,sahin.albayrak}@dai-lab.de

Abstract

Recommender systems attempt to find relevant data for their users. As the body of data available in the Web sphere becomes larger, this task becomes increasingly harder. In this paper we present a comparison of recommendation results when using different social and pseudo-social features commonly available in online movie recommendation communities. Social relations, whether inferred or not, hold implicit information about users' taste and interests. We present results of a simple method that extends standard collaborative filtering algorithms to include a social network and show that this explicit and implicit information (i.e. direct friendship, and indirect co-commenting etc.) can be used to improve the quality of recommendations.

1 Introduction

Estimates say that the currently accumulated amount of data in the digital universe reached 1.2 zettabytes (1 billion terabytes) in 2010, which corresponds to a 50% increase during the two last years [Gantz and Reinsel, 2010]. A body of data of this size presents substantial challenges for current information retrieval systems. Independent of whether the task is search-, classification- or recommendation-oriented, processing and personalizing results from these systems becomes one of the most important tasks in order to identify relevant information. Granted, most systems do not face data amounts of this size, it is however implied that this accumulated amount is reflected in many websites which have seen considerable increase of users during the same time, e.g. Netflix [Siedler, 2010].

In personalized recommender systems, the de facto standard *Collaborative Filtering* (CF) approach, is becoming an insufficient means to produce relevant results due to the information overload which follows from the rapid data growth [Montebello, 1998]. However, the significant increase in data brings benefits as well, benefits in the form of *richer* meta data, i.e. more information related to every transaction, consumption, movie rating, etc. Using this rich data to extend regular collaborative filtering approaches can result in better information management systems, no matter if they are retrieval or recommendation based.

In movie recommendation systems, recommender systems research has mostly been focused on algorithmic approaches to better use the available data. The two most popular movie recommendation datasets, from the Netflix Prize[1] and the Movielens[2] community, do not include any social or pseudo-social structures. However, this data is commonly available in other online recommendation communities.

1.1 Problem Statement and Contribution

In this paper, we evaluate how different social and pseudo-social relations can be employed in order to improve the quality of recommendations in a movie scenario. Our model presents how user-item interaction can be used to infer relations between users. We present early stage results where these relations, no matter if inferred or explicit, increase the performance of our collaborative filtering-based movie recommender

The main contribution of this paper is the evaluation of different types of social networks in order to improve recommendation quality.

1.2 Outline

In this paper, we limit ourselves to the domain of movie recommendation, using a dataset from the Moviepilot[3] online movie recommendation community, and present a simple extension of standard collaborative filtering which uses regular and inferred social networks similar to the method presented by Guy et al. [Guy *et al.*, 2009].

Our approach infers ties between users based on their history of *comments*, whether they have stated they are *fans* of the same people, whether they have stated they *like* the same news articles, and if they have an explicitly stated *friendship* relation.

The experiments performed in this paper show that when using these networks, we can improve recommendation results compared to regular collaborative filtering. The full details of our approach are presented in Section 3.

[1]http://www.netflixprize.com/
[2]http://www.movielens.org/
[3]http://www.moviepilot.de

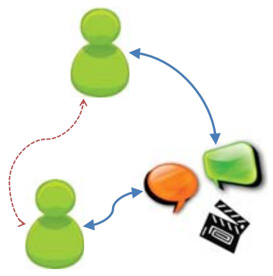


Figure 1: An inferred social tie (the red dotted line) is, in this case, created if two users have commented on the same movie, person or news article. The same principle is applied for users who are fans of the same actors, directors, or like the same news articles and comments.

2 Familiarity vs. Similarity

In standard collaborative filtering-based recommender systems, user similarities are calculated based on the user-movie relations (i.e. *similarity*), we use user-user relations in addition (i.e. *familiarity*). In our analysis and experiments we use a snapshot of the explicit friendship graph found in Moviepilot, as well as the implicit networks created when users interact with the same content (as shown in Figure 2), in order to improve the quality of our recommendations. The assumption is that a user's so-called *familiarity* networks hold implicit information about the user's so-called *similarity* (CF-based) network [Said *et al.*, 2010b]. We also present some statistical data on the dataset and its features.

3 Dataset and Experiments

Moviepilot is Germany's largest online movie recommendation community with more than one million users, over fifty thousand movies, and in excess of 10 million ratings.

3.1 The Dataset

Datasets provided by Moviepilot have been analyzed and researched previously [Said *et al.*, 2010a]. However, the dataset used in our evaluation differs from the ones used in prior publications. This dataset is a subset of the full, unfiltered data that creates the basis for the Moviepilot website. The dataset contains ratings by $10,000$ randomly selected users who have rated at least one movie. In addition to the ratings, the dataset also contains information on each user's friendship network within Moviepilot, as well as the comments posted by each user, the declarations of being a fan by each user (i.e. explicit statements saying a user is a fan of an actor, director, etc.) and the "diggs" of each user (i.e. users can "digg" different items such as comments, news articles etc.). The total number of ratings in our subset is $1,539,393$ spread over a period of four years (2006 to 2010). Table 1 shows the number of entities in the dataset and the approximate percentages of the full snapshot . The ratings are stored on a 0 to 100 scale with 0 being the lowest and 100 being the highest. The scale shown to the users is however 0.0 to 10.0. The networks used in this paper were either explicitly stated in the data (i.e. friendships) or were inferred from users' interactions with information available, i.e.:

Relation	Testset	%
Friendships	$3,764$	10%
Comments	$50,960$	30%
Fans	$170,092$	25%
Diggs	$25,259$	25%
Ratings	$1,539,393$	20%
Users$_{30+ratings}$	$10,000$	25%

Table 1: Dataset statistics for the snapshot we use and the (rough) percentage of the full dataset they represent. It should be noted that each Friendship relation is between two users, whereas each Comment-, Fan- and Digg-relation is a link between a user and the entity.

Type	Nodes	Edges
Friendship	$1,595$	$3,764$
Comments	$2,137$	$1,524,476$
Fans	$3,950$	$2,129,330$
Diggs	680	$20,028$

Table 2: The number of nodes and edges in every network. Similarly to the data in Table 1, each Friendship edge is between two users, whereas the other edges are between one user and the entity they interact with.

- the friendship graph - explicitly stated friendship relation between users
- the comments graph - an implicit network created when users comment on movies, actors, etc.
- the fan graph - an implicit network created when user are fans of the same people.
- the digg graph - an implicit network created when users "digg" the same news articles, comments, etc.

The sizes of the networks differ as the randomly selected users have diverse profiles, i.e. those with many friends and those with few, those who comment often and those who never comment, etc. The number of nodes and edges in each network is shown in Table 2, the number of ratings assigned by users in each of the networks is shown in Table 3.

3.2 Experimental Setup

For the experiments, 50 training and evaluation sets each for all networks were created. The evaluation sets consisted of

Type	Number	%
Friendship	$584,578$	38%
Comments	$697,012$	45%
Fans	$1,188,051$	77%
Diggs	$439,268$	29%

Table 3: The number and percentages of ratings assigned by users in the different networks (out of the 1.5 million ratings in our dataset). The sets are not necessarily overlapping.

circa 5000 ratings for 500 randomly selected users. In order to avoid problems related to cold start (when users have none or too few items for CF to generate good results) [Said *et al.*, 2009], for both users and items, we limit our evaluation to users who have rated at least 30 movies. For each of these users, 10 movies having been rated with a value above the user's average rating were extracted into the evaluation set (i.e. the set of true positive recommendations). The rest of the ratings were used for training. The recommendation algorithm was run twice for the 50 pairs of datasets, once taking the networks into consideration, and once neglecting the additional data. The results presented in this paper are averaged over all runs.

The recommendation algorithm used in our experiments was a slightly modified version of *K-Nearest Neighbor* using the Pearson Correlation Coefficient as the neighbor similarity measure. The pearson similarity of two users who were connected in the networks was multiplied by a factor of $10,000$ (the number of users in our dataset) in order to significantly affect the similarity measure. Experiments were performed with K set to 200. Additionally, a random recommender was used as a baseline for comparison. It should be noted that the algorithm itself is not the focus of our evaluation, rather the effects of using this additional information for recommendation.

3.3 Results

We evaluate our recommendations with the Mean Average Precision (MAP) and Precision at 10 (P@10) measures. These measures where chosen since they are well-known and widely-used in the field of Recommender Systems and Information Retrieval, providing a statistically sound estimate of the recommendation quality [Herlocker *et al.*, 2004].

Table 3(a) shows the precision levels obtained in our experiments. As the training and test splits for each network type have been created separately (due to the sets not necessarily being overlapping), they can thus not be compared to each other directly. Therefore, the table also shows the result of a standard Pearson-based KNN recommender on the same training and test split compared to the values of social recommendations. Table 3(b) shows the MAP values in a similar fashion.

Our resulting recommendations using social and pseudo-social networks perform between 0.2% and 5.4% better (in MAP values) than a regular KNN recommender and similarly in terms of P@10. We find that the pseudo-social network created from fan relations does not add much to the recommendation quality. Our belief is that this is related to the large number of edges in the network and the fact that people can be fans for different reasons. The other networks have larger impacts, with the explicitly stated social network performing better than the rest. We believe this is due to the relations expressing a type of "common ground" or agreement between the two parties.

4 Related Work

Recommender systems research originated in the late 1980's - early 1990's [Resnick *et al.*, 1994] and has since then become a ubiquitous topic found at almost every machine learning or information retrieval related conference.

(a) P@10

Type	P@10 10K	P@10	%
Friendship	$1.993E-3$	$1.847E-3$	7.9%
Comments	$3.551E-4$	$3.383E-4$	5.0%
Fans	$6.365E-4$	$6.342E-4$	0.4%
Diggs	$3.093E-4$	$2.845E-4$	8.7%

(b) MAP

Type	MAP 10K	MAP	%
Friendship	$5.154E-3$	$4.890E-3$	5.4%
Comments	$4.519E-3$	$4.417E-3$	2.3%
Fans	$5.208E-3$	$5.198E-3$	0.2%
Diggs	$4.493E-3$	$4.310E-3$	4.2%

Table 4: The Precision at 10 and Mean Average Precision values for our approach and for regular Collaborative Filtering for the same training and test datasets and the percental improvement.

More recently, much of the focus of the recommender systems community was on the Netflix Prize. Pilászy and Tikk [Pilászy and Tikk, 2009], presented provocative results showing that meta data related to movies is of little value when it comes to predicting movie ratings. Kirmenis and Birturk [Kirmenis and Birturk, 2008], on the other hand, show that a similar approach that utilizes user related meta data generates better recommendations than a metadata ignorant approach. A similar hybrid approach is evaluated by Lekakos and Caravelas [Lekakos and Caravelas, 2006], where similarity-based data is combined with its content-based counterpart to improve recommendations, with good results.

Similarly to the Netflix Prize dataset, the Movielens dataset, provided by the GroupLens[4] research lab, has been frequently used in recommender systems research. For instance, Herlocker et al. [Herlocker *et al.*, 2002] evaluated neighborhood-based recommendation using Movielens in order to create design guidelines for collaborative filtering-based recommenders. Rashid et al. [Rashid *et al.*, 2002] researched the problem every system encounters when a new user starts using the service. Which items to recommend, or to decide which few items will give the system the most information about the user.

Amatriain et al. [Amatriain *et al.*, 2009], pose that re-rating movies is of significantly higher value than rating new ones. They show how the amount of time that has passed since the original rating affects the users' new rating, and thus the quality of the recommendations.

Guy et al. [Guy *et al.*, 2009] create a system for recommending items based on a users' aggregated *familiarity* network. In this work, the familiarity network is created by assigning relations between users based on sources such as

[4]`http://www.grouplens.org/`

co-authorship of wiki pages within an organization's internal network, similar to the implicit networks studied in this paper. The results show that the familiarity network produces better recommendations than classical similarity based approaches. A similar approach is presented by Bonhard and Sasse [Bonhard and Sasse, 2006].

Another approach related to familiarity networks is the concept of trust-based recommendation. Golbeck and Hendler's [Golbeck and Hendler, 2006] present an approach based on explicitly defined trust gathered through the *FilmTrust*[5] movie recommendation website. FilmTrust asks its users to assign trust values to their peers, thus stating whose taste to follow and whose not to follow. They conclude that trust does add to the quality of the recommendations.

5 Conclusion and Future Work

In this paper we presented early stage results which indicate that the networks that users are part of contain latent information not present in the data found through ordinary user-based collaborative filtering methods. We showed, in a movie recommendation scenario, that the actions of users as well as their social networks are implicitly reflected in their rating behavior.

The work presented shows that there is much to gain by simple extensions of current standard algorithms. However, the approach needs to be extended and further researched in order to gain more insight into the different types of networks users can be part of, and how they affect the quality of recommendations. Also, combinations of networks, which we did not touch upon should be taken into consideration. Similarly, extending this research outside of the movie domain could provide a deeper understanding of network types and the users in them. Our current work focuses on combinations of several network types as well as the integration of demographic data, i.e. age, gender, etc.

The main contribution of our paper is an evaluation of different user-related (pseudo-) social networks, explicit and implicit. We have shown that, in a movie recommendation scenario, these types of networks appear to have an effect on the quality of recommender algorithms, even when implemented by very simple means.

6 Acknowledgments

The authors would like to express their gratitude to the Moviepilot team who contributed to this work with dataset, relevant insights and support. The work in this paper was conducted in the scope of the KMulE project which was sponsored by the German Federal Ministry of Economics and Technology (BMWi).

References

[Amatriain *et al.*, 2009] X. Amatriain, J. M. Pujol, N. Tintarev, and N. Oliver. Rate it again: increasing recommendation accuracy by user re-rating. In *RecSys'09*, 2009.

[Bonhard and Sasse, 2006] P. Bonhard and M. Sasse. Knowing me, knowing you using profiles and social networking to improve recommender systems. *BT Technology Journal*, 24(3), 2006.

[Gantz and Reinsel, 2010] J. Gantz and D. Reinsel. The digital universe decade are you ready?, 2010.

[Golbeck and Hendler, 2006] J. Golbeck and J. Hendler. FilmTrust: movie recommendations using trust in web-based social networks. In *CCNC'06*, volume 1, 2006.

[Guy *et al.*, 2009] I. Guy, N. Zwerdling, D. Carmel, I. Ronen, E. Uziel, S. Yogev, and S. Ofek-Koifman. Personalized recommendation of social software items based on social relations. In *RecSys'09*, 2009.

[Herlocker *et al.*, 2002] J. Herlocker, J. Konstan, and J. Riedl. Empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. *Informational Retrieval*, 5, 2002.

[Herlocker *et al.*, 2004] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22, 2004.

[Kirmenis and Birturk, 2008] O. Kirmenis and A. Birturk. A Content-Based user model generation and optimization approach for movie recommendation. In *Workshop on ITWP*. AAAI Press, 2008.

[Lekakos and Caravelas, 2006] G. Lekakos and P. Caravelas. A hybrid approach for movie recommendation. *Multimedia Tools and Applications*, 36(1-2), 2006.

[Montebello, 1998] M. Montebello. Information overload– an IR problem? *String Processing and Information Retrieval, International Symposium on*, 1998.

[Pilászy and Tikk, 2009] I. Pilászy and D. Tikk. Recommending new movies: even a few ratings are more valuable than metadata. In *RecSys'09*, 2009.

[Rashid *et al.*, 2002] A.M. Rashid, I. Albert, D. Cosley, S.K. Lam, S. McNee, J.A. Konstan, and J. Riedl. Getting to know you: Learning new user preferences in recommender systems. In *IUI'02*. ACM, 2002.

[Resnick *et al.*, 1994] P. Resnick, N. Iacovou, M. Sushak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *CSCW'94*, Chapel Hill, NC, 1994. ACM.

[Said *et al.*, 2009] A. Said, R. Wetzker, W.d Umbrath, and L. Hennig. A hybrid PLSA approach for warmer cold start in folksonomy recommendation. In *RecSys Workshop on RSWeb*, 2009.

[Said *et al.*, 2010a] A. Said, S. Berkovsky, and E. W. De Luca. Putting things in context: Challenge on context-aware movie recommendation. In *CAMRa'10*, 2010.

[Said *et al.*, 2010b] A. Said, E. W. De Luca, and S. Albayrak. How social relationships affect user similarities. In *IUI Workshop on SRS*, 2010.

[Siedler, 2010] MG Siedler. Netflix now 15 million users strong with over 60 percent of them streaming content. `http://techcrunch.com/2010/07/21/netflix-users/` (retrieved April, 2011), 2010.

[5]`http://trust.mindswap.org/FilmTrust/`

Modelling and Predicting Movements of Museum Visitors: A Simulation Framework for Assessing the Impact of Sensor Noise on Model Performance

Fabian Bohnert, Ingrid Zukerman, and David W. Albrecht
Faculty of Information Technology
Monash University, Australia
firstname.lastname@monash.edu

Timothy Baldwin
Dept. of Comp. Sci. and Soft. Eng.
The University of Melbourne, Australia
tb@ldwin.net

Abstract

We present a simulation framework to examine the impact of sensor noise on the performance of user models in the museum domain. Our contributions are: (1) models to simulate noisy visit trajectories as time-stamped sequences of (x, y) positional coordinates which reflect *walking* and *hovering* behaviour; (2) a discriminative inference model that distinguishes between hovering and walking on the basis of (simulated) noisy sensor observations; (3) a model that infers viewed exhibits from hovering coordinates; and (4) a model that predicts the next exhibit on the basis of inferred (rather than known) viewed exhibits. Our staged evaluation assesses the effect of these models (in combination with sensor noise) on inferential and predictive performance, thus shedding light on the reliability attributed to inferences drawn from sensor observations.

1 Introduction

The construction of models of visitors to public spaces, in particular museums, has been of interest to the user modelling and cultural tourism communities for some time [Cheverst *et al.*, 2002; Hatala and Wakkary, 2005; Stock *et al.*, 2007]. These models are used to predict visitors' interests in order to personalise the content of presentations, or make recommendations of locations (e. g., exhibits) to be visited. In most systems developed to date, these user models are acquired through the active participation of the visitors, e. g., by providing feedback through a device. This requirement imposes a burden on the visitors, which in turn may reduce the reliability of the obtained information, e. g., if visitors provide feedback only occasionally.

Recent advances in mobile computing and sensing technologies have enabled the instrumentation of physical public spaces, which in turn has enabled the automatic tracking of visitors' movements [Hazas *et al.*, 2004; Lassabe *et al.*, 2009; Philipose *et al.*, 2004]. Information regarding visitors' whereabouts and the time spent at different locations supports the automatic inference of visitors' interests and the prediction of their trajectories [Bohnert and Zukerman, 2009]. Clearly, inferences from positional and timing information are more indirect and uncertain than visitors' direct feedback. However, the information stream is reliable, as opposed to information obtained from visitors' direct participation.

In order to personalise content and generate recommendations on the basis of information provided by unobtrusive sensors (rather than from user participation), questions of interest include: (1) how to infer a visitor's viewed exhibits solely from sensor readings; and (2) how to predict the next exhibit(s) a visitor is likely to view. In this paper, we present a realistic simulation model which offers some insights to answer these questions, and may be employed to make decisions regarding the instrumentation of a space.

In previous research, we offered a simulation framework for investigating the impact of different sensing technologies on the predictive performance of user models [Schmidt *et al.*, 2009]. The aim was to provide a practical solution to the problem of assessing the accuracy of the user models that can be derived from a sensor-based system prior to actually deploying a particular technology. However, that work made strong simplifying assumptions that affected the realism of the framework, and hence the significance and usefulness of its results, viz: (1) sensors can detect, with some error, a single square (in a grid representation of the museum floor) where a visitor is *statically positioned* while viewing an exhibit i_k; and (2) the previously viewed exhibits $i_1, \ldots, i_{k-1}$ are known (not just the previous coordinates of a visitor) when predicting the next exhibit i_{k+1}. In reality, people tend not to remain stationary at an exhibit, and they certainly do not 'teleport' between squares on the floor. Rather, they *walk* between exhibits, and often *hover* around an exhibit to view it from different angles or distances. Thus, when sensing a visitor's movements in a museum, the best we can hope for is a time-stamped trajectory of (x, y) coordinates (sampled at a particular rate), where the observed coordinates diverge from the true positions of the visitor by some sensor error. As a result, the sequence of previously viewed exhibits cannot be known with certainty — at best a likely sequence of exhibits can be inferred from the sensor observations.

In this paper, we propose a simulation framework that eschews the above assumptions, significantly extending our previous work and the insights obtained from it. Specifically, our contributions are: (1) models to simulate noisy visit trajectories as time-stamped sequences of (x, y) positional coordinates which reflect *walking* and *hovering* behaviour; (2) a discriminative inference model that distinguishes between hovering and walking on the basis of noisy sensor observations; (3) a model that infers likely viewed exhibits from time-stamped sequences of hovering coordinates (instead of a single static grid square per exhibit as done in our previous work); and (4) a model that predicts the next exhibit on the basis of these inferred (rather than known) viewed exhibits. At present, we assume that the sensors can only track a visitor's position. However, our models may be extended to incorporate orientation information and occasional user feedback to improve the accuracy of inferences obtained from sensor readings, and hence the predictions of subsequent exhibits.

The research in this paper builds on the framework described by Schmidt *et al.* [2009], which comprises a predictive user model of exhibits to be viewed, and a spatial viewing model of positions from which each exhibit can be seen. Like Schmidt *et al.*, we evaluate our framework in the context of the Marine Life Exhibition at Melbourne Museum. In this paper, we augment the evaluations done by Schmidt *et al.*, presenting the results of a *staged* evaluation which examines the effect of different information-based models, in combination with sensor noise, on inferential and predictive performance.

This paper is organised as follows. Section 2 discusses related research. Section 3 briefly summarises the key components of our previous simulation framework. Our approach for simulating detailed coordinate-based visit trajectories is presented in Section 4, and our inference and prediction models are described in Section 5. The results of our evaluation are presented in Section 6, followed by concluding remarks in Section 7.

2 Related Research

The research community has initiated a wealth of projects that investigate user modelling and personalisation technology in the context of physical spaces. For example, in the museum domain, *HyperAudio* dynamically adapted hyperlinks and presented content to stereotypical assumptions about a visitor, and to what the visitor has already accessed through a mobile device and seems interested in [Petrelli and Not, 2005]. The *CHIP* project harnessed Semantic Web techniques to provide personalised access to digital museum collections both online and in the physical museum [Wang *et al.*, 2009]. This was done by using explicitly initialised user models. The *Kubadji* project investigated user and language modelling techniques that rely on mobile technology deployed in museums [Bohnert and Zukerman, 2009]. While the focus was on modelling visitors based on non-intrusive observations that can be derived from sensor readings, the project did not evaluate its models with real-world sensing technology.

In contrast to these projects, which did not employ real-world sensing technology, other research projects incorporated wireless technology or sensor networks. The *GUIDE* project developed a handheld tourist guide for visitors to the city of Lancaster, UK [Cheverst *et al.*, 2002]. It employed user models obtained from explicit user input to generate dynamic and user-adapted city tours, where the order of the visited locations could be varied. The project used wireless access points to stream content data to a user's device, but did not employ the wireless network to localise the user. The *PEACH* project developed technology which adapts its user model on the basis of both explicit user feedback and implicit observations of a user's interactions with a mobile device [Stock *et al.*, 2007]. This user model was used to generate personalised multimedia presentations for museum visitors. The *PEACH* project also explored simple localisation technology, but did not derive user modelling information from sensor readings. The augmented audio reality system for museums *ec(h)o* adapted its user model on the basis of a visitor's movements through the exhibition space and his/her interactions with the system [Hatala and Wakkary, 2005]. The collected user modelling data were used to deliver personalised information associated with exhibits via audio display. However, the project did not investigate the effect of localisation accuracy on the quality of the resultant user modelling information.

In contrast to the above research, this paper investigates the impact of using sensing technology as a means for gathering information about a user, i. e., to learn a user model. To this effect, we offer a simulation framework which generates noisy visit trajectories that reflect walking and hovering behaviour, and investigate the relationship between sensor noise and inferential and predictive user model performance.

3 Prerequisites

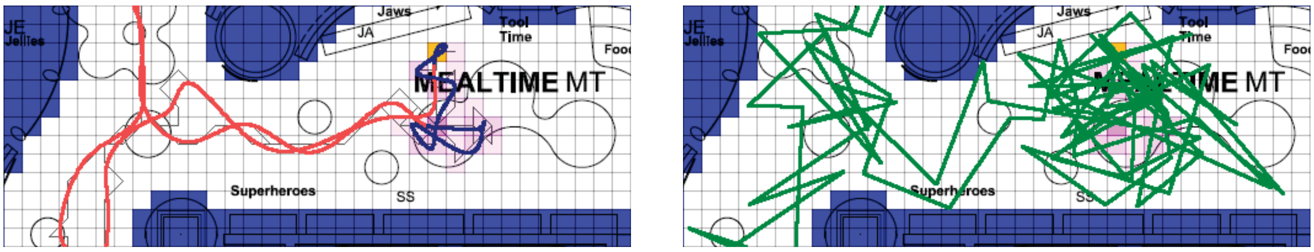
This section briefly summarises four key components of the simulation framework introduced by Schmidt *et al.* [2009], which is extended in this paper: (1) frequency-based *Transition Model*; (2) *Spatial Exhibit Viewing Model*; (3) generation of exhibit tours; and (4) generation of exhibit squares.

Frequency-based Transition Model. We use a frequency-based *Transition Model* to represent visitors' movements between museum exhibits [Bohnert *et al.*, 2008; Schmidt *et al.*, 2009]. This model, which is implemented as a 1-stage Markov model, estimates the transition probabilities $\mathrm{P}_{i,j}$ between exhibits i and j from frequency counts of exhibit transitions that are derived from observed visit trajectories. When estimating the transition probabilities, *additive smoothing* is applied in light of our small dataset of 44 observed trajectories (Section 6.1):

$$\hat{\mathrm{P}}_{i,j} = \frac{n_{i,j} + \alpha_i}{N_i + M\alpha_i} \quad \text{for } i, j = 1, \ldots, M$$

where $n_{i,j}$ counts the transitions from exhibit i to exhibit j, α_i is a smoothing constant, $N_i = \sum_{k=1,\ldots,M} n_{i,k}$ is the total number of times exhibit i was viewed, and M is the number of exhibits.

Spatial Exhibit Viewing Model. Our modelling framework employs a probabilistic model of the viewing areas for each exhibit in the museum space, which divides the space into a grid of squares (for the Marine Life Exhibition, the grid size is $47 \times 61 = 2,867$ squares, where a square is approximately 30 cm × 30 cm; Figure 1). The model specifies a discrete probability distribution which represents $\mathrm{P}(i \,|\, x, y)$, the probability of a visitor viewing each exhibit i from a square at position (x, y).



(a) Smooth representation (ground truth)

(b) Noisy representation ($\nu = 2$ metres)

Figure 1: Two representations of part of a simulated visitor pathway

Generation of exhibit tours. We generate tours of viewed exhibits as follows. Each tour begins at a fictitious *start exhibit* i_0 and ends at a fictitious *end exhibit* i_{end}. For each exhibit i_{k-1} already in the tour ($k = 1, 2, \ldots$), the next exhibit i_k is generated by sampling from a categorical distribution specified by the transition probabilities $\mathrm{P}_{i_{k-1}, i_k}$. This step is repeated for each added exhibit i_k until the end exhibit i_{end} is reached.

In addition to this sequence of exhibits, our walking/hovering model (Section 4) requires the time that a visitor spends at each viewed exhibit. We generate a viewing time T_i at exhibit i by randomly drawing from an exponential distribution, i. e., $T_i \sim \mathrm{Exp}(\lambda_i)$, where the average viewing time λ_i at each exhibit i is estimated by maximum likelihood from the 44 observed tours in the Marine Life Exhibition dataset.

Generation of exhibit squares. Once a tour of exhibits has been simulated, Schmidt *et al.* [2009] generate a *single* viewing square at position (x, y) for each viewed exhibit i in the tour. This is done by sampling from the categorical distribution $\mathrm{P}(x, y \,|\, i)$ over all exhibit squares, where $\mathrm{P}(x, y \,|\, i)$ is derived by applying Bayes' theorem to the viewing probabilities $\mathrm{P}(i \,|\, x, y)$ obtained from the Spatial Exhibit Viewing Model.

In this work, we use Schmidt *et al.*'s model to generate the *first* hovering square for each viewed exhibit (Section 4.2).

4 Simulation of Coordinate-based Visitor Pathways

The previous section outlined our method for generating exhibit tours with a single static grid square per exhibit. In this section, we simulate (smooth and noisy) coordinate-based visit trajectories which reflect two types of behaviour: *walking* between exhibits, and *hovering* at exhibits. Our approach comprises the following four steps, which are described below: (1) generation of connected paths of *walking* squares between exhibits; (2) generation of connected paths of *hovering* squares to simulate viewing behaviour at exhibits; (3) smoothing of the obtained square trajectory; and (4) simulation of noisy sensor observations from this smooth pathway representation.

Figure 1 depicts two representations of part of a simulated visit trajectory (we show the part for the Tool Time exhibit in the Mealtime section of the Marine Life Exhibition). Figure 1(a) shows the trajectory obtained after simulation (walking is represented by a red/grey line, hovering is represented by a blue/dark-grey line on pink/shaded squares, and wall squares are coloured in blue/grey), and Figure 1(b) is the representation obtained by applying Gaussian sensor noise at a level of $\nu = 2$ metres.

4.1 Generating Walking Squares

In Section 3, we generated one viewing square for each exhibit in a visitor's tour. However, visitors do not simply teleport between squares. To produce a more realistic continuous visit trajectory, we must build a path that links these squares. At first glance, it seems that a shortest-path algorithm may be used for this task. However, trajectories generated in this way exhibit an unnatural level of repetition and purposefulness, tending to run directly along exhibition walls. In practice, visitors tend to move more erratically. To simulate these behaviours, we incorporate stochastic effects into the shortest-path procedure. Specifically, we model the probability of moving into a square as being proportional to the probability of viewing the destination exhibit from this square, moderated by the visitor's propensity to avoid walls and to meander. Our approach uses parameters that control two behavioural aspects of visitors: (1) how erratic or purposeful their movement is; and (2) their propensity to avoid walls.[1] These considerations are implemented as follows.

Assume we want to generate a sequence of walking squares to connect two exhibits i and j in a tour. Let (x_s, y_s) denote the end square of exhibit i (i. e., the *source square*), and (x_d, y_d) the starting square of exhibit j (i. e., the *destination square*). Also, treating diagonal squares as adjacent, let the *candidate squares* of a square (x, y) be the eight squares surrounding this square. We start by employing Dijkstra's algorithm [Dijkstra, 1959] to generate a distance matrix $\boldsymbol{D}$ whose elements $D_{x,y}$ correspond to the shortest-path distances from each square (x, y) to the destination square (x_d, y_d). Then, we generate a sequence of walking squares as follows. For each square (x_n, y_n) (starting from the source square (x_s, y_s)), the next square (x_{n+1}, y_{n+1}) that a visitor moves into while walking is sampled from among

[1] In our evaluation, we use fixed parameter values. Alternatively, one could sample the values for each trajectory simulation. Also, certain parameter values in combination with different transition models may yield different types of museum visitors, e. g., the *ant*, *fish*, *butterfly* and *grasshopper* types [Véron and Levasseur, 1983; Zancanaro *et al.*, 2007].

(x_n, y_n)'s eight candidate squares, provided that the move does not take the visitor farther away from (x_d, y_d) (the distance information is obtained from $\boldsymbol{D}$). In this procedure, the sampling is performed from a categorical distribution over the eight candidate squares, whose probabilities are proportional to the probabilities of viewing the destination exhibit from each square, moderated by the visitor's propensity to avoid walls and to meander (the probabilities are zero for the squares that take the visitor farther away from (x_d, y_d)). The visitor moves in this fashion until (x_d, y_d) is reached. At that point, the trajectory between (x_s, y_s) and (x_d, y_d) is complete, and timestamps are iteratively added to the trajectory assuming a constant walking speed v_w for the visitor.

4.2 Generating Hovering Squares

Once at an exhibit, visitors usually observe the exhibit for some time before moving on to the next one. Additionally, visitors typically do not remain static, but move around to examine the exhibit from different angles and distances. This so-called *hovering* behaviour is included in our simulation framework by varying the movement model described in Section 4.1, so that a visitor is more likely to move towards a square from which the exhibit is more likely to be viewed, but may not move at all.

Timestamps are added to the generated hovering squares assuming a hovering speed of $v_h < v_w$ (as for the walking case, we assume a constant hovering speed). The hovering behaviour continues until the sampled viewing time T_i for the current exhibit i is exceeded (viewing time sampling is described in Section 3).

4.3 Smoothing the Square Trajectory

To obtain a smooth positional tour representation from a time-stamped trajectory of squares, i. e., $(\langle t_n, x_n, y_n\rangle; n = 1, 2, \ldots)$, we fit piecewise cubic splines to the coordinate-individual trajectories $\langle t_n, x_n\rangle$ and $\langle t_n, y_n\rangle$ (one piecewise cubic spline each). We do this by applying the `splinefit` package from the Matlab Central File Exchange [Lundgren, 2007]. This approach uses the method of least squares to fit splines with reduced degrees of freedom (we reduce the number of spline pieces by 70% compared to direct interpolation), and generates a smooth representation of the trajectory in the sense that (x, y), $(\dot{x}, \dot{y})$ and $(\ddot{x}, \ddot{y})$ are all continuous in time.

The resultant representation may be interpreted as a continuous positional representation of the visit trajectory, enabling us to obtain a visitor's position at any point in time. Figure 1(a) depicts part of one such smooth visit trajectory.

4.4 Simulating Sensor Noise

The visit trajectories obtained so far are smooth and continuous. However, in practice, any trajectory-based input to a user modelling system would be acquired through sensors that deliver only a visitor's *approximate* position (due to measurement error) at a certain sampling rate.

In this paper, we explore sensor noise that may be attributed to *range-based positioning technology*, e. g., WiFi and ultra-wide band (UWB) [Zhao and Guibas, 2004]. We follow a widely accepted model for sensor noise in this setting, and assume that the *measured* coordinates (x', y') are obtained by distorting the true coordinates (x, y) through additive Gaussian noise and sampling at regular time intervals (for our experiments, we use a constant sampling rate of one second). Specifically, the measured coordinates are found by sampling from a bivariate normal distribution $\mathrm{N}((x, y), \sigma^2\boldsymbol{I})$ with mean (x, y) and covariance $\sigma^2\boldsymbol{I}$, where σ is a constant which reflects the expected accuracy of the sensing infrastructure, and $\boldsymbol{I}$ is the identity matrix. For example, if the infrastructure is able to deliver positions within an accuracy level of ν metres 95% of the time, then $\sigma = \nu/2$ would be a suitable value, as this places approximately 95% of the probability mass within the circle defined by $(x' - x)^2 + (y' - y)^2 = \nu^2$. Figure 1(b) depicts part of a noisy visit trajectory which was sampled by following this procedure for the pathway shown in Figure 1(a) at a sampling rate of one second with $\nu = 2$ metres.

5 Inference and Prediction of Viewed Exhibits from Positional Coordinates

When information on a visitor's movements is automatically gathered through sensors, all that is available is a sequence of (typically noisy) time-stamped (x, y) coordinates (Section 4.4).[2] Assuming that we have a method for detecting whether a visitor is hovering (and hence viewing an exhibit), we can decompose the complete (x, y) sequence into subsequences of (x, y) coordinates that pertain to hovering behaviour (Section 5.1). From these, we can infer which exhibit the visitor is viewing (Section 5.2), and employ a model to predict which exhibit the visitor is likely to view next on the basis of this information (Section 5.3).

5.1 Classification-based Inference of Walking and Hovering

To infer walking and hovering behaviour from positional (x, y) coordinates, we employ a window-based approach. We first derive indicative features from a window comprising the previous ω sensor observations, and then provide these features to a purpose-trained classifier for inference. The output of this binary classifier is a label which indicates whether a visitor's activity is walking or hovering.

Prior to deriving the features, we smooth the noisy sensor observations $\langle t, x, y\rangle$ by fitting piecewise cubic splines to the $\langle t, x\rangle$ and $\langle t, y\rangle$ trajectories [Lundgren, 2007], and evaluating these splines at the original timestamps (similarly to Section 4.3). Using the resultant smoothed sensor observations, we compute the following feature set of size $2\omega + 7$ that pertains to (non-directional) velocity and acceleration:

- $\omega - 1$ velocities (each of them calculated as the length of one of the $\omega - 1$ velocity vectors, which in turn are derived from the ω smoothed positional coordinates from within the window)
- Minimum and maximum of the $\omega - 1$ velocities
- Mean and median of the $\omega - 1$ velocities
- Standard deviation of the $\omega - 1$ velocities

[2] For simplicity of notation, we use (x, y) instead of (x', y') in the remainder of the paper to denote the measured noisy coordinates.

- $\omega-2$ accelerations (each of them calculated as the length of one of the $\omega - 2$ acceleration vectors, which in turn are derived from the $\omega - 1$ velocity vectors)
- Minimum and maximum of the $\omega - 2$ accelerations
- Mean and median of the $\omega - 2$ accelerations
- Standard deviation of the $\omega - 2$ accelerations

In our experiments (Section 6), we use support vector machines (SVM) to train the classifiers. We employ C-SVC SVMs with an RBF kernel from LIBSVM [Chang and Lin, 2001], using features derived from the previous five (x, y) observations ($\omega = 5$).

5.2 Probability-based Inference of Exhibits

In this section, we describe how we infer the exhibits most likely viewed by the visitor while hovering.

After inferring a visitor's activity (i. e., walking or hovering) for each sensor observation $\langle t, x, y\rangle$, we extract from the complete (x, y) sequence the sub-sequences of (x, y) coordinates that correspond to hovering behaviour. For each sub-sequence of hovering-labelled (x, y) coordinates, we then calculate the following exhibit scores:

$$\mathrm{score}(i) = \prod_{(x,y)} \mathrm{P}(i\,|\,x, y) \quad \text{for all exhibits } i \qquad (1)$$

where $\mathrm{P}(i\,|\,x, y)$ is the probability of a visitor viewing exhibit i while hovering at position (x, y) (Section 3). To smooth out possible errors introduced in the classification step (Section 5.1), we delete walking labels that separate two consecutive sub-sequences of hovering labels for which the same exhibit has the highest score. We also remove hovering-labelled sub-sequences of length 1 (the exhibit scores of any affected sub-sequences of hovering labels are recomputed). Finally, all scores are normalised to obtain probabilities.

For each sub-sequence of hovering labels, this procedure yields a probability distribution which specifies how likely a visitor is to view each exhibit.

5.3 Model-based Prediction of Exhibits

Once the viewed exhibits are inferred, we can use this information to predict a visitor's next exhibit for each (x, y) position at which the visitor is hovering.[3] However, as seen in the previous section, there is some uncertainty regarding which exhibit the visitor is actually viewing. We therefore use the *Weighted* approach described by Schmidt *et al.* [2009] for predicting the next exhibit from positional information. For each possible next exhibit i, the *Weighted* approach estimates $\mathrm{P}_{\text{next}}(i\,|\,x, y)$ as the weighted average of the transition probabilities $\mathrm{P}_{j,i}$ from each possible current exhibit j to exhibit i. The weights are the probabilities $\mathrm{P}(j\,|\,x, y)$ of viewing exhibit j when standing within the square at position (x, y) (Section 3).

$$\hat{\mathrm{P}}_{\text{next}}(i\,|\,x, y) = \sum_{j=1}^{M} \{\, \mathrm{P}(j\,|\,x, y) \times \mathrm{P}_{j,i} \,\}$$

[3]Predictions of a visitor's next exhibits can be combined with predictions of the personally interesting exhibits to generate recommendations of exhibits that may be overlooked if the predicted next exhibits are actually visited.

In this calculation, the transition probabilities $\mathrm{P}_{j,i}$ are derived from the information provided by the Transition Model in Section 3 by setting to zero the columns of the transition matrix that pertain to the already viewed exhibits, and renormalising each row of the matrix to 1.[4]

6 Evaluation

This section presents our data collection method and datasets, and describes our experiments and results.

6.1 Data Collection and Datasets

Our dataset of real-world exhibit tours was obtained at the Marine Life Exhibition at Melbourne Museum. It consists of a (manually collected) record of the exhibits viewed by 44 visitors, and the viewing times at the exhibits. On average, each visitor viewed 7.2 of the $M = 22$ exhibits. The data for the viewing model described in Section 3 were obtained separately, by manually annotating a grid-based map to record the positions of visitors to the exhibition.

These data were used together with the method from Section 4 to generate 1000 simulated visits, where each visit comprises time-stamped sequences of (typically noisy) (x, y) coordinates at different noise levels — each element consisting of $\langle t, x, y\rangle$. These 1000 simulated visits are the basis for our evaluation. When generating the visits, we assumed a constant walking speed of $v_w = 3$ km/h and a hovering speed of $v_h = 1$ km/h. Also, we used a sampling rate of one observation per second.

Current range-based positioning systems are often based on processing radio signals, e. g., WiFi and ultra-wide band (UWB). WiFi-based technology typically achieves accuracy levels of 2 to 3.5 metres [Bahl and Padmanabhan, 2000; Lassabe *et al.*, 2009], while future UWB-based systems are expected to achieve accuracy levels of up to 0.15 metres [Hazas *et al.*, 2004]. We therefore considered accuracy levels of $\nu = 0$ to 4.5 metres when generating the visits.

6.2 Experiments and Results

To evaluate our models, we applied *bootstrapping* [Mooney and Duval, 1993] as follows. The 1000 generated visits were split into a training set of 100 visits and a test set of 900 visits. 200 *bootstrap* samples were then generated from the test set, with each bootstrap sample being constructed by sampling from the 900 visits with replacement (200 is the recommended upper bound on the number of samples for bootstrapping [Mooney and Duval, 1993]). The training set remained the same for all samples. Our results are averaged over the bootstrap samples.[5]

We conducted three experiments with these training and test sets: (1) walking/hovering classification; (2) inferring exhibits from positional hovering coordinates; and (3) predicting the next exhibit. All performance differences between models were found to be statistically significant with

[4]Our observations indicate that visitors rarely return to previously viewed exhibits. Hence, we focus on unseen exhibits.

[5]We employed bootstrapping, because only the test data varies for this technique, compared to cross validation which conflates the variation in the training and test data.

Table 1: Inference models and their experimental conditions

Models	**Time & (x, y)**	**Walk/Hover**	**Exhibits** Previous	**Exhibits** Current
TL_{all}	sequence of $\langle t, x, y \rangle$	Inferred	Inferred	Inferred
TLA_{all}	sequence of $\langle t, x, y \rangle$	Given	Inferred	Inferred
$\mathrm{Exh}_{prev}\mathrm{TLA}_{curr}$	sequence of $\langle t, x, y \rangle$	Given	Given	Inferred
Schmidt et al.	*one* $\langle x, y \rangle$ *per exhibit*	*N/A*	*Given*	*Inferred*
Exh_{all}	sequence of $\langle t, x, y \rangle$	Given	Given	Given

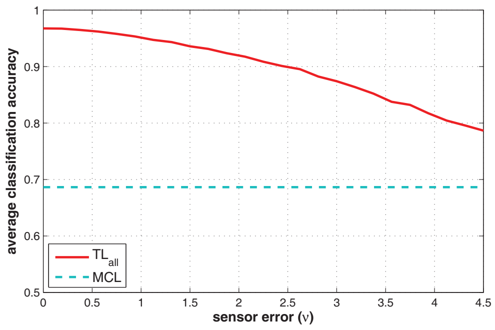


Figure 2: Average walking/hovering classification accuracy against sensor error

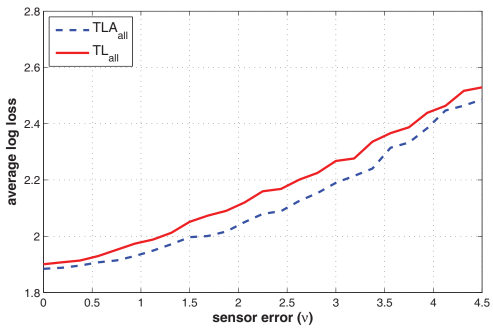


Figure 3: Average log loss of actually viewed exhibits against sensor error

$p \ll 0.001$ (evaluated using two-tailed paired t-tests on the bootstrap samples).

Table 1 summarises the models used in our experiments, indicating the inferred versus given information (only the first two models, i. e., those with grey background, are used in our first two experiments). The top model TL_{all} (*Time-Location* for *all* observations) is the most realistic, as its information is akin to that obtained from sensor readings (i. e., a sequence of time-stamped (x, y) coordinates). The models then become progressively less realistic, starting with TLA_{all} (*Time-Location-Action* for *all* observations), where the walking/hovering labels are considered given, up to Exh_{all}, where the walking/hovering labels, previous exhibits and current exhibit are given. To contextualise our work, Table 1 also shows Schmidt *et al.*'s model [Schmidt *et al.*, 2009] (typeset in italics), but its results are excluded from our evaluation, as it does not model trajectories or temporal information.

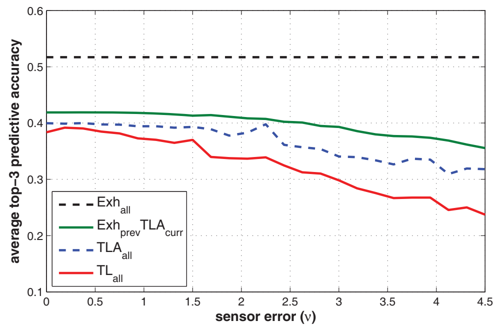
Walking/hovering classification. To evaluate the performance of our walking/hovering classification method (Section 5.1), we gave as input sequences of times and positions ($\langle t_n, x_n, y_n \rangle; n = 1, 2, \ldots$). For each walking/hovering classification, we considered the five positional observations made within the last four seconds ($\omega = 5$). As visitors hover slightly less than 69% of the time, and walk between exhibits for the rest of the time, we under-sampled the hovering portion of the training data to balance the classes.[6]

Figure 2 depicts classification accuracy as a function of sensor error, where the majority class baseline (MCL) assumes that a person is always hovering (the results are averaged over the 22 exhibits of the Marine Life Exhibition). Our results show that for no sensor error, our SVM classifier (TL_{all}) is able to infer whether a visitor is walking or hovering with approximately 97% accuracy. Classification accuracy decreases to about 88% as the sensor error increases to 2.75 metres (the middle of the range for WiFi technology).

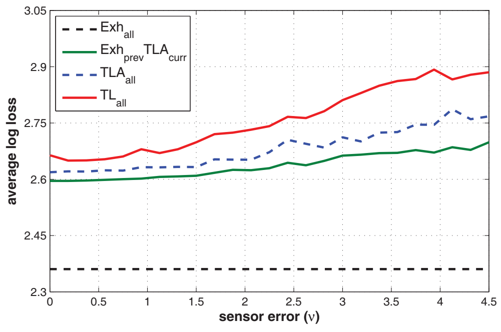
Inferring exhibits from positional hovering coordinates. To evaluate the performance of our mechanism for inferring the sequence of visited exhibits, we gave as input sequences of times and positions ($\langle t_n, x_n, y_n \rangle; n = 1, 2, \ldots$) and walking/hovering labels (one label for each element in a sequence). The probabilities of viewed exhibits were calculated once for given (known) walking/hovering labels, and once for labels inferred using the SVM classifier (Section 5.1). The inferences were made as described in Section 5.2, and resulted in a probability distribution of the exhibit being viewed by a visitor for each sub-sequence of hovering labels.

Figure 3 depicts the average *log loss* (negative log of the probability of the actually viewed exhibit), averaged over the 22 exhibits, as a function of sensor error. The figure compares the performance obtained when the walking/hovering labels are inferred (TL_{all}) with that obtained when the labels are given (TLA_{all}). It is worth noting that the comparison was done for the timestamps where the inferred and given hovering labels overlap, but the exhibit probabilities used for the

[6]We under-sampled the larger class, rather than over-sampling the smaller class, in order to retain the variance of the latter class. We also experimented with unbalanced data, but the performance was inferior to that obtained with the balanced data.



(a) Average top-3 accuracy



(b) Average log loss

Figure 4: Predictive performance of the four models against sensor error

comparison were calculated for all the inferred or given hovering labels in each continuous sub-sequence of hovering labels. This explains the (expected) slight drop in performance for inferred hovering labels, since, as seen in the first experiment, the inferred labels are sometimes wrong. Also, as expected, performance deteriorates as sensor error increases.

Predicting the next exhibit. This experiment determines the effect of different assumptions regarding available information on predictive accuracy. We consider our four models from Table 1, whose information ranges from time-stamped positional sensor logs (TL_{all}) to sequences of viewed exhibits (Exh_{all}). In line with Schmidt *et al.* [2009], for all four models, the next exhibit was predicted using the transition matrix learned from the 44 tours observed at the Marine Life Exhibition (Section 3). For Exh_{all}, we used the transition matrix directly (the transition probabilities for previously visited exhibits were set to zero), while for the other models, we used the *Weighted* approach described in Section 5.3.

Figures 4(a) and 4(b) show, respectively, the average top-3 accuracy and average log loss for various levels of sensor error for the four models described in Table 1 (the results are averaged over the 22 exhibits). For this experiment, *log loss* is defined as the negative log of the probability with which the exhibit actually viewed *next* is predicted, and *top-3 accuracy* measures how often the exhibit actually viewed next is one of the three exhibits predicted with the highest probability. We employ top-3 rather than top-1 accuracy because the top probabilities are often quite similar due to the physical layout of the exhibition. As seen in the figures, the higher the uncertainty about a visitor's behaviour and the higher the sensor error, the lower the accuracy and the higher the log loss (statistically significant). Note that Exh_{all} is invariant to sensor noise, as all the information is assumed given (Table 1). Interestingly, the differences in performance between the three lower-information models (TL_{all}, TLA_{all} and $\mathrm{Exh}_{prev}\mathrm{TLA}_{curr}$) are relatively small, and their performance profiles are quite flat up to $\nu = 1.5$ metres, diverging slightly from there on. The creditable performance up to $\nu = 1.5$ metres means that one can expect acceptable predictive performance from sensor-based systems.

7 Conclusions

This paper offered a realistic model of sensor-based information, significantly extending the work of Schmidt *et al.* [2009]. Our framework enables us to study the impact of different assumptions regarding sensor noise and available sensor information on inferential performance regarding viewed exhibits. The accuracy of these inferences in turn affects the performance of user models, viz models of visitors' interests and of exhibits they are likely to visit. As expected, predictive performance deteriorates for every experimental parameter that is inferred (rather than given), and also as sensor error increases. However, interestingly, performance remains quite stable for sensor noise up to 1.5 metres, which is an encouraging result for real-world systems.

Our inferential and predictive models in combination support the generation of recommendations of exhibits that may be of interest but are likely to be missed. Our models may also be used to influence the strength of recommendations as a function of the reliability of the information on which the recommendations are based. An additional application of our results is in guiding the layout of sensing devices in a museum, e. g., it may be advantageous to place more devices in locations where the inferences are more uncertain.

Acknowledgements

This research was supported in part by grant DP0770931 from the Australian Research Council. The authors thank Daniel F. Schmidt for his involvement at early stages of this research, Liz Sonenberg and Carolyn Meehan for fruitful discussions and their support, and David Abramson, Jeff Tan and Blair Bethwaite for their assistance with the computer cluster.

References

[Bahl and Padmanabhan, 2000] Paramvir Bahl and Venkata N. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *Proceedings of the 19th Annual Joint IEEE Conference on Computer Communications (INFOCOM-00)*, pages 775–784, 2000.

[Bohnert and Zukerman, 2009] Fabian Bohnert and Ingrid Zukerman. Non-intrusive personalisation of the museum experience. In *Proceedings of the 17th International Conference on User Modeling, Adaptation, and Personalization (UMAP-09)*, pages 197–209, 2009.

[Bohnert *et al.*, 2008] Fabian Bohnert, Ingrid Zukerman, Shlomo Berkovsky, Timothy Baldwin, and Liz Sonenberg. Using interest and transition models to predict visitor locations in museums. *AI Communications*, 21(2-3):195–202, 2008.

[Chang and Lin, 2001] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[Cheverst *et al.*, 2002] Keith Cheverst, Keith Mitchell, and Nigel Davies. The role of adaptive hypermedia in a context-aware tourist GUIDE. *Communications of the ACM*, 45(5):47–51, 2002.

[Dijkstra, 1959] Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[Hatala and Wakkary, 2005] Marek Hatala and Ron Wakkary. Ontology-based user modeling in an augmented audio reality system for museums. *User Modeling and User-Adapted Interaction*, 15(3-4):339–380, 2005.

[Hazas *et al.*, 2004] Mike Hazas, James Scott, and John Krumm. Location-aware computing comes of age. *IEEE Computer*, 37(2):95–97, 2004.

[Lassabe *et al.*, 2009] Frederic Lassabe, Philippe Canalda, Pascal Chatonnay, and François Spies. Indoor Wi-Fi positioning: Techniques and systems. *Annals of Telecommunications*, 64:651–664, 2009.

[Lundgren, 2007] Jonas Lundgren. SPLINEFIT, 2007. Software available at `http://www.mathworks.com/matlabcentral/fileexchange/13812-fit-a-spline-to-noisy-data`.

[Mooney and Duval, 1993] Christopher Z. Mooney and Robert D. Duval. *Bootstrapping: A Nonparametric Approach to Statistical Inference*. Sage Publications, Newbury Park, CA, USA, 1993.

[Petrelli and Not, 2005] Daniela Petrelli and Elena Not. User-centred design of flexible hypermedia for a mobile guide: Reflections on the HyperAudio experience. *User Modeling and User-Adapted Interaction*, 15(3-4):303–338, 2005.

[Philipose *et al.*, 2004] Matthai Philipose, Kenneth P. Fishkin, Mike Perkowitz, Donald J. Patterson, Dieter Fox, Henry Kautz, and Dirk Hahnel. Inferring activities from interactions with objects. *IEEE Pervasive Computing*, 3(4):50–57, 2004.

[Schmidt *et al.*, 2009] Daniel F. Schmidt, Ingrid Zukerman, and David W. Albrecht. Assessing the impact of measurement uncertainty on user models in spatial domains. In *Proceedings of the 17th International Conference on User Modeling, Adaptation, and Personalization (UMAP-09)*, pages 210–222, 2009.

[Stock *et al.*, 2007] Oliviero Stock, Massimo Zancanaro, Paolo Busetta, Charles Callaway, Antonio Krüger, Michael Kruppa, Tsvika Kuflik, Elena Not, and Cesare Rocchi. Adaptive, intelligent presentation of information for the museum visitor in PEACH. *User Modeling and User-Adapted Interaction*, 18(3):257–304, 2007.

[Véron and Levasseur, 1983] Eliséo Véron and Martine Levasseur. *Ethnographie de l'Exposition*. Bibliothèque Publique d'Information, Centre Georges Pompidou, Paris, France, 1983.

[Wang *et al.*, 2009] Yiwen Wang, Lora Aroyo, Natalia Stash, Rody Sambeek, Yuri Schuurmans, Guus Schreiber, and Peter Gorgels. Cultivating personalized museum tours online and on-site. *Interdisciplinary Science Reviews*, 34(2):141–156, 2009.

[Zancanaro *et al.*, 2007] Massimo Zancanaro, Tsvika Kuflik, Zvi Boger, Dina Goren-Bar, and Dan Goldwasser. Analyzing museum visitors' behavior patterns. In *Proceedings of the 11th International Conference on User Modeling (UM-07)*, pages 238–246, 2007.

[Zhao and Guibas, 2004] Feng Zhao and Leonidas Guibas. *Wireless Sensor Networks: An Information Processing Approach*. Morgan Kaufmann, 2004.

IntWEB: An AI-Based Approach for Adaptive Web

Oznur Kirmemis Alkan, Pinar Senkul
Middle East Technical University, Computer Engineering Department, Ankara , Turkey
{oznur.kirmemis, senkul}@ceng.metu.edu.tr

Abstract

The World Wide Web is an endless source of information, which is mainly represented in the form of Web pages. The way that the users browse the Web depends on both user-oriented factors like the information the users are seeking for and site-oriented factors like the attractiveness of the Web sites, the structure or more specifically, the navigational organization. However, the site's design changes through time, due to factors like including more information about items or introducing new items. In addition, the user's needs and preferences also change, which together bring difficulties for building Web sites that best suit users' needs. Therefore, it is a very important and challenging task to adapt the Web sites automatically in order to facilitate users' navigation in the Web site. This paper proposes a solution to adapt Web sites structural organization according to users' navigation patterns. In the proposed solution, the adaptive web problem is formulated as a classical AI search problem, and a novel Hill Climbing based solution is devised. The proposed solution is realized in a framework, namely IntWEB. The technique is applied to a real-life case study and results are discussed in order to evaluate IntWEB's performance.

1 Introduction

There is a huge growth of information sources on the Internet every day and together with this growth, the user base also grows. In such a situation, the necessity of managing this information for large number of users with possibly diverse needs arises. Automatic personalization is the key technique that is utilized for developing solutions in such situations. Different personalization methods have been utilized by different solutions in order to deliver and present relevant information for individual users. For instance, personalization can be needed for developing information systems that act according to user preferences for WAP-enabled devices [Cotter and Smyth, 2000], or adapting web sites for users' interests, which is the problem discussed in this paper.

The World Wide Web, as an endless source of information, is mainly represented in the form of web pages. The way that the users browse the Web depends both on the information that they are searching for and the navigational structure that the Web site provides to its users. Therefore, factors such as the structure of the Web sites, or more specifically, their navigational organization affect users' tendency for Web surfing. On the other hand, users may not be seeking any information, but they may just be examining some product information. In such cases, attractiveness of the Web site and how well it meets user's taste become even more important in order to facilitate the navigation and keep the user in the Web site as long as possible. Therefore, Web sites design should in some way get adapted to its users' needs. Adaptive Web sites research emerged from these requirements.

Adaptive Web site is an attractive and challenging topic and the definition of this problem is provided in [Perkowitz and Etzioni, 1998] as follows: "*adaptive Web sites are the Web sites that automatically improve their organization and presentation by learning from visitor access patterns*" Adapting Web sites is considered as a challenging task since the adaptation process is built on the preferences of users; however, preferences of users are generally highly diverse and demanding. In other words, each of the visitors of a Web site may be searching for something different, and each may have unique needs or concerns. One solution to this problem can be clustering users according to their usage patterns and then creating different user interfaces for each user group. However, handling all these interfaces can be very difficult and may not be preferred by sites' managers. Another solution can be handling as many users' navigational patterns as possible so as to satisfy most of the users' preferences. Although this will dissatisfy some of the users, it can be considered a more feasible solution than creating many interfaces.

Adaptive Web sites have been studied by different disciplines like Machine Learning and Data Mining to be able to provide solutions from the ideas from these fields. In addition, Perkowitz and Etzioni mentioned and discussed in their work [Perkowitz and Etzioni, 1998] that, "*the goal of creating self-improving Web sites requires breakthroughs in different areas of Artificial Intelligence (AI)*", and they discuss several aspects of adaptive web problem and try to show that the problem is in fact an AI challenge. In addition, in [Perkowitz and Etzioni, 2000], they view automatic improvement of a Web site as a search problem in the space of possible Web sites, and they further mention that different approaches to creating adaptive Web sites correspond to different ways of searching the space. They also present a solution that relies on conceptual clustering techniques to solve adaptive Web problem.

In this work, we propose an AI-based solution for adapting Web sites according to users' preferences. The solution utilizes Hill Climbing search algorithm [Russell and Norvig, 2003] and the problem formulation through providing state description, actions, evaluation function and the adapted Hill Climbing solution is given in detail. The proposed technique is realized in a framework named IntWEB (Intelligent Web). The proposed solution is applied to a real-life case study, more specifically, the department's Web site, in order to evaluate the results of the adaptation system proposed.

The rest of the paper is organized as follows: In Section 2, information about the related work in this area is presented. In Section 3, INTWeb is presented in three parts; problem formulation, system components that are designed to realize the approach and the algorithm. In Section 4, the evaluation of the system is presented and the results are discussed. Finally, concluding remarks and future directions of research are given in Section 5.

2 Related Work

The main goal of any user-adaptive system is to provide users with what they need without asking it to the users explicitly [Mulvenna, Anand and Buchner, 2000]. Therefore, automatic personalization is the main technology in such systems. For the case of adaptation of web sites to user preferences, web servers hold the common and very rich source of knowledge about user navigation patterns and interests. Large amounts of data can be collected from server log files, which include the clickstream (web usage) data. Personalization on the Web requires to take full advantage of the flexibility provided by this data, and to effectively use the discovered models in an automatic personalization system.

The process of personalization can be viewed as an application of data mining through applying steps of a general data mining solution like data collection, pre-processing, pattern discovery and evaluation in an off-line mode, and finally the deployment of the knowledge in real-time to mediate between the user and the Web [Mobasher, 2007]. Data mining techniques have been extensively applied in order to extract usage patterns in several studies [Mobasher, Cooley, and Srivastava, 2000], [Mobasher, Dai, Kuo, and Nakagawa 2001], [Nasraoui and Petenes 2003]. In e-commerce also, researchers of adaptive web are increasingly using clickstream data, which was originally collected for website performance analyses. For instance, clickstream data has been used to find out the behaviors of customers across websites through user-centric scenarios [Goldfarb 2002] and within specific websites through site-centric scenarios [Sismeir and Bucklin, 2004]. For the case of site-centric scenarios, some studies focused on single visits to a given website, whereas some others studied multiple visits.

One example of well known adaptive web solution is the WebWatcher [Joachims, Freitag, and Mitchell, 1997] which suggests links to users based on the online behavior of other users. Initially, the system ask the users to provide the information regarding their reason to enter the Web site, or in other words, what they are seeking for towards using the Web site. In addition to this information, before users leave the Web site, they are asked to provide whether they have found what they were looking for. After that, those users' navigation paths together with their feedback are used in order to create suggestions for future visitors that seek the same content. The resulting suggestions are presented by highlighting the already existing hyperlinks.

Avanti project [Fink, Kobsa and Nill, 1996] is another system that also requires user's explicit feedback so as to adapt Web sites. In that project, the user's final objective as well as his next step is aimed to be discovered. A model for the user is built, partly based on the information that the user provides about him and also from his navigation paths. Using this information, direct links to pages, that are considered to be liked by the users, are presented to them. In addition to the consideration of specific likeliness, hyperlinks that lead to pages of potential interest to each visitor are highlighted. A drawback of both the WebWatcher and the Avanti project is that, they require the users to be in active participation with the system in the adaptation process by asking them to provide information about themselves.

The Footprints system [Wexelblat and Maes, 1999] does not need explicit user feedback, which is an advantage of it over the described systems above. It only uses the navigation paths of the users. The Footprints system does not perform user identification and the most frequent navigation paths are presented to the visitor in the form of maps and also the percentage of people who have followed those maps are displayed next to each link. Possible enhancements to the Web site are presented as a list of suggestions to users.

As mentioned in the previous section, Perkowitz and Etzioni presented Web site adaptation as an AI problem in [Perkowitz and Etzioni, 1997]. They discuss that, many AI advances, both practical and theoretical; can be used as a solution to such challenges. They mention that, the quest to build a chess-playing computer, for example, has led to many advances in search techniques, and similarly, the autonomous land vehicle project at CMU [Thorpe 1990] resulted not only in a highway-cruising vehicle but also in breakthroughs in vision, robotics, and neural networks. They furthermore mention that, they believe the adaptive Web sites challenge will both drive AI advances and yield valuable technology.

Following this discussion, same authors proposed a conceptual framework for adaptive Web sites in [Perkowitz and Etzioni, 2000]. The focus is mainly on the semi-automatic creation of index pages which are created through discovering clusters of page visits. The assumption in their solution is that, if a large number of users visit a set of pages frequently, these pages should be related. In order to find out those frequently accessed pages, they have developed two cluster mining algorithms, namely, PageGather and Index-Finder. PageGather relies on statistical methods to discover candidate link sets, whereas the IndexFinder is a conceptual cluster mining algorithm, which finds link sets that are conceptually coherent. The work was evaluated using three Web sites in which the automatically generated pages are placed. They observed the user response to these adapted pages. Although Perkowitz and Etzioni discussed that adaptive web problem can be considered as an AI problem, they did not provide a full formalization of the problem from an AI perspective.

There have not been many studies that propose solutions to adaptive web from the view of AI. In addition, most of the adaptive web based systems developed so far are described as compound systems that consist of many modules such as user profilers, log and web usage miners, content managers, Web site authoring tools and information acquisitioners and searchers [Eirinaki and Vazirgiannis, 2003]. However, in many of these approaches, clear description of an algorithm that performs the adaptation process is not provided. Therefore, one of the main drawbacks of the related work so far is that, in most of the approaches, there is no clear focus on adaptation formalism and algorithm on how the adaptation is performed. In this paper, IntWEB, a full AI based solution to the adaptive web problem is presented where Hill Climbing search method is used and the system is evaluated against a real-life case study. The main motivation of the study presented in this paper is to first formalize the problem as a classical AI search problem through giving all components of a classical search, and implementing a novel adaptation algorithm as well as evaluating the proposed approach with a real-life case-study.

2 IntWEB: Adapting Web Sites by using Hill Climbing

In IntWEB, adapting Web sites to users' preferences is handled as a search problem. The proposed solution is formulated as a classical AI problem and Hill Climbing search method is utilized to provide a solution for the adaptive Web. In this section, firstly, the problem formulation is described below. The components that construct the IntWeb framework are presented next. Finally, details of the algorithm are given.

2.1 Problem Formulation

Any classical AI problem can be formulated with four components, namely; *initial state, possible actions, goal test,* and *path cost*. In the following subsections, the formulation details of the adaptive web problem which forms the basis for IntWEB system is provided.

States

Each state in IntWEB is the entire Web site with the links that are taken to be under consideration for the Web site adaptation task. A state is therefore modelled as follows;

$\mathbf{S} = < \mathbf{L_1}$<from_ L_1, to_ L_1>, $\mathbf{L_2}$<from_ L_1, to_ L_2>, **...** , $\mathbf{L_k}$<from_ L_k, to_ L_k>, **Covers>**

Here, ***S*** is any configuration of the Web site that includes links *{$L_1, L_2, ... L_k$}*. Each link $\boldsymbol{L_i}$ corresponds to a link from page ***from_*** $\boldsymbol{L_1}$ to page ***to_*** $\boldsymbol{L_1}$.

Covers parameter keeps the *goodness* of a state. The goodness of a state is described in the following subsections.

Initial State

Initial state refers to the initial configuration of the Web site. The search starts with the Web site that includes no links. It incrementally adds links to the initial state through state transitions so as to maximize the ***Covers*** value of the state. The search continues until no more improvements can be made to the current state at hand.

Action

Initial state refers to the initial configuration of the Web site. The IntWEB is based on Hill Climbing search which makes state transformations through **add_shortcut** action. The **add_shortcut** action is designed and implemented as a transformation function which takes a state **S** as input and produces a new state **S'** by adding a link from a page to another page such that the newly added link does not exist in **S**.

Evaluation Function

Evaluation function measures the *goodness of a state* according to the problem at hand. Here, the goodness of a state is measured in terms of the *number of sessions that the state's link configuration* covers. The assumption used in the evaluation function is that, *the pages that accessed together in a session should be related and they should have a link to each other*. Therefore, the evaluation function assigns higher scores to states which have better link configuration. Here, as more paths that the user follows in sessions are covered by a state, the link configuration gets improved. Moreover, as the link configuration of a state improves, higher score is assigned to that state by the evaluation function.

Stopping Condition

Since Hill Climbing algorithm is used, IntWEB will stop when no more improvements can be made.

2.2 IntWEB: System Components

In order to realize the solution described above, IntWEB is established on three components, namely, *Pre-processor*, *Web Crawler*, and *Web Adapter modules*. The description of each module is provided in the following subsections.

Pre-processor

The Pre-processor module takes log files, which contain accesses to pages for different users and from different sessions as input, and produces sessions, which are in fact sets of pages that are accessed together, as output. The module does its job in three phases. In the first phase, it cleans the log data so as to remove irrelevant access information like robot accesses, multimedia accesses, etc. In the second phase, it does user identification on this cleaned log file. In order to identify users, it will use the IP address of the users and the browser information. This is a commonly applied technique in literature [Srivastava, Cooley, Deshpande and Tan, 2000]. After users are identified, the log file together with user information is re-processed in the third phase, to identify sessions. In session identification, two subsequent accesses are assigned to the same session if they come from the same visitor and if the time interval between them does not exceed a certain threshold value. Therefore, pre-processor module creates the session list as the output, which is used by the evaluation function.

Web Crawler
The Web Crawler module processes the Web pages of the Web site in order to form the states. Hence, Web crawler extracts the web site's topology automatically and the states are formed accordingly.

Web Adapter
The Web Adapter module runs the proposed algorithm for the whole state space. It aims to maximize the *Covers* value of each state using the evaluation function, which utilizes the session list produced by the Pre-processor module. The Web Adapter presents all discovered adaptations as a list, where this output list contains suggestions of inserting new links for the web pages under consideration to the user. More specifically, adaptation list contains lists of suggestions where each suggestion contains three values, which can be represented as *<link_from, link_ to, Covers>*. Here, a shortcut is suggested from *link_from* page to *link_to* page and such a shortcut exists in *Covers* number of sessions. Sample output of the Web Adapter module is given in Figure 5.

2.4 IntWEB: The Algorithm

For the solution, Hill Climbing Search is adapted to the Web adaptation problem. Hill Climbing is a technique in AI that can be utilized to solve problems with many possible solutions, where all these solutions make up the whole search space. The pseudo code of the algorithm is provided in Figure 1. In general terms, in Hill Climbing Search, the search space contains different solutions that have different evaluation values. Hill Climbing starts with an initial state, or an initial solution which is generally a poor solution, and it iteratively makes transitions to possible states through applying actions. At each step, it chooses the next best state according to an evaluation function. When the current solution can no longer be improved, it terminates.

The algorithm contains two basic parts. The generation of possible next states is done by ***GENERATE_NEIGHBORS()*** function and the evaluation is performed by ***EVAL()*** function, as shown in Figure 1.

```
currentNode = startNode;
loop do
  L = GENERATE_NEIGHBORS(currentNode);
  nextCovers = -INF;
  nextNode = NULL;
  for all x in L
    covers =  EVAL(x);
    if (covers > nextCovers)
            nextNode = x;
            nextCovers = covers;
    else if nextCovers <= EVAL(currentNode)
  //Return current node since no better neighbors exists
          return currentNode;
  currentNode = nextNode;
```

Figure 1. Hill Climbing Search

GENERATE_NEIGHBORS() uses **add_shortcut** action described in Section 2.1. This action produces all possible next states by considering all possible links from one page to another page. Once a link is chosen to be added by using the evaluation function, call it a link from page **A** to **B**, then the next possible states will be generated such that, only the possible links from B to all other possible Web pages will be considered. Once no more improvements can be made, we will end up with a list of links that will traverse the Web site starting from page or request **A** to the last request found at the end. Then, the search will restart again by the same empty initial state, however this time considering only possible links that starts with a possible addition of a link such that, the link will not have **A** as the source. Here *source* refers to any link L_i that appears in the ***from_L_i*** field of that link.

```
Covers = 0;
for each L_k<from_ L_k, to_ L_k> in current state S
 for each session s in Session_List
   if((s contains path from  from_ L_k toto_ L_k)  and
      (length_of_path(from_ L_k to  to_L_k)  in s notexceeds
            path_length_threshold))
       Covers=Covers+length_of_path(from_L_k,to_L_k);
 return Covers;
```

Figure 2. The pseudocode for *EVAL (S)* where state S is $S = <L_1<from\_ L_1, to\_ L_1>, L_2<from\_ L_1, to\_ L_2>, L_k<from\_ L_k, to\_ L_k>, Covers>$ using threshold *path_length_threshold*

As it is mentioned in Section 2.1, the evaluation function, ***EVAL(),*** measures the *goodness of a state* according to the problem at hand. The pseudo code of ***EVAL()*** is presented in Figure 2. Here, the main aim is to make it easier for the users to traverse the Web site, therefore, if a path such as, access to page **C** occurs after pages **A** then **B**, exists in many sessions, then it will be beneficial for the users to add a direct link from **A** to **C**. Here, there exist two considerations. First of all, the degree of goodness of adding a shortcut depends on the length of the path that is shortened. With path, we mean, the number of pages that you have to traverse in order to get to the desired page. For instance, in the above scenario, path from **A** to **C** needs getting to **B** from **A** then **C** from **B,** which is of length 2**.** Therefore, if system proposes to add a direct link between such two pages, and if the actual path between them is long, then this is more valuable compared to shortening an already short path via adding a link. Accordingly, the EVAL() function increases the covers value as much as the length of the path that is being shortened, as shown in Figure 2.

However, one more thing that should be considered is that, if such a path is too long, then it is highly probable that these two pages, in our case pages **A** and **C**, are not really related. For instance, if the user accesses **C** after accessing many pages from **A**, then these pages may not really be related and adding a direct link between them may not be meaningful. In order to control this, a parameter is kept in the system, namely ***path_length_threshold.*** With this parameter, EVAL() examines only the paths whose length does not exceed *path_length_threshold* parameter.

As a result, ***EVAL()*** processes each session for a state, and considering the paths that exist in the state, it calculates and returns the **Covers** value of that state considering the above discussions.

Assume that we have three web pages, p1(5), p2(4), p3(3). The numbers in parenthesis show the number of sessions that these pages are included. Firstly, the function chooses p1 as the source, and generates all possible states that have p1 as *from* field and each pi as *to* field. EVAL() function calculates the coverage value of the state. Next, all of the states that have p2 as the source and all of the possible states except p1 as the destination state are generated. This continues until all pages are covered as source.

3 Experimental Evaluation

The described IntWEB framework is evaluated by using the log data taken from a live Web site. In the evaluation phase, **path_length_threshold** parameter is set to different values so as to determine the effect of it on the results. In addition, the resulting adaptation list under an optimal path_length_threshold parameter is presented and discussed. Before presenting the evaluation phase, following subsection gives details about the dataset, and the results of the data pre-processing step.

3.1 Dataset

The designed system is evaluated by using the log files taken from the Web site of Middle East Technical University Computer Engineer Department [1]. The web server log files include an entry for each access to the server from a user. Each entry in the log file includes information regarding the IP address, authentication name, date-time of the access, HTTP request, response status, size of the requested resource, referrer URL and browser identification. The log data used in the experimental evaluation spans one week during the semester, more specifically, within the exams period. However, when logs are examined for different periods, it is observed that the patterns do not change much during the semester.

The log file is processed in order to extract the sessions, where each session is a set of page references from a user during one logical period. One important thing that needs to be mentioned here is that, IntWEB is not designed for any specific domain. In other words, in this paper, the results are presented for an academic department's Web site; however, the framework can work on any other log data from any other domain.

The log data initially contains 214010 accesses. After removing the noisy and irrelevant data (multimedia and robot accesses, erroneous accesses, accesses to newsgroup, etc), we left out with 13979 accesses. From these accesses, 211 different sessions are identified. In Figure 3, the number of requests per each session group is presented in order to give a general idea about the dataset. Here, sessions are grouped according to the number of requests they contain. In the figure, x-axis corresponds to the number of sessions, and y-axis corresponds to the number of requests per those sessions. As it can be seen from this figure, the number of session groups that have requests more than 10 are less than 4, whereas the number of sessions that have requests less than 10 comprises about 78% of the dataset.

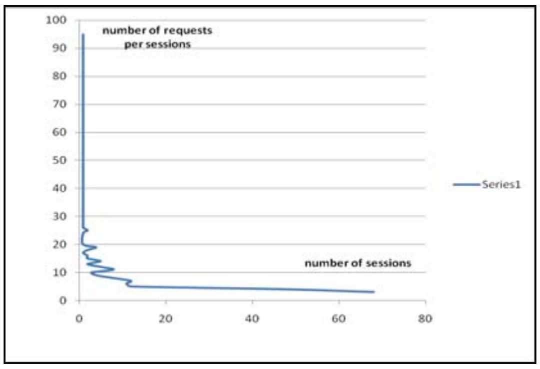


Figure 3. Number of Requests per Session

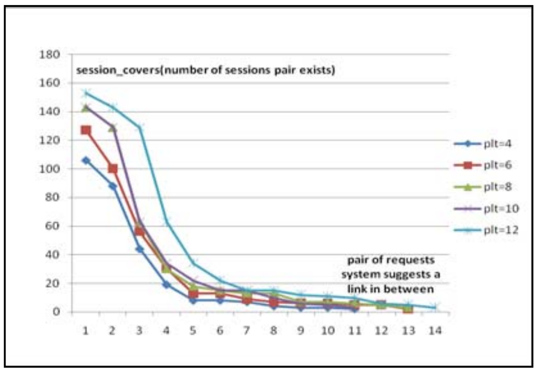


Figure 4. Covers value vs. path_length_threshold

[1] http://www.ceng.metu.edu.tr

From Page	To Page	Covers
/Course/?semester=2009&course=ceng242&cedit=0	/GradingTools/grades_histogram.php	136
/Course/?semester=2009&course=ceng140&cedit=0	/GradingTools/grades_histogram.php	127
/Student/homeworks.php	/GradingTools/grades_histogram.php	100
/Course/?semester=009&course=ceng334&cedit=0	/Student/homeworks.php?hid=1258	56
/GradingTools/grades_histogram.php	/Student/homeworks.php?hid=1165	30
/GradingTools/grades_histogram.php	/Course/?semester=2009&course=ceng280&cedit=0	13
/GradingTools/grades_histogram.php	/Student/homeworks.php?task_homeworks=list& selector_homeworks_course=ceng280	13
/Student/homeworks.php?hid=1258	/Student/homeworks.php?hid=1195	9
/Student/homeworks.php?hid=1165	/Student/homeworks.php?hid=1201	7
/Student/homeworks.php?hid=1201	/Student/homeworks.php	6
/Student/homeworks.php	Student/homeworks.php?task_homeworks=list& selector_homeworks_course=ceng382	6
/Course/?semester=2009&course=ceng280&cedit=0	/Course/?semester=2009&course=ceng232&cedit=0	5
/Student/homeworks.php?task_homeworks=list& selector_homeworks_course=ceng280	Student/homeworks.php?task_homeworks=list& selector_homeworks_course=ceng382	5
/Student/homeworks.php?hid=1195	Student/homework's.php?hid=1162	2

Figure 5. Sample Adaptation Results under path_length_threshold 6

2.2 Results

In order to evaluate the system's resulting adaptation list for different path_length_threshold parameter settings, the parameter is set to 4,6,8,10 and 12. Results can be seen in Figure 4. In this figure, x-axis corresponds to each pair of requests where the system suggests creating a link between, and y-axis represents the number of sessions that this pair exists, that is, the **session_covers** value. From the figure, we can say that, as the path_length_threshold parameter increases, the **session_covers** value for different pairs increases, since the possibility of pairs' existence in a session increase. Similarly, the length of the adaptation list also increases when path_length_threshold parameter increases, which is as expected, since, the number of possible link pairs increases as we permit longer sequences of requests.

In Figure 5, the resulting adaptation list under path_length_threshold 6 is presented. The resulting adaptation list proposes 14 suggestions, which can be summarized as follows:

1. Adding shortcuts from course web pages to grade web pages
2. Adding shortcuts from course web pages to the homeworks
3. Adding shortcuts from homework list to grades and grades to homework
4. Adding shortcuts between different homeworks (same semester, same class)
5.

Here, when the suggestions are examined, all of them appear to be logical and useful when the domain at hand is considered. For instance, from the resulting adaptation list we can conclude that, users enter web pages of homework one after another, therefore; shortcuts between homeworks of a specific years students (like 4th year students), and for a specific semester can facilitate users' navigation.

IntWEB presents the generated list, as shown in Figure 5, to Web site administrator as adaptation suggestions. S/he can then take into account the propositions that s/he thinks best suits the Web site, and then the accepted adaptation suggestions are applied. Afterwards, it is possible to check whether users get satisfied with the changes. This process can continue within the lifecycle of adaptation of the Web site. In order to test the user's satisfaction, Web site admin-istrator can again use the system. For instance, after s/he applies the proposed adaptations, s/he will again collect log files for a specific time period and check the resulting usage patterns in order to find out whether the added links are followed and facilitated user's navigation on the Web site.

From the results presented and discussed above, we can conclude that the system produces feasible adaptation suggestions. One important point to mention here is that, the Web site used in the evaluation does not contain many number of Web pages. Therefore, the log file does not include many different user access patterns. We believe that, with larger Web sites, such as e-commerce sites including information on various types of products, the log file would be richer and the resulting adaptation list could contain more number of suggestions.

4 Conclusion and Future Work

In this paper, adaptive Web problem is formulated as a Hill climbing algorithm through defining all the necessary components of a search problem and presenting a novel algorithm for the adaptation of Web pages from server access log files. The experimental results show that the proposed approach discovers useful adaptation suggestions.

The basic idea is to promote shortcuts that are proposed to shorten long paths, however, these paths should also be related. If the path is too long, then probably the user did not aim to traverse the whole path. This is controlled by cutoff threshold. In addition to adaptation through adding shortcuts, deleting unnecessary shortcuts is also supported. The search starts as if there is no connected Website, but only the Web pages. Then the generated results propose adding shortcuts. Assume that proposed shortcut set is P, and the set of current shortcuts existing in website is W. Then the Web administrator can consider to remove the shortcuts in the set W-P or to include shortcuts in P.

Concerning the notion of adaptive Web systems, in addition to statistical evaluation, human evaluation of the results is also needed. Therefore, as the future work, it is planned to present the adaptation list to the Web site administrator and let him/her evaluate the results. In addition, some sort of questionnaire-based evaluations will be very useful to gather the user's ratings and comments towards the produced adaptations. In addition, it is ex-pected that IntWEB will perform better if a richer Web site with various types of accesses is used in the evaluation. Therefore, as a future work, the system is planned to be tested and evaluated in different environments.

References

[Perkowitz and Etzioni, 1997] M. Perkowitz and O. Etzioni. Adaptive Web sites: an AI challenge. In Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, 1997.

[Perkowitz and Etzioni, 1998] Mike Perkowitz and Oren Etzioni. Adaptive Web Sites: Automatically Synthesizing Web Pages. In Proceedings of the Fifteenth National Conference on Artificial Intelligence, 1998.

[Russell and Norvig, 2003] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (2nd ed.)*. Prentice Hall, pages 111–114, ISBN 0-13-790395-2, Upper Saddle River, New Jersey, 2003.

[Perkowitz and Etzioni, 2000] Mike Perkowitz, and Oren Etzioni. Towards adaptive Web sites: Conceptual framework and case study. Artificial Intelligence 118(1-2): 245-275, 2000.

[Joachims, Freitag, and Mitchell, 1997] Thorsten Joachims, Dayne Freitag and Tom. Mitchell. WebWatcher: A Tour Guide for the World Wide Web. In *Proc. of International Joint Conference on Artificial Intelligence*, pages 770-775, Nagoya, Japan, 1997.

[Wexelblat, and Maes, 1999] Alan Daniel Wexelblat, and Pattie MaesFootprints. History-Rich Tools for Information Foraging. In *Proc. of Proceedings of Human Factors in Computing Systems (CHI)*, pages 270-277, Pittsburgh, Pennsylvania, United States, 1999.

[Fink, Kobsa, and Nill, 1996] Josef Fink, Alfred Kobsa and Andreas Nill. A. User-Oriented Adaptivity and Adaptability in the AVANTI project. In *Proc. of Conference "Designing for the Web: Empirical Studies"*, Microsoft, Redmond, WA, 1996.

[Thorpe, 1990] Charles E. Thorpe. *Vision and Navigation: The Carnegie Mellon Navlab*. Kluwer Academic, Boston, MA,1990.

[Eirinaki and Vazirgiannis, 2003] Magdalini Eirinaki and Michalis Vazirgiannis. Web mining for web personalization, *ACM Transactions on Internet Technology (TOIT),* v.3 n.1, pages 1-27, February 2003.

[Srivastava, Cooley, Deshpande, and Tan, 2000] Jaideep Srivastava , Robert Cooley , Mukund Deshpande , and Pang-Ning Tan. Web usage mining: discovery and applications of usage patterns from Web data, *ACM SIGKDD Explorations Newsletter*, v.1 n.2, January 2000.

[Cotter and Smyth, 2000] Paul Cotter and Barry Smyth. WAP-ing the Web: Content personalization for WAP-enabled devices. In: Brusilovsky, P., Stock, O., Strapparava, C. (eds.) Proc. of Adaptive Hypermedia and Adaptive Web-based systens. Lecture Notes in Computer Science, Vol. 1892. 98-108. Springer-Verlag, 2000.

[Mulvenna, Anand and Buchner, 2000] Maurice D. Mulvenna, Sarabjot S. Anand, Alex G. Büchner. Personalization on the Net using Web Mining. Communications of the ACM, Vol. 43, No. 8:23-125, 2000.

[Mobasher 2007] Bamshad Mobasher. Data Mining for Web Personalization. In The Adaptive Web: Methods and Strategies of Web Personalization, Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.). Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.). Lecture Notes in Computer Science, Vol. 4321, PP. 90-135, Springer, Berlin-Heidelberg, 2007.

[Goldfarb 2002] Avi Goldfarb. Analyzing Website Choice Using Clickstream Data, 209-230, Elsevier Science Ltd., 2002.

[Sismeiro and Bucklin, 2004] Catarina Sismeir and Randolph E. Bucklin. Modeling Purchase Behavior at an E-Commerce Web Site: A Task-Completion Approach. Journal of Marketing Research. XLI.306-323, 2004.

[Mobasher, Cooley, and Srivastava, 2000] Bamshad Mobasher,Robert Cooley, and Jaideep Srivastava. Automatic Personalization based on web usage Mining. Communications of the ACM, Vol. 43, No.8, pp. 142-151, 2000.

[Mobasher, Dai, Kuo, and Nakagawa 2001] Bamshad Mobasher, Honghua Dai, Tao Kuo, and Miki Nakagawa. Effective Personalization Based on Association Rule Discovery from Web Usage Data. In Proceedings of the 3rd ACM Workshop on Web Information and Data Management (WIDM01), in conjunction with the International Conference on Information and Knowledge Management (CIKM 2001). Atlanta, Georgia, 2001.

[Nasraoui and Petenes 2003] Olfa Nasraoui and Christopher Petenes. Combining Web Usage Mining and Fuzzy Inference for Website Personalization. In Proc. of WebKDD 2003, KDD Workshop on Web mining as a Premise to Effective and Intelligent Web Applications. Washington DC. p. 37, 2003.

Placing High-Diversity Items in Top-N Recommendation Lists

Mouzhi Ge
TU Dortmund, Germany
mouzhi.ge@tu-dortmund.de

Fatih Gedikli
TU Dortmund, Germany
fatih.gedikli@tu-dortmund.de

Dietmar Jannach
TU Dortmund, Germany
dietmar.jannach@tu-dortmund.de

Abstract

Most works in the domain of recommender systems focus on providing a list of accurate recommendations. From a user perspective, users may however feel frustrated when they are facing a monotonous recommendation list. To tackle this problem, a few recent works have proposed different algorithms to generate the recommendation list with the feature of diversity. However, little research has been done to determine the placement of the high-diversity items in a recommendation list. Therefore this paper attempts to provide a guideline to appropriately place the high-diversity items in the recommendation list. Our pilot study shows that it was easier to discover high-diversity items when arranged in a block than in the case when the high-diversity items were dispersedly positioned. In addition, providing explanations may help to improve acceptance of the high-diversity items.

1 Introduction

Over the last decade, the majority of the proposed improvements to recommender system algorithms have focused on computing more accurate predictions. Accuracy in this context means the closeness between the system's predicted rating and the user's real rating for an item [Herlocker et al. 2004]. Most recommender systems are designed to provide a list of items for which the system predicts high ratings. Accordingly, improving the accuracy means that the system can predict more reliably, which items are most probably liked by the user, which in turn is assumed to increase the overall user satisfaction.

A number of recent studies have found that beyond accuracy there are other quality factors such as diversity and novelty, which are also important to users. Only concentrating on accuracy may even negatively impact the systems [McNee et al. 2006]. Therefore, different techniques were recently proposed which compute recommendation lists that take into account alternative quality factors. Previous studies have for example argued that users may feel frustrated when there is little variance in the recommendation list [Zhang and Hurley 2008, Lathia et al. 2010]. *Diversity* therefore is an important factor that recommender systems need to take into account. Furthermore, Smyth and McClave [2001] point out that diversity is as important as accuracy and it is considered as an orthogonal measure to accuracy. In general, we can observe a trade-off between diversity and accuracy, which means that increasing the diversity of a recommendation list most probably results in a decrease of its accuracy and vice versa [Adomavicius and Kwon 2011]. Regarding this trade-off, some state-of-the-art works focus on controlling the balance between accuracy and diversity [Smyth and McClave 2001, Ziegler et al. 2005] or increasing the diversity of recommendations with a minimal loss of accuracy [Zhang and Hurley 2008, Adomavicius and Kwon 2011]. In most of the proposed algorithms, diversity is measured in terms of dissimilarity of the recommended items (intra-list similarity). Intra-list-similarity is determined by measuring the similarity between all pairs of items in the recommendation list [Ziegler et al. 2005]. This measurement can be based, for example, on the known features of the items.

Note that intra-list similarity does not depend on the ordering of the items, which means that rearranging the positions of the recommended items in the list will not affect the diversity metric [Ziegler et al. 2005]. In contrast to these previous works we however conjecture that the placement of the high-diversity items in a recommendation list may affect the perceived diversity and its utility as well as the overall quality impression by the user.

Consider a scenario in which a recommender has generated a list of ten recommendations for a user. Let us assume that three of the items were introduced by the system to increase the list's diversity. If we place these three high-diversity items in the top 3 positions in this recommendation list, the user will most probably see these items first and only then the more "accurate" items. This arrangement may confuse or disappoint the user since he could have the impression that the recommender does not understand his requirements or that the recommender's predicting ability is poor. Moreover, users may stop using the recommender system right after they viewed the first few items. We find it therefore important and valuable to study how to arrange the order of recommended items. Castells et al. [2011] point out that investigating the order of recommended items is mostly missing in recommender system research. We therefore intend to study whether and to which extent different orderings affect user satisfaction and the perceived diversity and the system's overall quality.

The final objective of this work is to propose a guideline of how to organize items in a top-N recommendation list. Accordingly, we will investigate how different placements of high-diversity items affect user satisfaction and the perceived diversity. Our results are expected to indicate where the high-diversity items should be placed in a recommendation list. As such, our contributions can also be used to improve the user interface design and the general user experience in recommender systems.

This research-in-progress paper is organized as follows. In Section 2 we shortly review papers related to diversity in recommender systems. Next, in Section 3, we propose an experimental design to study the effects of different placements of high-diversity items on user satisfaction and perceived diversity. Subsequently, we describe a pilot study and summarize our initial findings. We conclude this paper by discussing our experimental design and the identified indications of how to place the high-diversity items in a recommendation list.

2 Related Works

We propose to divide the concept of diversity into *inherent* (objective) and *perceived* (subjective) diversity. Inherent diversity refers to the diversity calculated based on the dissimilarity among the recommendations and can be further classified as *individual diversity* and *aggregate diversity*. While individual diversity, also named as intra-list diversity [Castells et al. 2011], is related to the diversity of a recommendation list for an individual user, aggregate diversity, which is also termed inter-user diversity [Zhou et al. 2010], is to address the overall diversity across all users. Considering the trade-off between accuracy and diversity, some researchers [Smyth and McClave 2001, Ziegler et al. 2005, Zhang and Hurley 2008] propose algorithms to increase the individual diversity by compromising accuracy. The goal of these works is to optimize the balance between accuracy and diversity so as to keep accuracy in a certain level when increasing diversity. To increase diversity and at the same time minimize the effect on accuracy, other researchers [Zhou et al. 2010, Adomavicius and Kwon 2011] focus on increasing the aggregate diversity to solve the dilemma between accuracy and diversity. Furthermore, Fleder and Hosanagar [2007] have shown that individual diversity and aggregate diversity are not necessarily related.

Perceived diversity refers to the diversity experienced by the user and can be divided into *current perceived diversity* and *temporal perceived diversity* [Lathia et al. 2010]. Current perceived diversity means the diversity perceived by one user at a single time. In contrast, temporal perceived diversity is the diversity perceived by the user over a period of time. It can be measured, for example, by comparing the differences between two recommendation lists provided to the same user at different times [Lathia et al. 2010]. The advantage of perceived diversity is that it can capture user's opinion towards diversity. However, since different users may perceive diversity differently, we admit that it is challenging to unify the different perceived diversities. We summarize the main focus of research of previous works related to *inherent* and *perceived* diversity in Table 1.

	Inherent diversity		Perceived diversity	
	individual diversity	aggregate diversity	current perceived diversity	temporal perceived diversity
Smyth and McClave 2001	×			
Ziegler et al. 2005	×			
Fleder and Hosanagar 2007	×	×		
Zhang and Hurley 2008	×			
Zhou et al. 2010		×		
Lathia et al. 2010				×
Castells et al. 2011	×			
Adomavicius and Kwon 2011		×		
This paper			×	

Table 1 Summary of our literature review

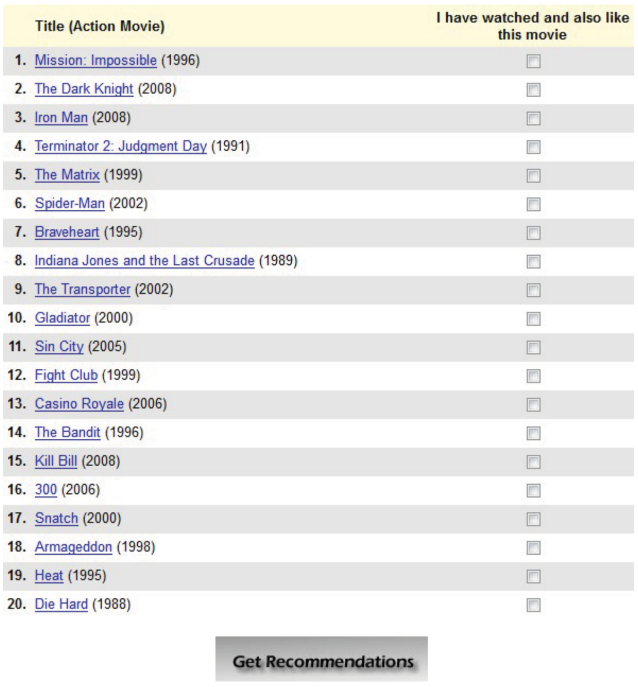
The selected papers are arranged in a chronological order in the table. We can observe that more recent works begin to focus on studying subjective diversity. Furthermore, we found that little work has been done to investigate the *current perceived diversity*. To bridge this gap, one of our research objectives is to study the effect of the placement of high-diversity items on the *current perceived diversity*.

3 Experimental Design

In this section, we will shortly present and review the experimental setup and measurement technique used in this study. Our general goal is to find out which placement order of high-diversity elements best suits the users' needs and is well accepted by the users. We also want to find out whether and to which extent diverse elements in a recommendation list can influence the user-perceived quality of a recommender system. Therefore we decided to conduct a user study because it is hard to simulate a user's perceptions of, for example, diversity or novelty in offline experiments. For this reason, we employ a *within subjects* user study, in which each subject is confronted with *all* variations of recommendation list tested in this work.

Our experiment consists of three different screens which were displayed for several movie genres. The first screen simulates a training or learning phase in the recommendation process. In this screen the user is provided with a list of 20 movies of one specific genre. Figure 1 shows an example list for the genre *action*. The users were asked to check the movies they have watched *and* also liked. This is to give the user the impression that there is a recommender system running in the background which is trying to learn the user's movie preferences. To support the illusion of intelligent behavior and background calculations, we showed a "Calculating" message for two seconds with the hint that the recommendations are computed after the user clicked on the "Get Recommendations" button. On the second screen, a recommendation list with 12 movies was then presented to the user. This procedure (Figure 1 and 2) is carried out for action movies, romantic movies, comedy movies and animation movies.

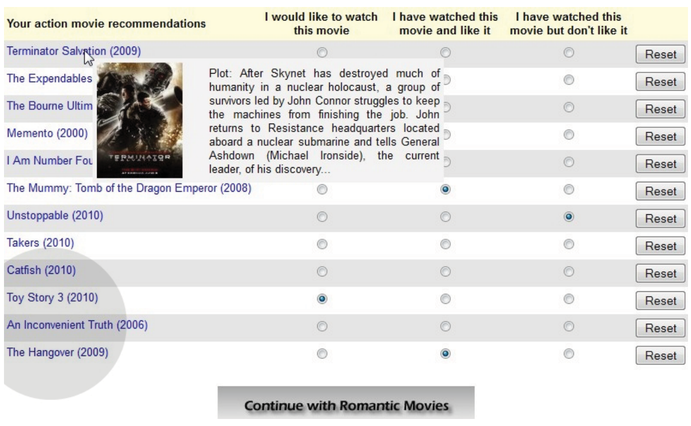
It is important to know that in the whole experiment we do not make use of a recommender system for computing the recommendations. Instead, we manually create a static list of movies for each genre and provide it to each user. Therefore the experiment looks exactly the same for each participant and each participant is confronted with exactly the same recommendations in the same order.



	Title (Action Movie)	I have watched and also like this movie
1.	Mission: Impossible (1996)	☐
2.	The Dark Knight (2008)	☐
3.	Iron Man (2008)	☐
4.	Terminator 2: Judgment Day (1991)	☐
5.	The Matrix (1999)	☐
6.	Spider-Man (2002)	☐
7.	Braveheart (1995)	☐
8.	Indiana Jones and the Last Crusade (1989)	☐
9.	The Transporter (2002)	☐
10.	Gladiator (2000)	☐
11.	Sin City (2005)	☐
12.	Fight Club (1999)	☐
13.	Casino Royale (2006)	☐
14.	The Bandit (1996)	☐
15.	Kill Bill (2008)	☐
16.	300 (2006)	☐
17.	Snatch (2000)	☐
18.	Armageddon (1998)	☐
19.	Heat (1995)	☐
20.	Die Hard (1988)	☐

Figure 1 Screen 1 - Acquiring user preferences for action movies.

In order to design different placements of high-diversity items, in the first list of action movies we show the diverse elements in one block at the end of the list, see Figure 2. In the second list of romantic movies, the diverse elements were presented in the middle of the list, again as a block. In the third list of comedy movies, the four diverse items are respectively placed at position 3, 6, 9 and 12 in the recommendation list. Finally, we use a list of animation movies as our control group containing no diverse recommendations. For the recommendation lists of action, romantic and comedy movies, we insert four high-diversity movie recommendations that do not fit into the genre-category of the corresponding list. For example, as shown in Figure 2, we add an animation movie (Toy Story 3) for kids to the recommendation list of action movies and consider the animation movie in the list of action movies as a diverse movie recommendation. Thus in our experiment the determinant of diversity is the genre difference between movies.



Your action movie recommendations	I would like to watch this movie	I have watched this movie and like it	I have watched this movie but don't like it	
Terminator Salvation (2009)	○	○	○	Reset
The Expendables			○	Reset
The Bourne Ultim			○	Reset
Memento (2000)			○	Reset
I Am Number Fou			○	Reset
The Mummy: Tomb of the Dragon Emperor (2008)	○	●	○	Reset
Unstoppable (2010)	○	○	●	Reset
Takers (2010)	○	○	○	Reset
Catfish (2010)	○	○	○	Reset
Toy Story 3 (2010)	●	○	○	Reset
An Inconvenient Truth (2006)	○	○	○	Reset
The Hangover (2009)	○	●	○	Reset

Plot: After Skynet has destroyed much of humanity in a nuclear holocaust, a group of survivors led by John Connor struggles to keep the machines from finishing the job. John returns to Resistance headquarters located aboard a nuclear submarine and tells General Ashdown (Michael Ironside), the current leader, of his discovery...

Figure 2 Screen 2 - Displaying action movie recommendations to users.

For example, Figure 2 shows the recommendation list for the category of action movies. As described above, the four movies at the bottom of the list (for example the animation movie "Toy Story 3") represent high-diversity items in the recommendation list of action movies. For each movie recommendation, the users can indicate whether they want to watch this movie; skip this movie, in case they do not like the recommendation; or indicate whether or not they like it in the case that they had already watched this movie. In the end, the user is asked to evaluate the provided recommendation list for each movie genre. On a rating scale of 1 to 5 with one point increments users could answer the following questions for each recommendation list:

- Are you satisfied with the movie recommendations? (1: not satisfied, 5: satisfied)
- Is the amount of the recommendations enough? (1: too few, 5: too many)
- Does this recommendation list surprise you? (1: not at all, 5: very surprised)
- Do you think this recommendation list is diversified? (1: not at all, 5: very diversified)

We also provided a textbox where the user could leave feedback regarding our recommendations. It took each participant between 5 and 10 minutes to complete the survey in our initial pilot study.

4 Pilot Study

As a pilot study, we invited 10 subjects to participate in the experiment. They were either working staff or students at the Technical University of Dortmund. The average age of the subjects was 29 years. 30% of the subjects were female and 70% male. All of them had little or no experience with research in recommender systems. For each subject, we supervised the whole experimental procedure. Based on this initial pilot study, we can summarize our first observations as follows:

(1) Most subjects were able to recognize that there are high-diversity items in the recommendation list. Three subjects in their feedbacks emphasized the existence of high-diversity item, for example, "*An Inconvenient Truth* is definitely not an action movie but good to know". In most cases when subjects identified high-diversity items, they further inspected the related information provided by the system (e.g. the movie plot). An interesting finding here is that some subjects were particularly interested in the high-diversity item. This indicates that although the high-diversity items may decrease the accuracy of the recommendation list, they can attract the users' attention and arouse the users' interest.
(2) For most subjects, it was easier to discover high-diversity items when arranged in a block than in the case when the high-diversity items were dispersedly positioned. This can be observed from the response of the participants. When subjects were facing a recommendation list with a block of high-diversity items, they sometimes directly told us that these items are not fitting into the recommendation list. However, this did not take place when the high-diversity items were dispersedly positioned. Therefore, we can propose a hypothesis that under the same individual diversity, a recommendation list containing a block of high-diversity items is perceived more diverse than one with dispersedly positioned high-diversity items.
(3) An explanation facility may help to increase the user acceptance of high-diversity items. In our pilot study, most subjects were particularly interested in reading the plot information and external links for the high-diversity movies. This indicates that when facing a high-diversity item, users hope to find out why the system recommends this item. Therefore we infer that if a system provides diverse recommendations but without corresponding explanations, this may decrease user satisfaction. We thus propose a hypothesis to be tested in our ongoing work that an explanation facility can increase the acceptance of high-diversity items.

5 Discussion and concluding remarks

In this research-in-progress work, we reported the preliminary observations from the pilot study of our experiment. As mentioned in the section of experimental design, we present four recommendation lists to subjects and observe their responses via changing the positions of the high-diversity items. In order to assure the effect is only from the varied positions, we need to keep the variance effect of accuracy, diversity and novelty between the four recommendation lists at a minimum. Note that our aim is to provide four recommendation lists at the same level, rather than providing high-quality recommendations. Thus, randomly choosing the popular movies for the four recommendation lists allows us to keep their accuracy at the same level. Furthermore since the total number of recommendations and the number of high-diversity recommendations are the same in the four recommendation lists, we can keep the four lists at the same diversity level. Also, considering the dependency between novelty and diversity, we control the mean and standard deviation of movie release years in the four recommendation lists at the same level. In our forthcoming work we will report more details of our experiment and data analysis results.

References

[Adomavicius and Kwon 2011] Gediminas Adomavicius, YoungOk Kwon. Improving Aggregate Recommendation Diversity Using Ranking-Based Techniques. IEEE Transactions on Knowledge and Data Engineering (forthcoming).

[Castells et al. 2011] Pablo Castells1, Saúl Vargas, Jun Wang. Novelty and Diversity Metrics for Recommender Systems: Choice, Discovery and Relevance. In *Proceedings of International Workshop on Diversity in Document Retrieval (DDR)*. Dublin, Ireland: 29-37.

[Fleder and Hosanagar 2007] Daniel Fleder, Kartik Hosanagar. Recommender Systems and their Impact on Sales Diversity. In *Proceedings of the 8th ACM Conference on Electronic Commerce*. San Diego, CA, USA: 192-199.

[Herlocker et al. 2004] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, John Riedl. Evaluating Collaborative Filtering Recommender Systems. *ACM Transactions on Information Systems* 22(1): 5-53.

[Lathia et al. 2010] Neal Lathiax, Stephen Hailesx, Licia Caprax, Xavier Amatriain. Temporal Diversity in Recommender Systems. *Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. Geneva, Switzerland: 210-217.

[McNee et al. 2006] Sean M. McNee, John Riedl, Joseph A. Konstan. Being Accurate is Not Enough: How Accuracy Metrics have hurt Recommender Systems. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*. Montréal, Canada: 1097-1101.

[Smyth and McClave 2001] Barry Smyth, Paul McClave. Similarity vs. Diversity. In *Proceedings of 4th International Conference. on Case-Based Reasoning*. UK: 348-361.

[Zhou et al. 2010] Tao Zhoua, Zoltán Kuscsika, Jian-Guo Liua, Matúš Medoa, Joseph Rushton Wakelinga, Yi-Cheng Zhang. Solving the apparent diversity-accuracy dilemma of recommender systems. *National Academy of Sciences of the USA.* 107(10): 4511-4515.

[Zhang and Hurley 2008] Mi Zhang, Neil Hurley. Avoiding Monotony: Improving the Diversity of Recommendation Lists. In *Proceedings of the 2nd ACM Recommender Systems*, Lausanne, Switzerland: 123-130.

[Ziegler et al. 2005] Cai-Nicolas Ziegler, Sean McNee, Joseph Konstan, Georg Lausen. Improving Recommendation Lists through Topic Diversification. In *Proceedings of the 14th WWW conference*. Chiba, Japan. 22-32.