Imperial College London

MEng Individual Project
Interim Report

Department of Computing

Imperial College of Science, Technology and Medicine

The Next Generation Personal Research Assistant

Author:
Julian Wong

Supervisor:
Dr. Sergio Maffeis

January 26, 2023

Submitted in partial fulfillment of the requirements for the MEng Honours Degree in Computing (Artificial Intelligence and Machine Learning) Part IV of Imperial College London

Contents

Chapter 1

Introduction

Zotero is a free-to-use, open-source research paper management platform that aims to help its users collect, organise, annotate, cite and share research [1]. The Zotero platform mainly consists of a client-side application that displays a library of collections of research papers and resources that can be synced and modified online via a web library. Figure 1.1 shows an example of what the main interface of the Zotero client looks like.

Aside from the essential functionality that Zotero offers out of the box, including tagging, annotations, bibliography generation, and web browser extensions for saving online resources, there is also an active community of Zotero users who develop extensions for the Zotero client [2] using Zotero's internal JavaScript API, using technologies similar to those found on legacy Firefox extensions [3]. The main areas of development of these extensions can be broadly generalised into categories such as metadata aggregation for research papers (e.g. Zotero Citation Counts Manager [4]), attachment file management for PDFs (e.g. ZotFile [5]), advanced graphical annotations (e.g. Better Notes [6]), and integration with other platforms such as Microsoft Word and LaTeX editors.

From this collection of extensions, it is apparent that whilst there is an abundance of support for the modification and analysis of existing resources within a user's library, there is lesser development surrounding the manipulation of external resources that are outside one's library, such as those online. This is a limitation for Zotero users, who may not be able to conveniently find related resources that are worth exploring

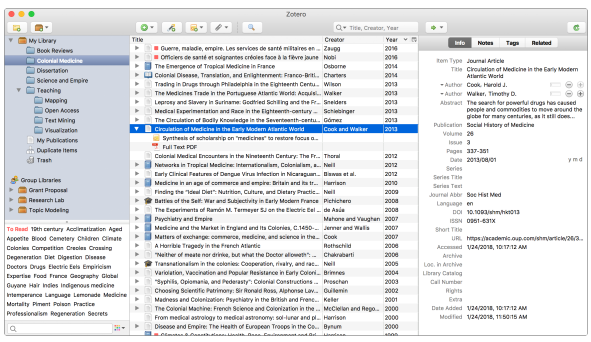

Figure 1.1: A screenshot of the Zotero client user interface.

further based on their current research interests.

Therefore, the aim of this project is to design and implement a research paper recommendation system as a plugin for the Zotero client. The project will also evaluate the resultant recommender system and provide metrics related to its performance. This recommender system should leverage upon the benefits of having existing tools provided by the Zotero platform, and thus be an improvement compared to existing solutions of a similar nature. For example, the system should be able to take in a single research paper as input, or it could take in a collection of papers, such as the collections that would have been created and organised by a Zotero user. Furthermore, the system could take in keywords highlighted by the user within a paper, on top of the paper itself, so that the system has information about what subject areas to focus on when generating recommendations.

Chapter 2

Background

This chapter compares and contrasts different types of generic recommendation methods, and explores the different architectures of existing research paper recommender system proposals, providing a comprehensive list of the advantages and disadvantages of using each approach.

2.1 Recommender systems

Recommender systems for academic and research papers have been part of a growing area of research in recent years, and continue to be a future trending research interest among researchers, publishers and students in higher education. In particular, throughout the last decade, various methods of implementing such a system have been suggested and illustrated, most of which utilise the power of generic recommender systems, natural language processing, and machine learning.

Nascimento et al. [7] provide a comprehensive overview of the problem from a more generic, platform-independent perspective, and highlights the advantages and disadvantages of more novel approaches to implementing research paper recommender systems in the past. For example, whilst directly utilising a paper's references for recommenders is an efficient and simple method, citations are ultimately a human-generated piece of 'metadata' from the paper, and is thus something that we shouldn't rely too heavily on. Furthermore, citations alone have no information about their specific relationship to the main concepts of a paper, some important references may not always be cited by the author(s), and citations can be considered proprietary information in some circumstances.

2.1.1 Collaborative and content-based approaches

Before diving into research-specific recommender systems, it is useful to first compare and contrast the major driving algorithms behind modern recommender systems – collaborative filtering and content-based methods. This leads us onto the discussion of the feasibility and suitability of using each type for our use case.

Collaborative filtering

It is often the case that recommender systems from other popular domains, especially multimedia content such as movies and music, rely on collaborative filtering, which is a user-to-user method. The objective of collaborative filtering is summarised mathematically from a high level by Adomavicius and Tuzhilin [8]. The main problem that a recommender system aims to solve is to generate an ordered list of items, sorted by their usefulness to the target user, or 'utility'. To achieve this, the system must first solve the problem of calculating or predicting the utility of some specified item. Formally, the problem space consists of a set of users C, a set of items S, and a utility function $u(c, s)$ that returns the utility of item s based on target user c, where $u : C \times S \to R$.

A collaborative approach first compares the target end-user with other end-users in C and selects the k most 'similar' end-users. The similarity of two users c_1 and c_2 is calculated based on the ratings both end-users have given in the past. Their ratings of items that both of them have rated in the past can be represented as two vectors x_1 and x_2, and their similarity score can be found using flexible methods, such as finding the cosine of the angle between these two vectors in memory-based approaches. Once a set of similar end-users C' has been collected, the utility $u(c'_i, s)$ of every similar end-user c'_i can then be calculated and used to predict the utility $u(c, s)$ of the target user c and target item s. For example, in a memory-based approach, one could predict the rating user c would give to item s by calculating the average rating other similar users have given to the same item, or:

$$r_{c,s} = \frac{1}{N} \sum_{c' \in C'} r_{c',s}$$

A problem that arises from this method is the 'cold-start' [9], where the few number of users at the beginning of the lifecycle of a system means that the quality of recommendations is often poor due to a lack of overall information about its users. This highlights a more fundamental concern regarding a collaborative approach, which is that it heavily depends on all other existing users on the system, and requires the system to have knowledge of each user's ratings and preferences. Relating this to Zotero integration, a collaborative approach would not be ideal as the Zotero platform is not a multi-user platform by design, unlike other online platforms where collaborative filtering would be the obvious approach, such as social media. For example, the Zotero web API requires authentication information such as a client key and client secret in order to access a given client's library and collections [10]. This data would rightly be considered proprietary and private by users, and should therefore not be incorporated into our recommender system. Furthermore, this would only be possible if the Zotero user had created an online account on the platform and registered their copy of the client application to their account, which may not always be the case, rendering this an infeasible approach.

Content-based methods

In contrast, content-based methods depend on the target user c and the set of all items S for prediction. The target user c would have a set of items S_c that they are interested in, or have rated highly of previously. Adomavicius and Tuzhilin [8] formalise that a vector ContentBasedProfile(c) can be generated based on the information from S_c, and is meant to represent the preferences of a user based on their interested content. The method then iteratively generates a vector Content(s) representing the information of each item s in S, and compares the similarity between Content(s) and ContentBasedProfile(c). Therefore, the utility function $u(c, s)$ can be defined as:

$$u(c, s) = \text{score}(\text{ContentBasedProfile}(c), \text{Content}(s))$$

In practice, the vectors Content(s) and ContentBasedProfile(c) can be generated using natural language processing techniques, such as aggregating the term frequency-inverse document frequency (TF-IDF) measure [11] for every keyword in a document, or for every document in a user's profile.

As mentioned above, this approach would be more preferable for our use case, since the method solely relies on the target user, their personal research paper library and collections, and other publicly available research publications, potentially online. In simpler cases where the system is recommending related resources for a single paper, the single target paper s_c is the only source of information we need from the user.

However, a problem that may arise from an NLP-oriented approach such as TD-IDF is that a paper is solely abstractly represented by its important keywords in the form of a vector. Adomavicius and Tuzhilin [8], and Shardanand and Maes [12] both mention that this could be a problem when two papers, for example, a well-written one and a poorly-written one, are represented by the same keywords, and there is no other metric the system can use to determine which paper would be a better recommendation for the user.

2.1.2 Existing recommender system proposals

Having established preferable approaches to implementing recommender systems in a generic context, this section focuses on analysing some of the existing proposals for implementing research paper-specific recommender systems, comparing and contrasting the concrete techniques and technologies used in each approach. This knowledge can then be used to assess what improvements could be made to existing solutions to this problem, and thus what concrete approaches our Zotero-oriented recommender system could take.

Scalable source-independent architecture

Nascimento et al. [7] propose a 'source-independent' solution that aims to mitigate the problem of legacy recommender systems overly relying on the use of user profiles and proprietary information, including a user's private library collection and citations. Instead, the objective of the proposed framework is to only rely on information that can be gathered from the single target paper in question and other publicly available

sources of data online. All of this eventually accumulates to an end-user experience that requires low effort, which is a property that would greatly suit the interests of implementing such a recommender system on the Zotero client. The flow of a query within the framework can be summarised in the following steps:

1. the target paper is parsed into its sections (e.g. title, abstract etc.) and keywords are extracted from each section,

2. sections of texts and keywords are arranged into a pool of queries which are sent to public databases, with the results then aggregated into a single list,

3. this list of recommended papers is removed of duplicates and ranked using content-based methods,

4. and finally, this list is returned to the end-user as output.

One key advantage that this framework brings is that it is scalable at every intermediate stage of the process. For example, this design enables the end-user to specify which public databases they would like their recommendations to come from, depending on the databases' popularity, authoritativeness, relevance, and so on. It also allows the end-user to specify the method in which queries to the databases are generated depending on their use case. For example, the structure and the keywords used in a query can differ if the user is trying to learn more about a specific research domain, as opposed to if they are trying to find more authoritative sources to cite for a research concept of their interest.

However, such an approach proposed by this framework relies heavily on the search interface, or API, of the public databases in question. How a certain database collates its results and how many fields of query the API accommodates can become key bottlenecks to the overall quality of the recommendations provided by the framework. This distils down to the lack of control the framework has over how it can manipulate the data within a public database.

Utilising bag-of-word models and k-NN

Lee et al. [13] propose an architecturally-similar system that also suggests how the keyword extraction and similarity measurement stages could be implemented differently, compared to the previous proposal. The system can be summarised in the following steps:

1. a set of candidate research papers is downloaded using a self-implemented web data gatherer that crawls through public databases such as IEEE Xplore and the ACM Digital Library,

2. the gathered papers are then represented as vectors using bag-of-word models,

3. a user-item matrix is built based on the end-user's research interests and preferences,

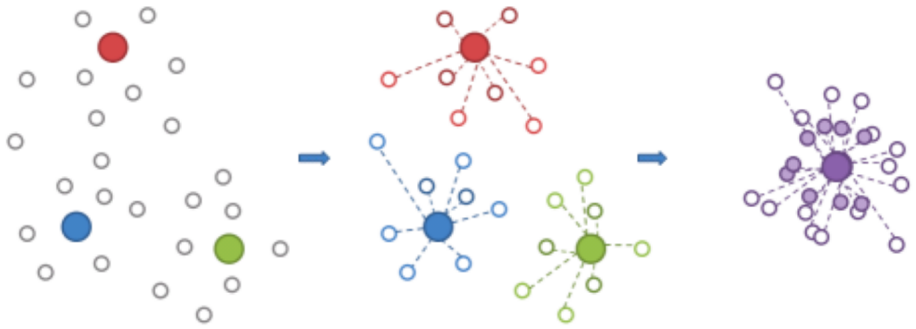

Figure 2.1: A visual representation of how K-means clustering and k-NN can be used to list recommendations.

4. both K-means clustering and k-NN algorithms are used to select a fixed number of to-be-recommended papers, based on their similarities to the highest-scoring items in the user-item matrix,

5. and finally, the cosine similarity measure is used to rank the selected candidate papers, which are then sent to the end-user as output.

The proposal mentions the use of cosine similarities on the bag-of-word vector representations as a measure of similarity. This is similar to the previous proposal. However, one major difference that is worth analysing between this proposal and the previous is the use of K-means clustering and k-NN algorithms to select candidate papers to recommend to the end-user. Considering the more simple example of having to recommend k related papers based on a single target paper, a k-NN algorithm provides a straightforward method of doing that exact task. However, this may not work directly when having to select a number of candidate papers based on a arbitrary number of target papers, as is the case when using a user-item matrix. Lee et al. [13] propose the use of K-means clustering before running the k-NN algorithm. Each target paper in the user-item matrix is considered as the centroid of their own cluster. All candidate papers are then compared in terms of similarity and assigned to a cluster whose centroid they are closest to. Finally, k-NN is run on each cluster. Assuming that there are n clusters, this means that the system is able to recommend $n \times k$ related papers as its output. Figure 2.1 from [13] illustrates this process visually.

Although this method of handling multiple target papers is easy to understand and simple to implement, it poses some limitations towards the quality of the recommendations. For example, consider a user-item matrix that contains two target papers A and B that belong to the same research domain. This clustering method is unable to extract commonalities of both A and B, and calculate similarity measures based on those commonalities. Instead, the method is only able to find papers that are related to A and those that are related to B separately. This could be a concern, as an end-user would most likely want to find related papers based on all the research content in their library as a whole, not just based on individual papers on their own, one by one.

One element to note is that this system prioritises recommending research papers that are similar to papers that are written by the end-user, and in particular,

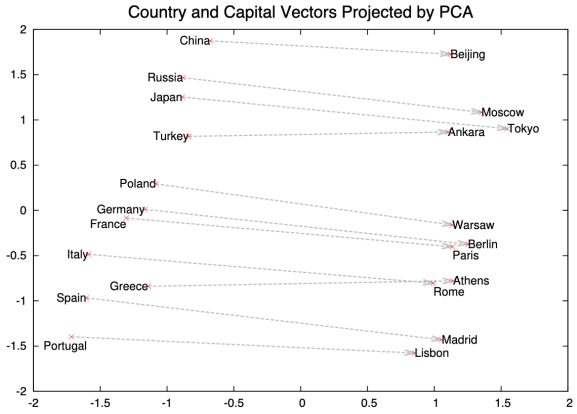

Figure 2.2: A visual representation of word embeddings produced by word2vec.

ranking papers that are written by the end-user themselves above those written by other authors. The system assumes that its end-users are most likely research authors themselves, and is more suited towards such a use case. However, this can easily be modified for our Zotero-oriented use case. For example, instead of populating the user-item matrix with papers that are written by the end-user, our system can populate it with papers and resources that are in the end-user's Zotero library.

Utilising deep learning

Hassan [14] proposes an architecturally simple content-based recommender system, where the bulk of the processing and prediction work is done using deep learning. The architecture can be summarised as follows:

1. the end-user's research interests are modelled in the form of a user profile,
2. a set of to-be-recommended papers is downloaded from a public database, such as PubMed, through web crawling,
3. the title and abstract of the set of to-be-recommended papers is represented as vectors through word embedding via the word2vec [15] library and LSTM techniques,
4. and finally, papers in the user's profile and papers in the to-be-recommended set are compared using cosine similarities, and thus ranked and sent to the end-user as output.

The deep learning element of this architecture comes in the form of the word2vec library, which consists of a continuous skip-gram language model that needs to be trained from a large corpus of text. This trained model is then used to predict and generate word embedding representations. word2vec is a form of deep learning because the features of input keywords are extracted automatically. Figure 2.2 from [15] provides an example of how keywords are represented in the word2vec library.

While the proposed architecture here is structurally simpler, and is less reliant on third-party services such as the querying interface of external public databases, the

proposal does come with some areas of ambiguity that will need to be addressed if one were to move towards this approach. Firstly, the proposal does not concretely explain how to-be-recommended papers should be extracted from the PubMed database through the use of the crawler, nor the process of deciding what papers should be considered as part of the to-be-recommended set. There is a possibility that this process also relies on the use of the querying interface of the PubMed database, and would need to be explored further. Secondly, the fundamental issue of using word embedding techniques such as word2vec is that it has to be trained with the corpus of text that you or your use case are interested in. In the case of the proposal, they suggested training the model using a sample collection of scientific publications. However, this drastically limits the research areas that the recommender system could be used for, and also introduces the additional task of collecting a large sample of publications to be used as training material. This process would require a lot of consideration into whether the training material is biased and whether it covers all of the research domains the system should be able to recommend.

2.2 Conclusion

Whilst each of the existing proposals as mentioned above suggest different methods of implementing certain components of their systems, they do share a somewhat similar underlying structure. Namely, all the recommender systems are composed of the following sub-components:

1. a data gatherer connected to some public database(s),
2. a model that abstractly represents the gathered data,
3. and a mechanism to compare and rank data according to their similarities to the target.

Data essentially goes through each of the above components in that order, in series. This provides some insight as to how our recommender system should also be structured from a high level.

However, one vital improvement that can be made to almost all the proposals mentioned above is their exclusive use of STEM-related research databases in their data gatherers, affecting how their modellers are trained and tuned, and therefore restricting the research domains in which these recommender systems can be used. For example, Lee et al. [13] use IEEE Xplore, an engineering- and technology-oriented research library, as their database source, and solely use computer science-related papers as part of their evaluation process, such as those related to ML, HCI and DB. Nascimento et al. [7] also use IEEE Xplore, as well as ScienceDirect, as their database source as part of their evaluation. Meanwhile, Hassan [14] uses the PubMed database, a library oriented towards biological and medical sciences. Therefore, improvements can be made towards future recommender systems being more inclusive, and covering more research domains such as those in arts, humanities and social sciences. This could potentially be done simply by changing the data sources that are used to shape the models and prediction algorithms in these proposals.

Chapter 3

Project Plan

This chapter discusses the work that needs to be done in order to implement our recommender system, and proposes a timetable with the key milestones and major tasks of the project. Table 3.1 provides a preliminary timetable for both research and development work for the project.

Start Date	**Due Date**	**Task Description**
30/01/2023	13/02/2023	Further assessment and exploration of the approaches used in each stage of existing recommender systems
13/02/2023	27/02/2023	Propose a high-level architecture for the back-end of our Zotero-oriented system, and the technologies used in each stage
27/02/2023	13/03/2023	Experiment with the Zotero JavaScript API, and develop skeleton code for the front-end of the plugin
27/03/2023	24/04/2023	Develop an MVP for the back-end recommender system using our chosen technologies
03/04/2023	10/04/2023	Propose a plan for conducting experiments that will evaluate the system's performance
24/04/2023	01/05/2023	Develop an MVP for the front-end of the plugin and connect it with the back-end service
01/05/2023	05/06/2023	Continuous evaluation of the system's performance and development work to implement improvements
05/06/2023	19/06/2023	Write-up and publication of the final report of the project

Table 3.1: A preliminary timetable for the project.

The preliminary plan for implementing our Zotero-oriented recommender system is to separate the front-end and back-end of the Zotero plugin, where the front-end

would be implemented using the Zotero JavaScript API, and the back-end using whatever would be most suitable for the chosen approach of our recommender system (e.g. Python, if we were implementing some sort of machine learning model as our back-end). These two components would be connected via a RESTful API in a client-server fashion, where the back-end would be hosted publicly on a cloud server and accessed via the Internet, and the front-end would be part of the Zotero client application that is installed on the end-user's machine. This approach would allow us to minimise conflict between the two aspects of the project, allowing us to conduct research, exploration and development work for both aspects separately.

Should there be sufficient time, an interesting extension to this project would be to implement multiple versions of the recommender system using different NLP modelling techniques (e.g. word2vec, bag-of-word models, skip-gram models etc.) and ranking algorithms (e.g. cosine similarity, TF-IDF, k-NN etc.), and compare the performances of each approach to see which would be the most effective.

Chapter 4

Evaluation Plan

A method of evaluating the performance of our recommender system could be something similar to that as suggested and ran by Nascimento et al. [7], where a set of participants supply a research paper as input to the system, and give a rating for each recommendation that is sent back to them. The exact format of the input to the evaluation system could also be one of the following:

- a single research paper,
- a single research paper, with a list of keywords that represent the words in the paper that would have been highlighted by the user on Zotero using the 'Highlight Text' tool,
- a collection of papers, which would simulate the system giving recommendations based on an end-user's entire library or sub-collections.

Details such as who the participants will be, how they are recruited, how the experiments are conducted (e.g. in-person or online), and how we restrict the subject domain of the papers used for evaluation need to be discussed further and agreed upon.

The ratings provided by the participants can then be aggregated and used to calculate metrics of performance, such as precision and recall. For example, for each recommendation they receive, each participant can give a number rating, ranging from 0 to 5 inclusively, corresponding to how relevant and useful they found the recommendation to be. For example, 0 would correspond to the recommendation being not *at all* relevant, 3 would correspond to it being relevant, and 5 would correspond to it being both relevant and useful to the user. Using these ratings, the following metrics could then be calculated, where R is the set of all participant ratings, and where a 'relevant' recommendation is defined as having been given a rating of 3 or above:

$$\text{average user-reported score} = \frac{1}{N} \sum_{r \in R} r$$

$$\text{precision} = \frac{\text{number of relevant recommendations}}{\text{number of recommendations}}$$

Having defined how precision may be calculated, the recall metric may prove to be more difficult to calculate. Following the above definition for precision, the recall metric should be defined as:

$$\text{recall} = \frac{\text{number of recommendations that were relevant}}{\text{number of papers in the database that were relevant}}$$

However, there is no feasible method of finding out how many papers in the entire database the participant *would have* rated as relevant, unless the participant were to go through the entire database and provide ratings for every paper.

Chapter 5

Ethical Issues

This chapter discusses the wider ethical, legal, professional and societal issues surrounding the project and its accompanying research.

5.1 The Zotero code base and API

Zotero is an open-source project managed by the Corporation for Digital Scholarship, with an open-source, publicly viewable code base [16]. The Zotero client is distributed under the GNU Affero General Public License. Should source code from the Zotero client itself be extracted and used in our project, our project must 'assert copyright on the software' in the documentation before we can 'copy, distribute and/or modify the software', and that 'modified source code becomes available to the community' [17]. This may or may not be relevant to our project, as our software deliverable is simply a plugin component to the Zotero client, rather than an extension or modification to the existing source code. This will depend on whether the source code of our plugin requires any components of internal source code from the client itself.

The Zotero name is a registered trademark. Therefore, the software deliverable from our project should not be branded, labelled or documented as a Zotero product.

5.2 Use of proprietary research information

As mentioned by Nascimento et al. [7], existing recommender systems for research often utilise 'privileged' information from users' libraries in order to generate recommendations, including private document collections, sets of citations, sets of user profiles etc. Not only does the process of storing such data have to comply with GDPR laws and regulations, there are more fundamental privacy concerns that allude from such an implementation, such as if the academic research in a researcher's library involves personal data from others, or if the research in question is meant to be kept unpublished due to corporate reasons (for example, research being done for a private company). Therefore, the software design and architecture of our project will aim to not utilise this information when generating recommendations, and instead, solely rely on the current user's personal research library and publicly available data only.

Bibliography

[1] Zotero | Your personal research assistant. URL https://www.zotero.org/. pages 1

[2] plugins [Zotero Documentation], . URL https://www.zotero.org/support/plugins. pages 1

[3] dev:client coding:plugin development [Zotero Documentation], . URL https://www.zotero.org/support/dev/client_coding/plugin_development. pages 1

[4] Erik Schnetter. Zotero Citation Counts Manager, 2022. URL https://github.com/eschnett/zotero-citationcounts. Last accessed 18 January 2023. pages 1

[5] Joscha Legewie. ZotFile, 2022. URL https://github.com/jlegewie/zotfile. Last accessed 18 January 2023. pages 1

[6] windingwind. Better Notes, 2023. URL https://github.com/windingwind/zotero-better-notes. Last accessed 18 January 2023. pages 1

[7] Cristiano Nascimento, Alberto H.F. Laender, Altigran S. da Silva, and Marcos André Gonçalves. A Source Independent Framework for Research Paper Recommendation. In *Proceedings of the 11th annual international ACM/IEEE joint conference on Digital libraries*, pages 297–306, Ottawa Ontario Canada, June 2011. ACM. ISBN 978-1-4503-0744-4. doi: 10.1145/1998076.1998132. URL https://dl.acm.org/doi/10.1145/1998076.1998132. pages 3, 5, 9, 12, 14

[8] 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*, 17(6):734–749, June 2005. ISSN 1041-4347. doi: 10.1109/TKDE.2005.99. URL http://ieeexplore.ieee.org/document/1423975/. pages 4, 5

[9] P Resnick, N Iacovou, M Suchak, P Bergstrom, and J Riedl. GroupLens: an open architecture for collaborative filtering of netnews. In *Proc. ACM Conf. on Computer Supported Cooperative Work*, pages 175–186. 1994. pages 4

[10] dev:web api:v3:oauth [Zotero Documentation], . URL https://www.zotero.org/support/dev/web_api/v3/oauth. pages 4

[11] Gerard Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley Longman Publishing Co., Inc., USA, 1989. ISBN 0201122278. pages 5

[12] Upendra Shardanand and Pattie Maes. Social information filtering: Algorithms for automating "word of mouth". In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, page 210–217, USA, 1995. ACM Press/Addison-Wesley Publishing Co. ISBN 0201847051. doi: 10.1145/223904.223931. URL https://doi.org/10.1145/223904.223931. pages 5

[13] Joonseok Lee, Kisung Lee, and Jennifer G. Kim. Personalized Academic Research Paper Recommendation System, April 2013. URL http://arxiv.org/abs/1304.5457. arXiv:1304.5457 [cs]. pages 6, 7, 9

[14] Hebatallah A. Mohamed Hassan. Personalized Research Paper Recommendation using Deep Learning. In *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*, pages 327–330, Bratislava Slovakia, July 2017. ACM. ISBN 978-1-4503-4635-1. doi: 10.1145/3079628.3079708. URL https://dl.acm.org/doi/10.1145/3079628.3079708. pages 8, 9

[15] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed Representations of Words and Phrases and their Compositionality. In C. J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf. pages 8

[16] Corporation for Digital Scholarship. Zotero, 2023. URL https://github.com/zotero/zotero. Last accessed 18 January 2023. pages 14

[17] Free Software Foundation. GNU Affero General Public License, 2023. URL https://github.com/zotero/zotero/blob/master/COPYING. Last accessed 18 January 2023. pages 14