GUapp: A Conversational Agent for Job Recommendation for the Italian Public Administration

Vito Bellini, Giovanni Maria Biancofiore, Tommaso Di Noia,
Eugenio Di Sciascio, Fedelucio Narducci, Claudio Pomo
Politecnico di Bari - Bari, Italy
name.surname@poliba.it

***Abstract*—`GUapp` is a platform for job-postings search and recommendation for the Italian public administration. The platform offers recommendation services with the aim of matching user skills and requests with job positions available in a given period of time. The recommender system implemented in `GUapp`, based on Latent Dirichlet Allocation, computes the k-nearest neighbors job positions most similar to the user profile. Furthermore, in order to improve the user experience, `GUapp` implements a chatbot whose goal is to allow users to interact with the app through natural language. Thanks to that, the search and recommendation process becomes incremental and the user can add new requirements at each stage of the interaction. In this paper we present `GUapp`, its recommender system, and the chatbot developed for achieving an effective interaction with the user. In the next future, we will carry out in-vivo and in-vitro experiments for evaluating the system and its components in deep.**

***Index Terms*—Recommender System, Public Administration, Topic Modeling, Conversational Recommender System**

I. Introduction

The information overload is a well-known problem that impacts the digital experience of users when they need to find interesting items in a large set of possible options [1]. That is the case of looking for a book to read, a digital camera to buy, a movie to watch, and so on. Another scenario when this problem is particularly felt is when, for example, a user is looking for a new job. In this context the only strategy people can adopt is to manually selecting interesting and not interesting job calls, thus resulting in a clumsy user experience. Moreover, job calls usually have a limited time span for applying and, for this reason, it is important to constantly look for interesting job openings over time as quick as possible.

The `GUapp` platform has been designed and developed to find and discover job positions among job offers in the Italian public administration[1].

In the literature, this problem has been investigated in the information-retrieval (IR) and recommender-system (RS) research areas. In the first case, systems do not take into account the user past preferences and retrieve the most relevant items according to the user query. On the other hand, even though the goal of a RS might seem similar to that of an IR system, a profile of the user is built. This allows to propose services tailored on the specific characteristics of the user, and in our case to provide a *personalized* ranked list of items.

The `GUapp`'s RS classifies jobs according to their topics and a k-Nearest Neighbor (k-NN) methods is used to compute a personalized recommendation list. That list is daily updated and provided to the user. The user's past preferences are used for ranking the job positions she might be interested in. Moreover, `GUapp` offers a natural-language based interaction through a chatbot. The `GUapp`'s chatbot allows the user to define her interests, describe her skills, and filter out results that do not match whit her requirements. Obviously, different back-end services of `GUapp` have to be invoked in order to satisfy the user requests.

`GUapp` is available both as mobile app and as responsive Web client so that the UI is dynamically adapted to the user behavior in order to proactively show the right view as the app opens. Hence, the overall experience of the user is improved by anticipating her needs. Furthermore, the first view of the app is adapted to the previous user interactions. Hence, `GUapp` shows the list of new jobs, the recommendations, or the chatbot at startup, based on the user preferences. An evaluation of the whole system and the individual components has not been carried out yet. However, this is our next step as future work. The rest of this paper is organized as follows: Section II introduces the job-recommendation task, whereas Section III analyses the most relevant researches. Section IV describes the architecture of the system. The interaction with the app is described in Section V. Finally, Section VII outlines the principal conclusions and the future work.

II. The Job Recommendation Task

A set of peculiar aspects of the job-recommendation task has been defined in the literature [2], [3]:

- acquisition of the user skills and abilities;
- bidirectional recommendations, for the recruiter and for the candidate;
- recommendations based also on the relational aspects between the candidate and the team members;
- candidate considered only one time, contrary to other domains such as movie or music recommendation.

[1]All the documents are freely available online at https://www.gazzettaufficiale.it/30giorni/concorsi

Figure 1 graphically depicts the workflow of a job recommender system. In the context of GUapp, the attention is focused on the first aspect, namely the acquisition of user skills and abilities. Indeed, the goal of GUapp is to propose relevant job positions to the candidate, thus a crucial step is to build a user profile as accurate as possible. Given the nature of the job positions managed by GUapp, provided by Public Administrations that are obliged to hold public competitions, the bidirectional recommendation is not implemented. Also the team-relation matching, is not considered, since it is a characteristic of private organizations.

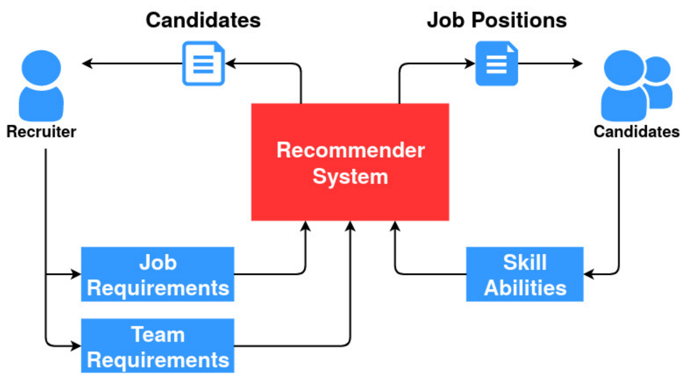


Fig. 1: The Job Recommendation Task

III. Related Work

The work proposed in this paper cuts across two principal research areas: recommender systems and conversational agents. Recommender Systems (RSs) can in turn be grouped into two main classes, according to the paradigm implemented: Collaborative Filtering (CF) and Content-based (CB). Independently from the paradigm adopted, the aim of any RS is to help users to address the information overload problem. RSs prioritize the delivery of information for individual users based on their learned preferences [4]. Hence, RSs support the user during the decision making process when she has to decide among a large set of different options. In CF the recommender system exploits the user community in order to identify items potentially interesting for a given individual. This kind of RS is generally implemented in online platforms like Amazon [5], Netflix [6], and so on. The basic idea behind these recommendation strategy is that similar users like similar items. Conversely, CB RSs do not need any community and recommendations are based on the matching between user preferences and textual content associated to the items [1]. For GUapp we implemented a CB RS since the goal of the application is to match the user requests/preferences with the textual description of the job call. The specific task of job recommendation is widely investigated in the literature given the large diffusion of internet-based recruiting platforms [7]. In the past, the most used approaches for job recommendation were based on boolean search and filtering [2]. Later, the attention has been focused on the problem of catching the user preferences and building a user profile. In [8] the authors propose a system that builds the user profile by passively detecting click-stream and read-time behaviour of users. This is the same strategy adopted in GUapp with the goal of reducing the user effort for defining her preferences. Malinowski et al. [2] builds a multi-slot profile with information about demographic data, job experience, language, and IT skills. Similarly, in GUapp, this kind of information is acquired from the LinkedIn social network. This strategy is also useful for addressing the cold-start problem. As regards the recommendation algorithm, Amato et al. [9] compared several algorithms for matching candidate profiles with job descriptions. From their study, LDA emerged as the best algorithm in terms of precision, recall, and f-measure, compared to other machine-learning and rule-based approaches. LDA is also used in [10] for building a set of features of job postings and user profiles. A hybrid approach is proposed in [11]. The authors combine content-based and KNN in a fashion very similar to that used in GUapp.

The second research area relevant for this work is that of Conversational Agents (CAs). CAs are software agents able to interact with the user through natural language. Generally, CAs can be classified in two main classes: end-to-end and modular systems [12]. The former are generally based on neural networks [13], [14], [15], [16] and they learn a dialog model from a set of past conversations [17]. These systems demonstrated good performances for chit-chat dialogs. The modular systems are pipeline-based agents and are composed of a set of modules, each with a specific function [18], [19], [20], [21]. Modular agents generally work fine for goal-oriented system. A Conversational Recommender System (CoRS) can be surely classified as a goal-oriented system, whose goal is precisely to make recommendations in a given domain. The main difference between a CoRS and a traditional RS is the interaction with the user that is more efficient and natural [22]. Accordingly, the construction of the user profile is incremental and the user can express her preferences at different stages of the recommendation process by an interactive human-like dialog [23]. That is the aspect that we tried to emphasize by implementing a chatbot in GUapp. There is not a very extensive literature on CoRSs. In [24] the authors propose the idea of combining Virtual Assistants and CoRSs. Recently, CoRSs have been effectively used and tested in different domains such us music, movie, and book [25], [26]. However, to the best of our knowledge, GUapp represents the first attempt of implementing a CoRS for the job-recommendation task.

IV. The GUapp's Architecture

In Figure 2 the GUapp's architecture is sketched. We can find five main components: the Orchestrator, the Chatbot, the Recommender System, the User Profiler, and the Crawler.

The *Orchestrator* is a sort of hub for GUapp. It manages the interaction between the different components of the system in order to satisfy the user requests. Hence, for example, it invokes the recommender system when user asks for receiving a personalized list of job positions based on the information stored in her profile.

The *Chatbot* is the component of GUapp which allows user to interact through natural language. It is powered by DialogFlow[2] and performs two main tasks: intent and entity recognition. The *Intent Recognizer* (IR) analyzes the request of the user expressed in natural language and understands her goal. That is a crucial step in order to identify the back-end services to be invoked for accomplishing the request. For example, if the user writes *I am looking for a new job in Bari* the IR identifies as intent *new_job*. Actually, at this stage, GUapp is able to recognize three intents: *welcome*, *new_job*, *refine*. The intent *welcome* is automatically recognized when user activates the *chatbot*, which results in starting the conversation with a welcome message from the agent. The *new_job* intent is activated when the user sends a message with the goal of looking for a new job, as in the example above. Finally, *refine* is an intent that is activated when the user adds new requirements to her previous request. As an example, after the first message *I am looking for a new job in Bari*, user might write *I prefer jobs at university*. In that case, *refine* is triggered and this new request is added to the user request. Once the intent has been recognized, the *Entity Recognizer* (ER) is invoked in order to check whether the sentence contains mentions to real-world entities. ER is implemented through DialogFlow as well. It adopts a fuzzy-matching strategy in order to identify entities in the user sentence. The set of entities is explicitly defined in a vocabulary. In the case the match succeeds, the recognized entity is returned. In the example above, the entity is *Bari*, a city of southern Italy. At this time, the entities in the vocabulary are related only to cities and job-place categories (e.g. university, municipality).

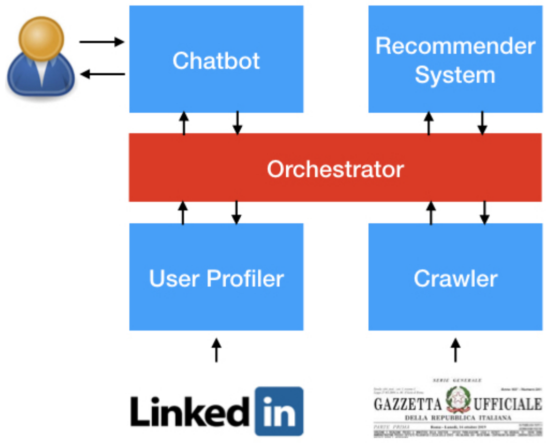


Fig. 2: The GUapp's Architecture

The third component of the system is the *User Profiler* (UP). UP stores and updates information about user preferences and habits. The user profile is built by analysing implicit and explicit feedback. As for the explicit ones, user can bookmark job positions and give "Like" feedback. Furthermore, explicit feedback is also gathered by analysing natural language messages sent by the user to GUapp. Indeed, user can *refine* the retrieved list of relevant job positions as described above. Other important sources for getting information about user are social networks, and more specifically LinkedIn[3]. Indeed, GUapp offers authentication by LinkedIn APIs. Thanks to this integration, the user profile in GUapp can be fed with data coming from public information stored in the social network. GUapp is able to get information about reachable cities and skills.

This is an effective strategy for addressing the *cold-start* problem. As a matter of fact, when user signs in to the app for the first time, her profile is empty (i.e. *cold-start* situation). Thanks to the LinkedIn integration, GUapp is able to rank job calls by exploiting information stored in the social profile.

As for the implicit signals, in the mobile app we measure how long users read articles and from which view. More specifically, for each user we record: i) list of open positions, ii) searches, iii) recommendations.

The *Recommender System* is another core component of GUapp. Its behaviour and implementation is described in Section VI.

Finally, GUapp implements a *Crawler* that daily extracts the job positions from *Gazzetta Ufficiale*, the official journal of record of the Italian government.

V. GUAPP@WORK

As mentioned above, every day new job opportunities in the public administration are announced. Our back-end infrastructure crawls new public calls daily. GUapp offers two kind of UI: a Web Interface and an Android Application. Let us consider a running example in order to explain how GUapp works.

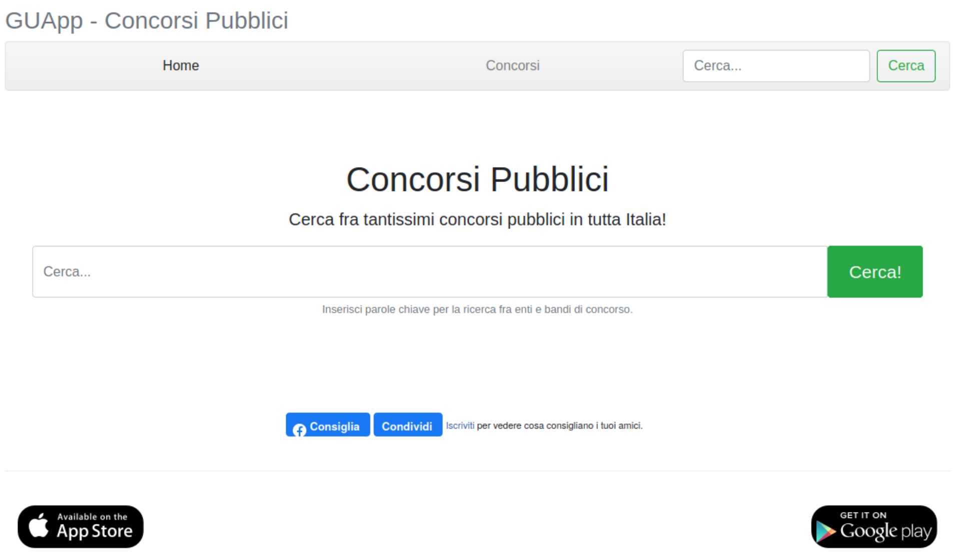
Paolo is looking for a new job and he visits the GUapp Web Site[4]. The home page is shown in Figure 3. Here Paolo types *Bari* in the search box and he receives a list of open job positions located in Bari as shown in Figure 4. Calls are sorted in descending order of deadline, putting in the first positions those that expire first. Paolo browses the list of retrieved calls, but since the list is very large, he changes his query in the hope of better filtering the results. Since he would find a job position in a health institution, Paolo types as keywords *healthcare job calls in Bari*, which results in a more refined list of open job positions. Paolo finds an interesting job call, thus he clicks on *Details* (i.e. Dettagli) and all the information published for that job position are shown in a new tab as well as a link to the official document provided from the recruiter.

The details include the job place, type, the deadline for applying, and a summary. However, the selected call does not fully satisfy Paolo, thus he explores the *Competitions* (i.e. Concorsi) tab, which contains all the available job calls provided by the *Gazzetta Ufficiale*. The GUapp Web Site can not really helping Paolo in finding his job, since it was

[2]https://dialogflow.com/

[3]https://www.linkedin.com/

[4]https://guapp.sisinflab.poliba.it


GUApp - Concorsi Pubblici

Home | Concorsi | Cerca... | Cerca

Concorsi Pubblici

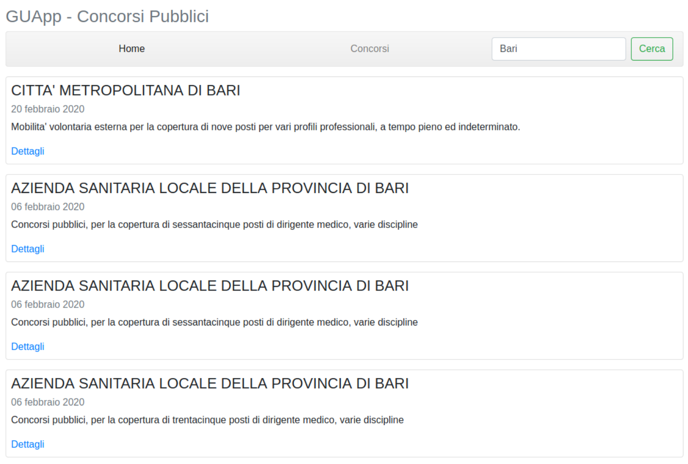
Cerca fra tantissimi concorsi pubblici in tutta Italia!

Cerca... | Cerca!

Inserisci parole chiave per la ricerca fra enti e bandi di concorso.

Consiglia | Condividi | Iscriviti per vedere cosa consigliano i tuoi amici.

Fig. 3: `GUapp` Web Site


GUApp - Concorsi Pubblici

Home | Concorsi | Bari | Cerca

CITTA' METROPOLITANA DI BARI
20 febbraio 2020
Mobilita' volontaria esterna per la copertura di nove posti per vari profili professionali, a tempo pieno ed indeterminato.
Dettagli

AZIENDA SANITARIA LOCALE DELLA PROVINCIA DI BARI
06 febbraio 2020
Concorsi pubblici, per la copertura di sessantacinque posti di dirigente medico, varie discipline
Dettagli

AZIENDA SANITARIA LOCALE DELLA PROVINCIA DI BARI
06 febbraio 2020
Concorsi pubblici, per la copertura di sessantacinque posti di dirigente medico, varie discipline
Dettagli

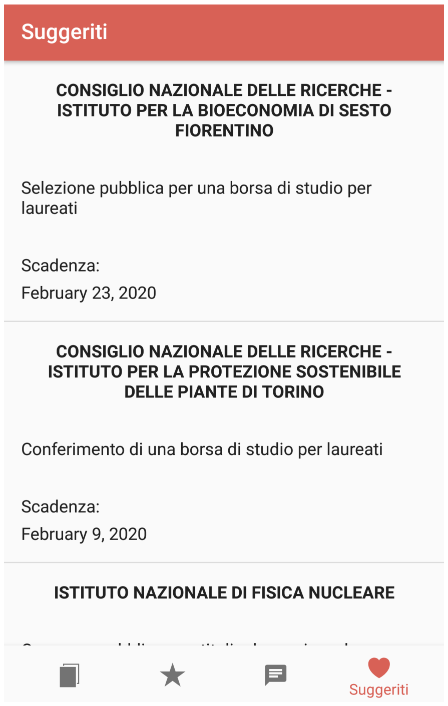
AZIENDA SANITARIA LOCALE DELLA PROVINCIA DI BARI
06 febbraio 2020
Concorsi pubblici, per la copertura di trentacinque posti di dirigente medico, varie discipline
Dettagli

Fig. 4: Searching results in `GUapp` Web Site

designed as an open-job-position search engine, thus Paolo ends his search hoping to find an interesting job call in the next days. Some days later, Paolo visits again the `GUapp` Web Site, but he has to start over his search since the website does not store his user profile and his previous interactions have not been recorded.

Let us now to consider the interaction through the Android app. The scenario is the same, thus Paolo is looking for a new job and he launches the `GUapp` Android Application on his smartphone. `GUapp` asks the Paolo's credentials for the login. Indeed, the main difference between the mobile app and the web-site version is the capability of building a user profile. That is a great advantage for the user since he can receive personalized services. Here Paolo has two options: to create his own account providing e-mail address and password, or to sign in through the LinkedIn account. Paolo decides to sign in to the app through his LinkedIn account and he chooses the information to share with `GUapp`. He grants to `GUapp` permissions for getting information on reachable locations and skills. Now, the main default view of the app is a list of all job calls ordered by deadlines. Since Paolo does not want to browse a huge amount of items provided by the system, he selects the *Recommendations* (i.e. Suggeriti) view, which shows a list of recommended open job positions computed on his preferences, as depicted in Figure 5a. It is worth to note that the recommendation list is also daily pushed to Paolo in a proactive way. The application exploits data gathered from LinkedIn in order to build a first list of recommendations. The list includes job calls whose location and requirements match with Paolo's approachable cities and skills got from LinkedIn. Here Paolo browses the recommended items that can potentially match his interests, and once he has found some interesting job calls, he taps them for getting more details. Hence, `GUapp` shows more information for the selected job calls in a fashion very similar to the web app. Paolo now expresses his preferences by tapping the corresponding like button for the calls deemed interesting. After he rated all the open job positions in which he is interested in, Paolo selects the *Favourite* (i.e. Favoriti) view for marking on his agenda the deadlines.

`GUapp` implements also a conversational interface through a chatbot (Figure 5b). Accordingly, user can dialog with `GUapp` performing requests in natural language. Users can ask for a job position in a given location or saying her preferences about a specific job opportunity by simply writing natural language sentences. The main difference compared to the search box is that the chatbot allows to acquire the user preferences incrementally. Hence, Paolo who is still looking for a job, selects the `GUapp Bot` view. `GUapp Bot` asks to exploit the LinkedIn's information for the current session. However, Paolo ignores this option and sends the message *I am looking for a job at university*. Hence, `GUapp` shows a ranked list of


Suggeriti

CONSIGLIO NAZIONALE DELLE RICERCHE - ISTITUTO PER LA BIOECONOMIA DI SESTO FIORENTINO

Selezione pubblica per una borsa di studio per laureati

Scadenza:
February 23, 2020

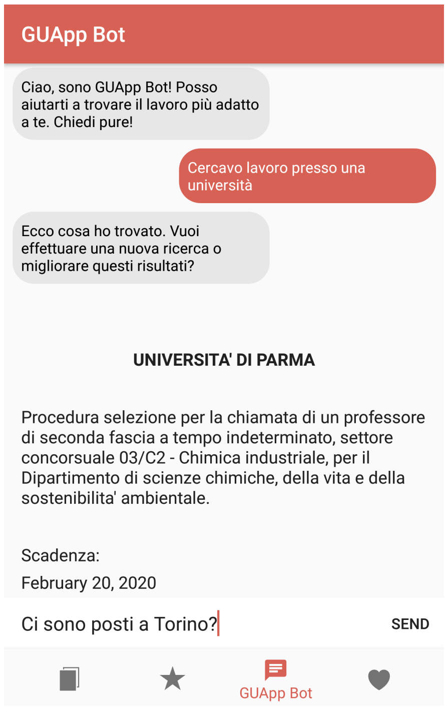
CONSIGLIO NAZIONALE DELLE RICERCHE - ISTITUTO PER LA PROTEZIONE SOSTENIBILE DELLE PIANTE DI TORINO

Conferimento di una borsa di studio per laureati

Scadenza:
February 9, 2020

ISTITUTO NAZIONALE DI FISICA NUCLEARE

(a) Recommendations


GUApp Bot

Ciao, sono GUApp Bot! Posso aiutarti a trovare il lavoro più adatto a te. Chiedi pure!

Cercavo lavoro presso una università

Ecco cosa ho trovato. Vuoi effettuare una nuova ricerca o migliorare questi risultati?

UNIVERSITA' DI PARMA

Procedura selezione per la chiamata di un professore di seconda fascia a tempo indeterminato, settore concorsuale 03/C2 - Chimica industriale, per il Dipartimento di scienze chimiche, della vita e della sostenibilita' ambientale.

Scadenza:
February 20, 2020

Ci sono posti a Torino?

(b) Chatbot

Fig. 5: Screenshots of the Android application.

job calls. The GUapp behaviour is the same of the search box so far. However, Paolo can refine his request. He writes *Are there positions in Torino?*. At this point GUapp has a new information to exploit and it uses it for refining the list of retrieved job positions. In the case Paolo wants to perform a new search, he sends the message *new search* and he can start over.

VI. The Document-Based Recommender System behind GUapp

The recommender system behind GUapp helps users in finding interesting job postings among all the calls daily published. GUapp recommends jobs by analyzing unstructured text that describes both requirements for the job application and duties of the job itself.

A pre-processing step is performed on the text and it consists in removing stop-words from all the documents. Furthermore, words having a frequency greater than 95% and words that appear in the collection less than 2 times are removed, given to their low informative power. The size of vocabulary is limited to $1,000$ features (words). This means that the final vocabulary $V = [w_1, \dots, w_m]$ contains the m top words in the collection, ordered by their term frequency in the whole dataset. Each document d is represented in a Bag-of-Words matrix $R \in \mathbb{N}^{n \times m}$ where n denotes the number of documents and m the size of V. Each entry $r_{ij} \in R$ of the matrix is an integer that counts the number of occurrences for the term w_j in the document d_i.

Since we have no training data, the only option we can leverage to generate recommendations is to exploit *Unsupervised Learning* algorithms. We decided to use topic modeling to build clusters of words. Each document is then represented by all the topics t, each having a specific weight.

We adopted LDA [27] for topic modelling. It is a generative probabilistic model that tries to find groups of words that appear frequently together across different documents. It is widely adopted to link text in a document to specific topics. LDA builds a *topic per document* Q and a *word per topic* P model, as shown in Figure 6. The algorithm has only one hyperparameter k representing the number of topics modeled. In GUapp, we empirically found that 10 is a good number of prior topics in our setting.

Given n documents, m words, and k prior number of topics, the model is trained to predict: i) the distribution of words for each topic; ii) the distribution of topics for each document. LDA works by assigning a random topic t to each word of the document. Iteratively, for each word w and for each document d, it computes $p(t \mid d)$ as the proportion of words in d that are currently assigned to t and $p(w \mid t)$ as the proportion of word w assigned to the topic t over the whole collection. Eventually, it reassigns each word w to a new topic, using the probability that topic t generates word w, as

$$t(w) = p(t \mid d) \cdot p(w \mid t)$$

In GUapp we normalize the matrix Q which represents the "document to topic" distribution so that topics for each row (document) are scaled between 0 and 1. Since we want

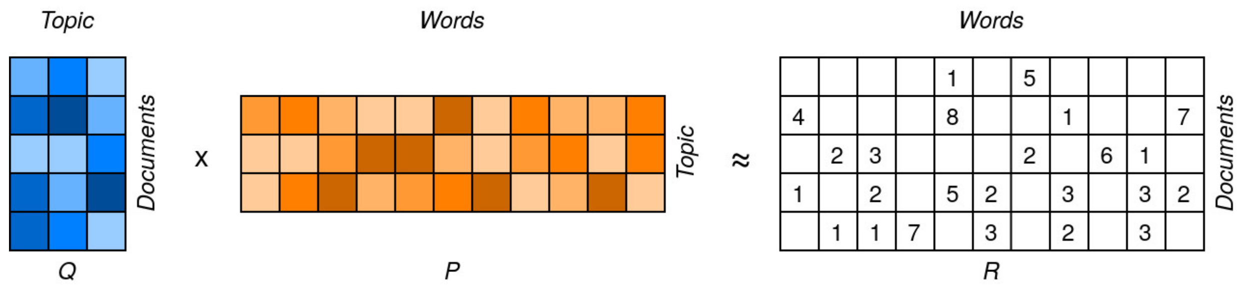


Fig. 6: Topic modeling with Latent Dirichlet Allocation.

to recommend only open positions to which the user might want to apply, we filter out all the expired documents from R and we compute the cosine similarity on the new matrix. For each topic, we select the top-50 documents never rated by the user, ranked according to their similarity score in the matrix. Documents with a score equal to zero are filtered out. Thus, we generate a top-50 recommendation list of job positions for each user. It is noteworthy that, following [28], the topics of the recommended documents are in the same proportion of the user's liked topics.

VII. Conclusion and Future Work

In this paper we presented a platform for searching jobs in the Italian public administration. `GUapp` is composed by a recommender system that suggests relevant jobs to the users, and a mobile app client that allows users to interact with the platform. One of the most interesting aspect of the mobile version is the integration of a chatbot that makes the interaction more natural. Indeed, the preference elicitation becomes an incremental process, with the possibility of refining and improving the user requests. Although very useful from a practical perspective, the entire ecosystem results to be an ideal candidate for AB tests related to user interactions as well as for the recommendation engine to be adopted in similar scenarios. Finally, we are setting up a platform in order to share the dataset coming from `GUapp` usage with the industrial and academic community.

Acknowledgments. The authors acknowledge partial support of the following projects: Innonetwork CONTACT, Innonetwork APOLLON, ARS01_00821 FLET4.0, Fincons Smart Digital Solutions for the Creative Industry.

References

[1] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich, *Recommender systems: an introduction*. Cambridge University Press, 2010.

[2] J. Malinowski, T. Keim, O. Wendt, and T. Weitzel, "Matching people and jobs: A bilateral recommendation approach," in *Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06)*, vol. 6. IEEE, 2006, pp. 137c–137c.

[3] T. Keim, "Extending the applicability of recommender systems: A multilayer framework for matching human resources," in *2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07)*. IEEE, 2007, pp. 169–169.

[4] D. H. Lee and P. Brusilovsky, "Fighting information overflow with personalized comprehensive information access: A proactive job recommender," in *Third International Conference on Autonomic and Autonomous Systems (ICAS'07)*. IEEE, 2007, pp. 21–21.

[5] B. Smith and G. Linden, "Two decades of recommender systems at amazon. com," *Ieee internet computing*, vol. 21, no. 3, pp. 12–18, 2017.

[6] C. A. Gomez-Uribe and N. Hunt, "The netflix recommender system: Algorithms, business value, and innovation," *ACM Transactions on Management Information Systems (TMIS)*, vol. 6, no. 4, pp. 1–19, 2015.

[7] S. T. Al-Otaibi and M. Ykhlef, "A survey of job recommender systems," *International Journal of the Physical Sciences*, vol. 7, no. 29, pp. 5127–5142, 2012.

[8] R. Rafter and B. Smyth, "Passive profiling from server logs in an online recruitment environment," in *Workshop on Intelligent Techniques for Web Personalization at the the 17th International Joint Conference on Artificial Intelligence, Seattle, Washington, USA, August, 2001*, 2001.

[9] F. Amato, R. Boselli, M. Cesarini, F. Mercorio, M. Mezzanzanica, V. Moscato, F. Persia, and A. Picariello, "Classification of web job advertisements: A case study," in *23rd Italian Symposium on Advanced Database Systems, SEBD 2015, Gaeta, Italy, June 14-17, 2015*, 2015, pp. 144–151.

[10] S. Bansal, A. Srivastava, and A. Arora, "Topic modeling driven content based jobs recommendation engine for recruitment industry," *Procedia computer science*, vol. 122, pp. 865–872, 2017.

[11] T. De Pessemier, K. Vanhecke, and L. Martens, "A scalable, high-performance algorithm for hybrid job recommendations," in *Proceedings of the Recommender Systems Challenge*, 2016, pp. 1–4.

[12] A. Bakarov, V. Yadrintsev, and I. Sochenkov, "Anomaly detection for short texts: Identifying whether your chatbot should switch from goal-oriented conversation to chit-chatting," in *International Conference on Digital Transformation and Global Society*. Springer, 2018, pp. 289–298.

[13] L. Shang, Z. Lu, and H. Li, "Neural responding machine for short-text conversation," *arXiv preprint arXiv:1503.02364*, 2015.

[14] O. Vinyals and Q. Le, "A neural conversational model," *arXiv preprint arXiv:1506.05869*, 2015.

[15] A. Sordoni, M. Galley, M. Auli, C. Brockett, Y. Ji, M. Mitchell, J.-Y. Nie, J. Gao, and B. Dolan, "A neural network approach to context-sensitive generation of conversational responses," *arXiv preprint arXiv:1506.06714*, 2015.

[16] J. Dodge, A. Gane, X. Zhang, A. Bordes, S. Chopra, A. Miller, A. Szlam, and J. Weston, "Evaluating Prerequisite Qualities for Learning End-to-End Dialog Systems," *arXiv:1511.06931 [cs]*, Nov. 2015. [Online]. Available: http://arxiv.org/abs/1511.06931

[17] A. Bordes, Y.-L. Boureau, and J. Weston, "Learning end-to-end goal-oriented dialog," *arXiv preprint arXiv:1605.07683*, 2016.

[18] J. Dodge, A. Gane, X. Zhang, A. Bordes, S. Chopra, A. Miller, A. Szlam, and J. Weston, "Evaluating prerequisite qualities for learning end-to-end dialog systems," *arXiv preprint arXiv:1511.06931*, 2015.

[19] T. Zhao and M. Eskenazi, "Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning," *arXiv:1606.02560 [cs]*, Jun 2016, arXiv: 1606.02560. [Online]. Available: http://arxiv.org/abs/1606.02560

[20] J. Williams, A. Raux, and M. Henderson, "The dialog state tracking challenge series: A review," *Dialogue & Discourse*, vol. 7, no. 3, pp. 4—-33, Apr 2016.

[21] B. Liu and I. Lane, "An End-to-End Trainable Neural Network Model with Belief Tracking for Task-Oriented Dialog," *Interspeech 2017*, pp. 2506–2510, Aug. 2017, arXiv: 1708.05956. [Online]. Available: http://arxiv.org/abs/1708.05956

[22] P. Wärnestål, "User evaluation of a conversational recommender system," in *Proceedings of the 4th IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, 2005, pp. 32–39.

[23] M. Jugovac and D. Jannach, "Interacting with recommenders: Overview and research directions," *ACM Trans. Interact. Intell. Syst.*, vol. 7, no. 3, pp. 10:1–10:46, Sep. 2017.

[24] D. Rafailidis, "The technological gap between virtual assistants and recommendation systems," *arXiv preprint arXiv:1901.00431*, 2018.

[25] F. Narducci, P. Basile, M. de Gemmis, P. Lops, and G. Semeraro, "An investigation on the user interaction modes of conversational recommender systems for the music domain," *User Modeling and User-Adapted Interaction*, pp. 1–34, 2019.

[26] A. Iovine, F. Narducci, and G. Semeraro, "Conversational recommender systems and natural language: A study through the converse framework," *Decision Support Systems*, p. 113250, 2020.

[27] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, 2003. [Online]. Available: http://jmlr.org/papers/v3/blei03a.html

[28] H. Steck, "Calibrated recommendations," in *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018, Vancouver, BC, Canada, October 2-7, 2018*, S. Pera, M. D. Ekstrand, X. Amatriain, and J. O'Donovan, Eds. ACM, 2018, pp. 154–162. [Online]. Available: https://doi.org/10.1145/3240323.3240372