Knowledge-Aware Recommendation System with Entity Representation and Generalization Extensions for Multilingual and Multidomain Applications

Andrea Parolin
Politecnico di Torino
Torino, Italia
s291462@studenti.polito.it

Jasmine Guglielmi
Politecnico di Torino
Torino, Italia
s303105@studenti.polito.it

Lorenzo Melchionna
Politecnico di Torino
Torino, Italia
s304651@studenti.polito.it

Abstract—**Recommendation systems are widely used to personalize content recommendations for users, and news recommendation is a particularly important task due to the time sensitivity of news and the need to provide users with relevant and engaging content. This work focuses on KRED, which is based on a document representation enriched with entity information and the relationship between entities. Starting from KRED, two extensions have been proposed in order to explore its generalization power: Adressa Exploration to adapt the model to another language (Norwegian) and Jobs Recommendation to adapt it to the domain of job offers. The work also proposes a new way to encode entities within the document. The code is available at: https://github.com/drewparo/kred**

I. Problem Statement

With the development of the internet and social networks, online news has increasingly become a source of information for users. Considering the abundance of articles to choose from, for each day, recommendation systems become essential in order to provide user the best possible experience and to be able to extend their time spent on the website [1]. The objective of recommendation systems is to analyze user needs and preferences, process a vast amount of information, and present the most appropriate option to the user. News RecSys have some peculiarity with respect to other domains:

- **Timeliness**: news per se is highly time-sensitive, about 90% of news expires within just two days. The concept of latency is critical in this domain and must take into account factor such as popularity, trends, and a high magnitude of incoming stories.
- **Relevant entities**: identify the key message conveyed by the article to acquire the real interest of the user by considering the position, frequency, and relationships of the entities.

Regarding the latter point, a knowledge graph contains comprehensive external information about entities and constitutes a promising research direction. *Liu Danyang et al. propose a Knowledge-Aware Representation Enhancement model for News Documents*, which enhances the document vector by fusing it with knowledge entities to produce a representation vector. The new document vector is used in various downstream tasks such as User-To-user, User-To-Item, Item-To-Item, category classification, and local news detection. To fully utilize knowledge information KRED exploits an entity representation layer that considers surrounding, a context embedding layer to encode contextual information such as position, category, and frequency, and an information distillation layer that aims to merge all entities into one fixed-length embedding vector using an attention mechanism. Due to the computation power required to process large datasets, a primary focus has been placed on User-To-Item RecSys however further experiments can be conducted on other applications. Proposed extensions on:

- **Adressa Exploration**: The study implements a pipeline for processing news articles in Norwegian, by examining associated challenges. The SmartMedia Adressa Dataset [2] is utilized as data source. The primary focus of the research involves Name Entity Disambiguation, achieved through the implementation of the ”ReFinED” [3] architecture and additionally to replace the TransE method [4] used in previous literature with a BERT-based approach.
- **Jobs Recommendation**: The study aims to transfer the news recommendation system to a different learning, leveraging the knowledge gained from the initial task to another related task. The authors developed a pipeline to process tech job offers from LinkedIn using a combination of `spaCy` and `nltk` for Named Entity Recognition, and to generate synthetic behaviors of hypothetical users whose information are extracted from the answers of Stack-Overflow survey.

II. Metodology

The use of a Knowledge-Aware Representation Enhancement model (KRED) as the main building block in this project is a key feature that sets it apart from other models. Unlike BERT, which does not consider knowledge entities in its representation of text, KRED takes into account these entities in order to provide a more comprehensive and complete understanding of the text being processed. The ability of KRED to incorporate knowledge entities allows it to perform better on recommendation tasks compared to other models like DKN [5], which only rely on a specific type of text

encoding. This is especially important in the context of news and jobs offers, as they often contain references to people, places, events, and other entities that can provide important context and information.
KRED aims to create a knowledge-enhanced document vector from any document representation that can be used in various applications, and this is achieved through 3 essential components:

a) Entity Representation Layer: This layer has the objective of producing an entity representation that also embeds the information of the relations with other entities. This is achieved by constructing a knowledge graph represented as a collection of entity-relation-entity triples, and using TransE to learn representations for each entity and relation. These representations along with the knowledge graph are kept fixed, and the knowledge graph attention network (KGAT) [6] is used to learn the final entity representation, including information about the neighbors of each entity.

b) Context Embedding Layer: Instead of feeding the entire document to the model, it is more efficient to extract entity information related to the document and embed it into its representation.
To get an overall representation of the context, 3 different context embedding features are used:

- Position encoding: where the entity appears.
- Frequency encoding: how frequent the entity appears.
- Category encoding: category to which the entity belong.

The final representation is obtained performing an element-wise addition between the output of the previous layer and the 3 context embedding vectors.

c) Information Distillation Layer: An attentive mechanism is used to highlight the most important entities within an article, where the initial document vector is the query and both key and value are entity representations from the previous layer. By summarizing the attention-weighted entity representations, concatenating the result with the original document vector, and feeding it to a fully connected feed-forward neural network, the final Knowledge-aware Document Vector (KDV) is obtained.

A. Extension - Adressa

Before starting with the processing of the dataset, it has been defined what are the essential features in the creation and adaptation: the main idea was to keep the structure of MIND as described in III-A0a , so as to modify the original architecture minimally.

1) News Articles: The dataset is structured as a click log, in which every click within the portal is recorded and saved. Each item clicked by a user is associated with a unique hash code to which an item corresponds. The hash code of all clicked articles is then extracted and mapped to the corresponding title, category, subcategory, link, and *description* which does not perfectly match the abstract as in MIND but expands the title and gives a brief summary of it. At this point it is necessary to extract the entities present within the title and description. Given the lack of pre-trained Entity Linking to run the task directly in Norwegian, it was necessary to translate the text into English to achieve a higher hit rate.

a) Machine Translation: SMaLL-100 [7] [7], a shallow multilingual machine translation model, was implemented to translate from Norwegian to English using limited resources. SMaLL-100's architecture consists of a 12-layer transformer encoder and a 3-layer decoder. The encoder makes use of the language codes, and the flat decoder has been designed to increase the speed of inference. The training was carried out using a balanced dataset that ensured uniform sampling across all language pairs, with an emphasis on maintaining performance for low-resource languages. The translated text is stored in a temporary variable and will only be used for Name Entity Disambiguation.

b) Entity Linking: ReFinED [3] is an entity linking (EL) system which links entity mentions in documents to their corresponding entities in Wikipedia or WikiData. It uses a Transformer model to perform mention detection, entity typing, and entity disambiguation for all mentions in a document in a single forward pass. The model is trained on a dataset generated using Wikipedia hyperlinks, which consists of over 150M entity mentions. The model uses entity descriptions and fine-grained entity types to perform linking.

As shown in Table I, the model has the ability to extract entities based on the context in which they are represented. A list of likely candidates and their scores is generated for each entity. The top candidate with the highest score is selected by normalizing the score across all candidates. The output is matched to a dictionary that matches the format (described in A) of a call to the Azure REST API [8].

c) Entity Vector: Originally, TransE is used to learn representations for each entity and relationship. Due to the computational power required to complete the method, it was decided to implement a new system that would be able to represent and describe these entities in the best possible way. The WikiData portal was queried using 'SPARQL Protocol and RDF Query Language' after extracting the list of entities (L=[Q6537, Q47094, ..., Q392897]) within the articles (title/description). The query was for the title and description of each entity. The title and the description were concatenated

```
Input  : Q392897
Output: Title: Polytechnic University of Turin
        Description: Technical university in Turin, Italy
```

and through the SentenceTransformer [9] library with the all-mpnet-base-v2 model, the corresponding embedding is obtained. A 768-dimensional dense vector space is used to represent each embedding. The same procedure is used for relationship encoding: given an entity, WikiData was queried to obtain the relationship between the entity itself and its neighbors, and the corresponding embedding was calculated from the description of the relationship.
It was not possible to load all entity and neighbour embeddings into memory due to the large space requirement of loading the entire knowledge graph into memory. By construction, these

are loaded into memory in the form of tensors. In order to proceed, a PCA was applied to reduce the size to 100. The schematic view can be seen in figure 2.

d) Document Vector: The SentenceTransformers [9] library was used to construct the document vector (DV) and encode within it the title and description features of each article. In particular, the nb-bert-base model [10] has been chosen. Starting from nb-bert-base, the model is trained on a machine translated version of the MNLI dataset [11]. Based on the large digital collection of the National Library of Norway nb-bert-base is a generic BERT-base model. The model maps title/abstract onto two 768-dimensional dense vector spaces.

B. Extension - Job Recommendation

As for recommending news, where a news could be recommended to a user, the same kind of recommending could also be done for jobs. This extension makes use of a transfer learning technique where, given a synthetic list of job histories for each user, a synthetic positive/negative sample of user clicks on job listings, and a collection of job listings, it could perform one of the following tasks: *user-to-item, item-to item, and category classification*. It is necessary to analyze step by step how to obtain each data set required for aforementioned tasks.

1) Job Offers: The dataset is structured as a collection of unique tech job offers, extracted from LinkedIn. Each offer is processed with a combination of nltk and spaCy pre-trained statistical models for text processing that include Named Entity Recognition. The main element of each job offer is the *description*, because it may contain important features that characterize one job from the others. Next, methods for extracting entities and summarizing each listing were explored.

a) Job advertisement abstract: It was decided to extract a summary, similar in style to a news abstract, as the job advertisements contain many unnecessary parts for the purpose of the recommendation system. Since the abstract of the articles is not extractive, the decision was to perform an abstractive text summarization.The architecture adopted is the one proposed by Google, that is, Pre-training with Extracted Gap-sentences for Abstractive Summarization (PEGASUS) [12].A hybrid method for creating the abstract was also tried for comparison: the most important sentences within the job advertisement were extracted and then through the *txtai* library an abstractive abstract was created.

b) Wikidata Id Extraction and Entity Linking: For the extraction of WikiData Ids featured in the job description that will identify the main peculiarities that represent a job, it is used a pipeline of library `nltk`. The model of `nltk` is trained on a dataset of annotated text, the CoNLL 2003 [13] dataset, which contains news articles with named entities annotated. Each job description is tokenized using `TreebankWordTokenizer` along with `PunktSentenceTokenizer`. On the obtained list, POS Tagging is applied in order to gather tagged tokens to be fed to the `nltk` recommended named entity chunker. This kind of classifier adds category labels such as person, organization, and GPE. This distinction of labels is needed to discard useless entities that do not "characterize" the job offer (i.e. Verbs). Once this list of entities has been obtained, the way in which corresponding WikiData IDs are retrieved is through the creation of a query. Each query is used to generate a request using the WikiData API, which is used to get the analogous WikiData ID. The list of unique WikiData Ids is produced in order to generate afterwards dictionary required for job recommendation. After the extraction, the name of the entity is used as a label for the corresponding entity and a small version of the spaCy English model suitable for Wikipedia entity identification, "`en_core_web_sm`", is adopted to extract the offset of the occurrence of the token of the entity and the type of an entity. The various types that could be assigned to an entity are: person, location, organization, date, time, quantity, event, product and work of art.

2) User Surveys: The dataset is structured as a collection of responses to the annual survey of unique StackOverflow users, regarding their skills and careers.
The goal of the study is to use the information provided by this survey of users to simulate their behavior, assuming that they are LinkedIn users who are looking for a job in the field of technology and basing the simulation on information previously extracted from job postings. A statistical approach is chosen to generate random clicks in this scenario to ensure that biases are not introduced into the simulation, to control the frequency and distribution of generated clicks, and to capture the unpredictable behavior of real users. This approach makes the simulation more representative of real-world behavior and ensures reliable and meaningful results.

a) WikiData Id Extraction and Entity Linking: See II-B1b.

b) Histories Generation: the method aims to generate a history of job offers clicked by the user in the past. It has been selected a list of feasible jobs for each user considering, feasible a job, for which it has extracted at least one entity that has been extracted also from user features.
A job is chosen from the list of feasible jobs with probability proportional to the number of common entities between user and each job. The probability of each job to be chosen is obtained by applying a softmax to the vector representing the number of common entities. At each iteration, the click generation process is stopped with probability p, which is set to an average number of clicks per user $\simeq 5$.

c) Behaviours: to generate each user's history with respect to job offers, it was decided to set the number of clicked announcements equal to 20. To proceed, a number of job ads not already in the user's history were selected by assigning a positive label if the ad is compatible with the user and a negative label otherwise.

III. Experiments

A. Data description

a) MIND: KRED trained the model on MIcrosoft News Dataset [14], a large-scale dataset for news recommendation research, contains behaviours and news article coming from

Microsoft News website. See Appendix A for more information on items, users, and entities

To proceed with the experiments the Demo version of the dataset is used to construct the baseline, which is already split into train, test, and validation.

b) Adressa: the dataset is offered by Adresseavisen, which is a Norwegian news portal. It is structured as a click log data set with approximately 20 million page visits from a Norwegian news portal as well as a sub-sample with 2.7 million clicks. The datasets are event-based and include anonymized users with their clicked news logs. In addition to click logs, the dataset contains some contextual information about users, such as geographic location, active time (time spent reading an article), session limits, etc.

The experiments were conducted using a one-week dataset, with the data divided according to the following specifications: the initial six days of history data were allocated for training, 20% of the data from the final day for validation purposes, and the remaining 80% for testing. As mentioned above: the objective is to obtain the same structure as MIND.

c) LinkedIn Tech Job Data [15]: Refer to Appendix B for further information. The only non-dropped columns for tasks are those containing industries, job function, description, post_id, post_url and title.

d) StackOverflow Developer Survey: For more information refer to dataset and parameters used for data selection refer to appendix C

B. Experimental design

The code was run in a virtual machine provided by Paperspace with a GPU between Nvidia P5000, RTX4000 or A4000. Unfortunately, the runtime of the freemium version is limited to 6 hours, and this, combined with the fact that the session had to be kept active, did not allow the model to be trained in a reasonable amount of time.
To evaluate the performance of KRED, different evaluation metrics are used depending on the task:

- User-To-Item: AUC score, NDCG@10
- Item-To-Item: NDCG@10, Hit Rate
- Category Classification: Accuracy, F1-score Macro
- Popularity Prediction: Accuracy, F1-score Macro
- Local News Detection: AUC score, Accuracy and F1-score macro

AUC (Area Under the Curve): especially in the case of user-to-item, this metric is useful because it is able to evaluate the model's ability to discriminate between positive and negative samples and is insensitive to a possible imbalance between the two classes, providing a comprehensive evaluation of a recommendation system's ability to rank positive items higher than negative items. **NDCG (Normalized Discounted Cumulative Gain)**: commonly used evaluation metric in information retrieval and RecSys. NDCG@10 measures the quality of a ranked list of 10 items or search results by comparing it to a perfect ranking. In user-to-item recommendation systems, it measures the quality of the list of recommended items by considering both the relevance of the recommended items to the user and their position in the list. In item-to-item recommendation systems, it measures the quality of the list of similar items to a given item by considering both the relevance of the items and their position. It provides an accurate evaluation of the system's performance because it considers both the relevance of the recommended items and their position in the list.

C. Execution times

Based on the tests conducted, the training and inference time for the first implementation is 50 to 60 minutes per epoch for the user2item task, while the second extension is 5 minutes per epoch.

D. Results

As explained KRED had the ability to perform multiple tasks simultaneously, but it has been chosen to focus on User-to-Item task, suggesting items that are most likely to pique users' interest analyzing their historical clicks.
The Hyper-Parameters used in the experiments, both for MIND and implementations, is showed in Table II. In the case of the first extension, i.e., implementation with Adressa, the score is very high: this is assumed to happen because negative sampling was set to 20. Obviously, the result of each prediction is a probability so it would be interesting to consider the behavior in the following days or weeks. Dealing with a dataset of articles the expiration of articles suggested is very high, an old article should be penalized over time. In the case of the second extension, the job recommendation system, the same problem occurs again and this could be caused by the same reason (high negative sampling), but also by the way the synthetic clicks were generated. More runs should be run trying to decrease or increase the negative labels.
Final results of this work are showed in Table III.

IV. Conclusions

Given the strong generalisation power of Knowledge-Aware Document Representation, the proposed extensions aims at extending its application domain dealing with different languages and document types.
To better understand the effectivenes of KRED, some considerations can be done for further improvements and comparison:

- Instead of extracting the sentence embedding from the concatenation of WikiData title and description for each entity and relation, it might be useful for the explained extensions, with more time and computational power, to use TransE to learn entity and relation embeddings as done in the original KRED implementation, to check the usefulness of this embedding technique to obtain more accurate predictions.
- For the Adressa extension, it might be interesting to vary the number of negative samples to measure their impact on the final results, and to use an entity linking model fine-tuned with Norwegian documents to reduce the noise introduced by machine translation.

- For the Job Recommendation extension, exploring different approaches for generating synthetic user behaviours, for instance, by exploring the cosine similarity between a representation of user and job. The job is assigned to the history or to the impression list, with a positive label if this similarity is higher then a fixed threshold. Of course, it would be interesting to test the model on a job dataset containing real user behaviours.

TABLE I: Expected output for *England won the FIFA World Cup in 1966*

Text	**WikiData Id**	**Wiki Title**	**Type**
England	Q47762	England national football team	ORG
FIFA World Cup	Q19317	FIFA World Cup	EVENT
1966	...	...	DATE

TABLE II: Hyper-Parameters set

Hyper-Parameters	**Values**
Epochs	5
Batch-Size	128
Optimizer	Adam
Learning Rate	$2\ 10^{-5}$
Weight Decay	10^{-6}

TABLE III: Results

Configuration	**AUC**	**NDCG@10**
MIND BERT	0.6914	0.2684
MIND demo SBERT *all-mpnet-base-v2*	0.6136	0.3285
Adressa One Week	0.7398	0.4878
Jobs recommendation with hybrid extraction	0.8972	0.6918
Jobs recommendation with PEGASUS extraction	0.8976	0.6962

References

[1] J. Lian, F. Zhang, X. Xie, and G. Sun, "Towards better representation learning for personalized news recommendation: a multi-channel deep fusion approach," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pp. 3805–3811, International Joint Conferences on Artificial Intelligence Organization, 7 2018.

[2] J. A. Gulla, L. Zhang, P. Liu, Ö. Özgöbek, and X. Su, "The adressa dataset for news recommendation," in *Proceedings of the international conference on web intelligence*, pp. 1042–1048, 2017.

[3] J. F. C. C. A. P. Tom Ayoola, Shubhi Tyagi, "ReFinED: An efficient zero-shot-capable approach to end-to-end entity linking," in *NAACL*, 2022.

[4] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," *Advances in neural information processing systems*, vol. 26, 2013.

[5] H. Wang, F. Zhang, X. Xie, and M. Guo, "Dkn: Deep knowledge-aware network for news recommendation," 2018.

[6] X. Wang, X. He, Y. Cao, M. Liu, and T.-S. Chua, "KGAT," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery &amp Data Mining*, ACM, jul 2019.

[7] A. Mohammadshahi, V. Nikoulina, A. Berard, C. Brun, J. Henderson, and L. Besacier, "SMaLL-100: Introducing shallow multilingual machine translation model for low-resource languages," in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, (Abu Dhabi, United Arab Emirates), pp. 8348–8359, Association for Computational Linguistics, Dec. 2022.

[8] Microsoft, "Microsoft text analytics api - entity linking," 2023.

[9] N. Reimers and I. Gurevych, "Making monolingual sentence embeddings multilingual using knowledge distillation," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 11 2020.

[10] "Operationalizing a national digital library: The case for a Norwegian transformer model,"

[11] A. Williams, N. Nangia, and S. Bowman, "A broad-coverage challenge corpus for sentence understanding through inference," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1112–1122, Association for Computational Linguistics, 2018.

[12] J. Zhang, Y. Zhao, M. Saleh, and P. J. Liu, "PEGASUS: pre-training with extracted gap-sentences for abstractive summarization," *CoRR*, vol. abs/1912.08777, 2019.

[13] E. F. Tjong Kim Sang and F. De Meulder, "Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition," in *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pp. 142–147, 2003.

[14] F. Wu, Y. Qiao, J.-H. Chen, C. Wu, T. Qi, J. Lian, D. Liu, X. Xie, J. Gao, W. Wu, and M. Zhou, "MIND: A large-scale dataset for news recommendation," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, (Online), pp. 3597–3606, Association for Computational Linguistics, July 2020.

[15] Mlawrence95, "Mlawrence95/linkedin-tech-job-data: A compilation of job posts and metadata scraped from various tech categories on linkedin."

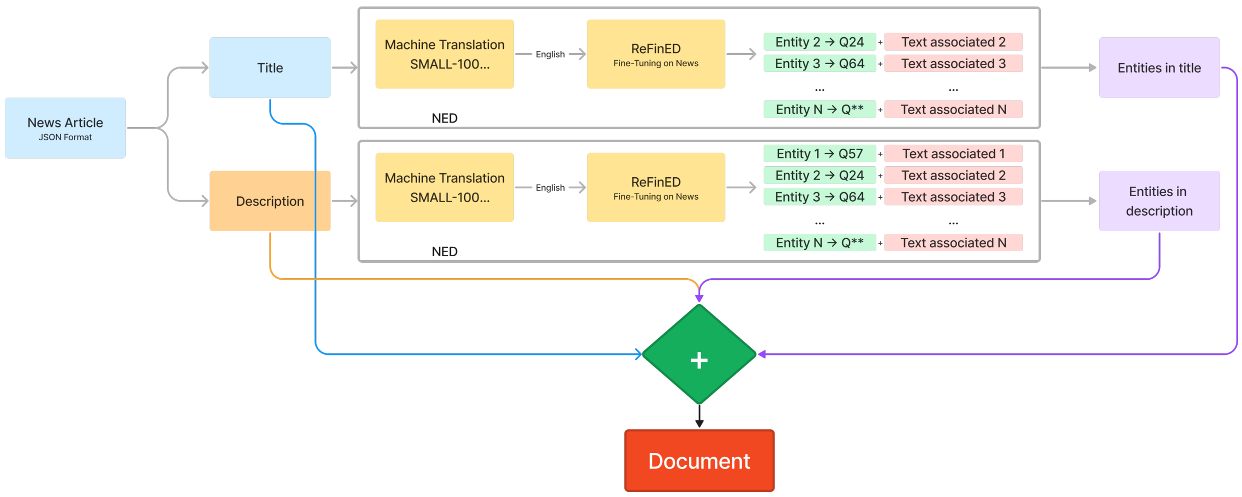

Fig. 1: Adressa Pipeline.

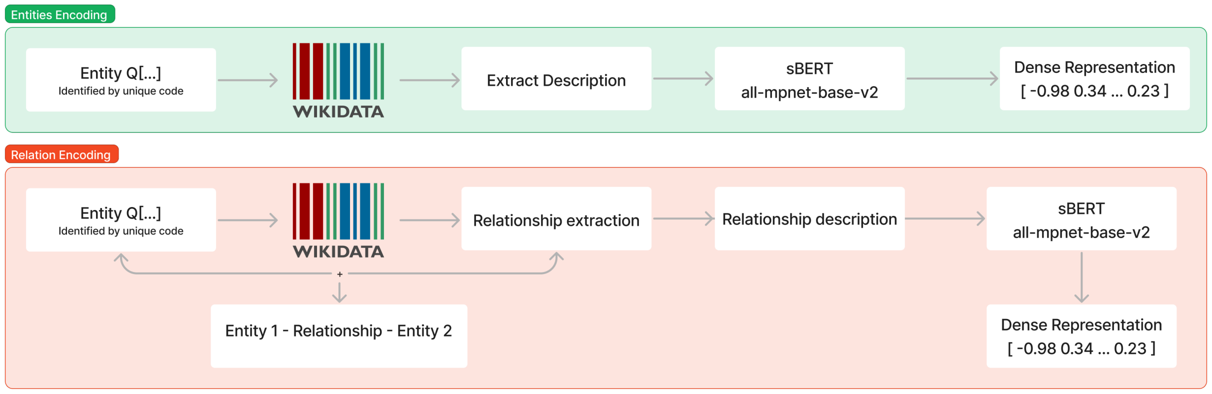

Fig. 2: Entity/Relationship Information Extraction.

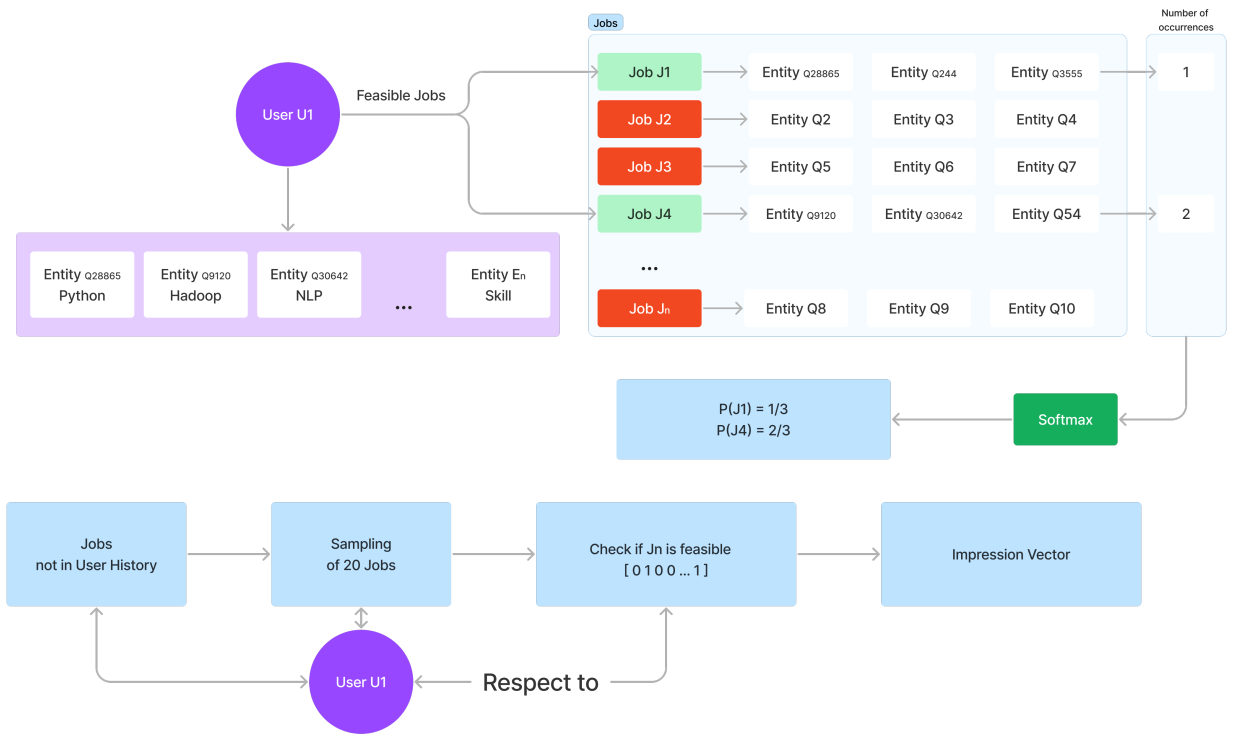

Fig. 3: History and Impression Generation.

Appendix A
Microsoft News Dataset

The dataset comprises approximately 160.000 english news articles and over 15 million impression logs created by 1 million users who clicked on at least five news articles within a six-week span from October 12th to November 22nd, 2019. Every news article contains rich textual content including:

- Article ID
- Category
- Sub-category
- Title
- Abstract
- Link
- Entities in the title
- Entities in the abstract

As cited in the original MIND paper the entities in the title and abstract were extracted using an *"internal NER and entity linking tool"*. It is safe to assume that Azure's Cognitive Services Entities Linking tool (more information [8]) is used since the entities within each article have the same format, namely:

- Label: The entity name in the Wikidata knowledge graph
- Type: The type of this entity in Wikidata
- WikidataId: The entity ID in Wikidata
- Confidence: The confidence of entity linking
- OccurrenceOffsets: The character-level entity offset in the text of title or abstract
- SurfaceForms: The raw entity names in the original text

Entities are encoded as 100-dimensional embeddings of the entities and relations learned from the subgraph (from WikiData Knowledge Graph) by TransE method.

Each impression log contains information about a user's click events, non-click events (negative sampling), and prior news click history. Each user is anonymized by hashing personal information into an anonymous ID.

- Impression ID: the ID of an impression.
- User ID : the anonymous ID of a user.
- Time: the impression time with format ”MM/DD/YYYY HH:MM:SS AM/PM”.
- History: The news click history (ID list of clicked news) of this user before this impression.
- Impressions: list of news displayed in this impression and user's click behaviors on them (1 for click and 0 for non-click).

To proceed with the experiments the Demo version of the dataset is used to construct the baseline, which is already split into train, test, and validation.

Appendix B
LinkedIn Tech Job Data

The dataset contains the following fields:

- Employment type: Full-time, Part-time, Contract...
- Industries: the field related to the job offer's working area
- Job function: the sub-sector of the company which is the provider of work
- Seniority Level: the length of time that an individual has worked in the job in question.
- Company: the name of the firm that offers the job
- Company ID: a unique ID related to a unique firm
- Context: JSON string like containing posting date, location, an HTML description of the job, experience requirements, minimum months of experience, the estimated salary (all information contained in the document after the following)
- Date: always `NULL`
- description: a depiction of all the information written in context but in a more human-readable way
- education: bachelor degree, master degree...
- location: where the job offer is located
- months_experience: the minimum number of months previously spent on the job
- post_id: a unique ID to identify each offer
- post_url: the link for the job
- sal_high: the maximum remuneration offered by the company for that position
- sal_low: the minimum remuneration offered by the company for that position
- Salary: the range of the job compensation
- Title: the actual job offered (e.g. Data scientist, Software Engineer, Machine Learning Engineer...)

The dataset is accessible https://github.com/drewparo/LinkedIn-Tech-Job-Data

Appendix C
Stack Overflow Developer Survey

Stack Overflow's annual developer survey is the largest survey of people coding around the world. This survey is based on a poll of 65,000 software developers from 186 different countries, but due to lack of computing time, the data set is truncated to 10,000 responses.
The survey is divided into 6 sections:

- Basic information
- Education, work, and career
- Technology and tech culture
- Stack Overflow usage plus Community
- Demographic information
- Final questions

The most important part is the one related to technology and tech culture because in order to make an appropriate job recommendation for each anonymous user, it is necessary to extract all the programming languages, database environments, platforms, web frameworks, libraries, tools, and collaboration tools required for a hypothetical job offer. For this reason, only responses that could reasonably be expected to contain important characteristics for the given purpose were extracted, such as developer type (professional or not), education level, operating system used, and databases, frameworks, libraries, and tools worked with, so all other columns were dropped before processing the dataset. The dataset is accessible https://info.stackoverflowsolutions.com/rs/719-EMH-566/images/stack-overflow-developer-survey-2020.zip