University Of Bari "Aldo Moro"

IT department

Degree in Computer Science

Artificial intelligence

Prompt-based recommendation system

Professor:

Giovanni Semeraro

Cataldo Musto

Marco Polignano

Students:

Roberto Barile 761030

Nunzia Lomonte 751040

Didactic Regulations 2022/2023

Index

Chapter 1

Abstract

In this work we investigate how to ground the whole recommendation task to natural language understanding. For such purpose we think that prompting is an interesting framework to explore because through prompting solving a problem means just to design a prompt, so finally the aim of this work is to design (or compute) prompts able to solve recommendation. The explored approaches are applied to the T5-language model in a sequence to sequence scenario. We also perform experimental evaluation on the DBbook dataset highlighting that at the current state recommendation can achieve through language modeling and no additional architecture performances comparable to collaborative baselines.

Chapter 2

Introduction

Pre-trained large language models (PLMs) have been extremely successful, and a variety of methods have been developed to apply these all-purpose models to many tasks. By carefully adjusting the PLMs extensive knowledge to task-specific data, they can be applied to numerous NLP tasks. Despite the success of fine-tuning PLMs, current research highlights a significant difficulty: the lack of objective forms between pre-training and fine-tuning. In order to achieve a task-specific objectives, we must synchronize all PLM parameters and task-specific minds during the fine-tuning process. This objective mismatch may prevent the transfer of PLM knowledge to subsequent tasks, according to Pan and Yang's [1] transfer learning theory. Prompting has been introduced to close the mentioned gap. Brown et al. [2] proved that prompt design, often known as "priming," is very effective at changing the behavior of a frozen GPT-3 model. Prompts frequently include a task description and/or a number of classic examples. It is interesting to go back to "freezing" pre-trained models, especially as the model size keeps growing. A single generative model can simultaneously serve many diverse purposes as opposed to needing a new copy of the model for each downstream task.
Prompt-based adaptation, however, has several significant disadvantages. The efficiency of a prompt is hampered by how much conditioning text can

fit into the model's input and the need for human engagement. As a result, the quality of tasks still falls well short of that of customized models. Finding acceptable templates and relevant label words manually to differentiate between classes, however, becomes difficult for projects with numerous classes. For many tasks, this takes a lot of effort and is unintuitive, but more crucially, the models are very sensitive to this context, and poorly designed contexts artificially degrade performance.

In this work we attempt to design and adapt prompts able to effectively represent, and consequently perform, the recommendation task. This research line is motivated by the fact that recommendation may be seen as a task that consists in learning the latent relationships between users and items, this kind of relationship may be effectively expressed in a natural language sentence containing the words "user", "item", maybe the specific entity that the item represents, e.g. "book" and words expressing the concept of relevance, e.g. "like", "interact". In addition since we focus on content-based recommendation which is an approach that very often make textual content available, it is easy to fit textual metadata about users and items in the chosen prompts. There have been relatively few studies that attempt to apply prompting approaches in the task of recommendation. (e.g. [3, 4]).

[5] suggest an orthogonal taxonomy to categorize current PLM-based recommender system according to their training methodologies and goals. They identify four main types of prompting:

- *Fixed-PLM+Prompt-tuning*: prompt-tuning is effective for especially few-shot recommendation tasks since it just requires refining a small number of parameters for the prompts and labels;
- *Fixed-prompt+PLM-tuning*: similar to the "pre-train, fine-tune" approach, fine-tunes PLM parameters while also using prompts with preset parameters to direct the recommendation task;

- *Tuning+Free-prompting*: this training method is known as zero-shot recommendations since it just relies on the input prompts and produces the outcomes of the recommendation or/and related subtasks without altering the PLMs' parameters;

- *Prompt+PLM-tuning*: the prompt relevant parameters and the model parameters make up the parameters in this setup. At the tuning phase, all parameters are optimized for certain recommendation jobs.

For our study, the *Fixed-prompt+PLM-tuning* technique was used on T5 models with manually constructed prompts. Then a *Prompt+PLM-tuning* technique was used adopting the prompt built by Autoprompt [6]. The figure 2.1 illustrates the process that summarizes the idea used in both experiments.

The rest of this documentation is organized as follows: the second chapter contains all definitions helpful to understand the techniques used; the third chapter collects research articles related to prompting and recommendation systems; the fourth chapter offers an extensive explanation of the two techniques used; the fifth chapter presents and addresses the results; the final chapter concludes with a brief overview and proposals for future developments.

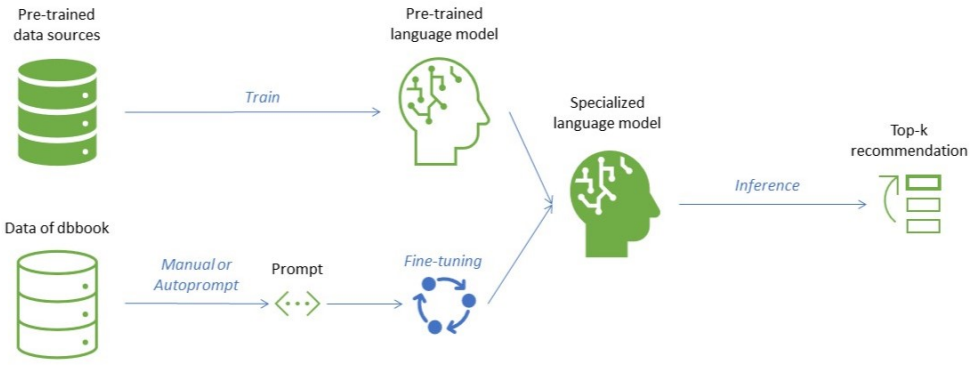

Figure 2.1: Illustrates the process used for this experimentation. In the diagram, the description of the method starts from the left and proceeds to the right. Initially we can observe two parallel ways: the first one was carried out previously by the creators of the language model; the second, performed by us, creates the prompt to be used to then fine-tune the PLM. The final result is a specialized language model from which it is then possible to make inferences and calculate the top-k recommendation.

Chapter 3

Preliminaries

A prompting function $f_{prompt}()$ is applied to change the input text x into a prompt

$$x_1 = f_{prompt}(x)$$

The majority of earlier work used a two-step procedure for this function:

1. apply a template, which is a textual string with two slots: an input slot $[X]$ for input x and an answer slot $[Z]$ for a preliminary generated answer text z that will subsequently be mapped into y;

2. insert the input text x into slot $[X]$.

We will refer to the first variety of prompt with a slot to fill in the middle of the text as a **cloze prompt**, and the second variety of prompt where the input text comes entirely before z as a **prefix prompt**. Flexible changes can be made to the amount of $[X]$ slots and $[Z]$ slots depending on the demands of the current duties [5].

We can identify at least four different approaches used with PLMs. These settings can be seen as lying on a spectrum of how much task-specific data they tend to rely on:

- **Fine-Tuning (FT)** has been the most common approach in recent years and involves updating the weights of a pre-trained model by

training on a supervised dataset specific to the desired task. The main advantage of fine-tuning is robust performance on many benchmarks. The main disadvantages are the need for a large new dataset for every task.

- **Few-Shot (FS)** refer to the setting where the model is given a few demonstrations of the task at inference time as conditioning, but no weight updates are allowed. Few-shot works by giving K examples of context and completion, and then one final example of context, with the model expected to provide the completion. We typically set K in the range of 10 to 100. The main advantages of few-shot are a major reduction in the need for task-specific data and reduced potential to learn an overly narrow distribution from a large but narrow fine-tuning dataset. The main disadvantage is that results from this method have so far been much worse than state-of-the-art fine-tuned models. Also, a small amount of task specific data is still needed.

- **One-Shot (1S)** is the same as few-shot except that only one demonstration is allowed, in addition to a natural language description of the task.

- **Zero-Shot (0S)** is the same as one-shot except that no demonstrations are allowed, and the model is only given a natural language instruction describing the task. This method supplies maximum convenience, potential for robustness, and avoidance of spurious correlations, but is also the most challenging setting. In some cases, it may even be difficult for humans to understand the format of the task without prior examples.

Before starting the experiment, an analysis of the state of the art was carried out looking for research focused on prompting for NLP tasks.
[7] introduces Pattern-Exploiting Training (PET), a semi-supervised train-

ing method that reformulates input examples into cloze-style phrases using natural language patterns. Three phases make up the PET process: first, a distinct PLM is trained on a tiny training set (T) for each pattern; second, the ensemble of all models is utilized to annotate a large dataset (D) with soft labels; and third, a conventional classifier is trained on the dataset with labels.

Given a task, e.g., sentiment analysis, [6] creates a prompt by combining the original task inputs (e.g., reviews) with a collection of trigger tokens according to a template. A variation of the gradient-based search method is used to learn the same set of trigger tokens, which are used for all inputs. The original task inputs, which were a collection of one or more sequences of tokens, are used to create the prompt by mapping them to a sequence of tokens using a template.

Prompt tuning with rules (PTR) [8] is for many-class classification tasks. Given a categorization task, the program first codifies prior task knowledge into rules, breaks the task down into smaller tasks, designs the necessary sub-prompts, and then uses the rules to create sub-prompts to complete the task. Prior knowledge encoding and employing rules to generate prompts is more effective and efficient with PTR.

[9] propose prompt tuning as a further simplification for adapting language models. They freeze the entire pre-trained model and only allow an additional k tuneable tokens per downstream task to be prepended to the input text. This "soft prompt" is trained end-to-end and can condense the signal from a full labeled dataset, allowing this method to outperform few-shot prompts and close the quality gap with model tuning. Soft prompts are learned through backpropagation and can be tuned to incorporate signals from any number of labeled examples.

As an additional simplicity for modifying language models, [10] suggests prompt adjusting. They only permit an additional k tuneable tokens for

downstream task and they freeze the entire pre-trained model. This strategy outperforms few-shot prompts and closes the quality gap with model adjustment because this "soft prompt" is trained end-to-end and can condense the signal from a full labeled dataset. Using backpropagation, soft prompts can be customized to include signals from any number of tagged samples.

[11] requests a textual description of classification labels from a language model that has already been trained. The classification job is reformulated as an image-text matching task, which maintains the text decoder's frozen state while updating the weights of the image encoder to map picture features in the same spatial domain as text features. This training uses a contrastive loss, such as text-to-image and image-to-text loss.

Name	Year	Prompt type	Prompt generation	Fine-tuning
Exploiting Cloze Questions for Few-Shot Text Classification and Natural Language Inference [7]	2021	discrete	semi-supervised	yes
AUTOPROMPT: Eliciting Knowledge from LMs with Automatically Generated Prompts [6]	2020	discrete	supervised	no
PTR: Prompt Tuning with Rules for Text Classification [8]	2022	discrete and continuous	automatic without ML	yes
The Power of Scale for Parameter-Efficient Prompt Tuning [9]	2021	continuous	supervised	no
Instance-aware Prompt Learning for Language Understanding and Generation [10]	2022	continuous	supervised	no
OrdinalCLIP: Learning Rank Prompts for Language-Guided Ordinal Regression [11]	2022	continuous	supervised	no

Table 3.1: The articles found in the state-of-the-art study are listed in this table in summary. The name of the author, the publication year, the prompt type (discrete or continuous), the method used to produce the prompt, and whether or not the fine-tuning of the model is performed.

Chapter 4

Related work

For this experiment, two papers about recommendation through prompting were of central interest. Both make recommendations using prompting techniques:

- [4] reformulates the session-based recommendation task using prompts. In language model recommender systems, each item is first mapped to its tokenized word description (such as the title of a movie), and then the item sequence is transformed into a text sequence to serve as the context using the prompt $f()$. Following the creation of the language model, they utilize it to estimate the probability distribution of the following item by aggregating the probabilities of each token in the item description, a process known as multi-token inference. It is similar to how prompting is typically used, which is to predict the next token in a series;

- the research [3] introduced P5, a framework for shared language modeling and natural language generation that integrates various recommendation tasks. They convert all raw data, such as the user-item interactions, user descriptions, item metadata, and user reviews, to the same format - input-target text pairs - by constructing a set of cus-

tomized prompts that span five recommendation task families. Then, they pre-train P5 in a whole language context to help it understand deeper semantics for a range of recommendation tasks. Their experiments demonstrate that P5 can use a variety of conventional methods to outperform or match performance on each of the five task families. Moreover, P5 exhibits generalization skills while carrying out zero-shot transfers to new products, new domains, and new tailored cues.

For our experimentation the paradigm of the first article was used. Instead, the manually created prompt was built by analyzing those used in the second research paper.

Chapter 5

Material and methods

For our project, two parallel experiments were conducted:

- the first uses a *Fixed-prompt+PLM tuning* technique where the prompt was created manually and used to do fine-tuning;
- the second uses a *Prompt+PLM tuning* technique where the prompt was built using Autoprompt [6] and then used for fine-tuning.

In both cases the T5[12] model was used and the starting dataset is DBbook but in the first case unstructured and in the second case structured. Both prompt used are prefix prompts.

5.0.1 Dataset: Dbbook

This dataset contains information about user preferences in the book field reported in the triples format: user_id, item_id, rank. Two different versions of the same dataset were used:

- *structured*: in this case, in addition to the ids and ranks, there is also information regarding: author, subjects, genre and genres that the user likes. Each piece of information has been collected from DBPedia and is represented by a string.

- *unstructured*: instead, for this dataset what was collected next is the genre information and genres that the user likes and a text describing the item. The description of the item is not complete since, to respect the input limits of T5, it has been truncated.

5.0.2 Fixed-prompt + PLM-tuning

This approach is inspired by LM-BFF [13], a work in which the authors explore the possibility of applying prompting to small, medium - sized pre-trained language models like BERT, roBERTa or similar; in particular this research line goes under the hood of prompt-based fine-tuning. This means that we translate the problem that we aim to solve into the prompting scenario as reported in the introduction, but then instead of aiming directly for correct predictions on unseen examples, we use the prompts to perform predictions on training examples and use the results to compute losses and update the weights through back-propagation.

In this work we extend this approach to T5 in its flan variant released by Google [1], the largest model size that we were able to fit on the available hardware is "base".

We think that prompt based fine-tuning is adequate for recommendation for the following reasons:

- language models like T5-base are not big enough to obtain satisfactory performance in a (few) zero-shot setup;
- we are not constrained in a (few) zero-shot setup because we have quite large datasets available for recommendation.

This approach consists in the manual design of a prompt that allows to map the recommendation task to a sequence to sequence task that can be easily performed by a T5 like language model. In this way the architecture of

[1] https://huggingface.co/docs/transformers/model$_doc/flan-t5$

the language model remains unchanged, we just need to reformulate recommendation problem, this is done by handcrafting a sentence that naturally describes in words what it means to classify an item as relevant or not given its content and a simple description of the user profile. In the figure 5.1 it is possible to observe a schema that summarizes the process performed.

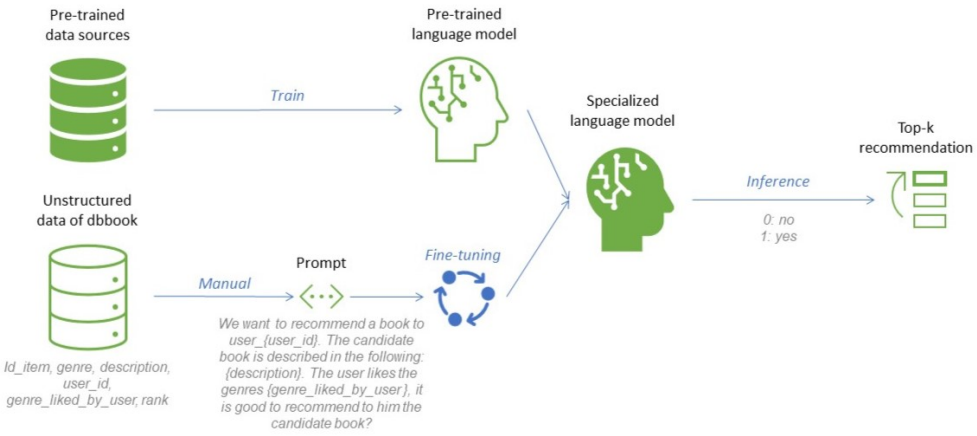

Figure 5.1: It represents the procedure used in the experiment called Fixed-prompt + PLM-tuning. In this case the prompt is created manually. Below some illustrations there are examples written in gray, useful to better understand what they represent.

In the following we report the manually designed prompts that we evaluated in our experiments:

1. **Input**: We want to recommend a book. The candidate book is described in the following: {description}. The user likes the genres {genre_liked_by_user}, it is good to recommend to him the candidate book?
   Target: Yes or No

2. **Input**: We want to recommend a book to user_{user_id}. The can-

didate book is described in the following: {description}. The user likes the genres {genre_liked_by_user}, it is good to recommend to him the candidate book?

Target: Yes or No

3. **Input**: We want to recommend a book to user_{user_id}. The candidate book item_{item_id} is described in the following: {description}. The user likes the genres {genre_liked_by_user}, it is good to recommend to him the candidate book?

   Target: Yes or No

4. **Input**: We want to recommend a book. The candidate book is described in the following: {description}. The user likes the genres {genre_liked_by_user}, it is good to recommend to him the candidate book?

   Target: Yes or No

5. **Input**: We will describe a book and the reading tastes of the reader, we want to know if such book is a good fit for the reader. {description}. User likes the book genres: {genre_liked_by_user}, will he appreciate the book?

   Target: Yes or No

6. **Input**: We will describe a book and the reading tastes of the reader user_{user_id}, we want to know if such book is a good fit for the reader. {description}. User likes the book genres: {genre_liked_by_user}, will he appreciate the book?

   Target: Yes or No

7. **Input**: We will describe the book item_{item_id} and the reading tastes of the reader user_{user_id}, we want to know if such book is a good fit for the reader. {description}. User likes the book gen-

res:{genre_liked_by_user}, will he appreciate the book?

Target: Yes or No

5.0.3 Prompt + PLM-tuning

The Autoprompt open-source project [6] has been a guide for the realization of this second experimentation. In this case, unlike the previous one, the prompt was built automatically. In the original Autoprompt documentation there is no experimentation done for the recommendation task. The figure 5.2 illustrates the process used which differs from the previous one only in the part of creating the prompt and in the examples.

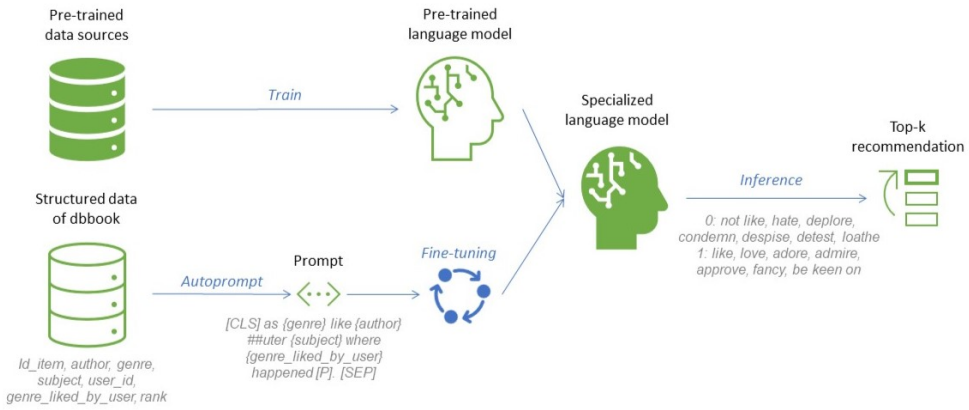

Figure 5.2: It represents the procedure used in the experiment called Prompt + PLM-tuning. In this case the prompt is created with Autoprompt. Below some illustrations there are examples written in gray, useful for better understanding what they represent.

Trials were carried out with both structured and unstructured datasets and we immediately found that the results obtained with structured datasets were more significant. The library to create the prompt needs a template from which to start, in our case the chosen template is: [CLS] [T] {genre} [T]

{author} [T] {subject} [T] {genre_liked_by_user} [T] [P]. [SEP]. Specifically, [CLS] e [SEP] are special BERT's characters that indicate the start and end of the string, [T] represent the tokens that the library must calculate, called trigger tokens, and [P] the target feature used to "fill-in-the-blank" by the language model, in our case it is the rank. The other chosen parameters were: number of iterations of 180, batch-size of 24 and label map equal to

- 0: not like, hate, deplore, condemn, despise, detest, loathe;
- 1: like, love, adore, admire, approve, fancy, be keen on.

The only flaw of the library is that it works only with the transformers BERT and roBERTa, since the versions of the internal libraries are prior to T5. In any case, the template was calculated on BERT but subsequently used to fine-tune on T5. The final prompt obtained is the following: **[CLS] as {genre} like {author}** ##**uter {subject} where {genre_liked_by_user} happened [P]. [SEP]**, with a dev metric equal to 0.65.

Chapter 6

Result and discussion

In this section we illustrate the experiments carried out for assessing the validity of our proposed solution. We specify the evaluation metrics and the experimental setting for the considered approaches. We conclude the section with discussing the obtained results.

The metrics used for evaluation are the standard classification metrics: Precision, Recall, F1, MAP and MAR, such metrics are adopted in their setting "@k", adopting $k \in \{3, 5\}$. We are dealing with a top-N recommendation task so this variants are more appropriate because a user is interested only to a limited number of recommendations. Since we are in a content-based recommendation scenario we need to specify which is the strategy adopted for candidate selection, that means the strategy to select, for each user, which items to classify, because for computational limitations it is prohibitive to perform, iteratively on each user, the prediction for each item in the catalog. In our work we decided to adopt only the candidates for which the ground truth label is known.

Both in the structured data and in the unstructured data we notice that we have good results in terms of Recall and MAR, while Precision values are unsatisfactory. This behaviour is highly related to the fact that for some user the number of candidate is less than the k value adopted in the "@k" setting

Model	Epochs	Prompt	Precision	Recall	F1	MAP	MAR
flan-t5-base	2	1	0.577	0.786	0.612	0.681	0.623
flan-t5-base	4	2	0.586	0.796	0.621	0.687	0.629
flan-t5-base	7	3	0.570	0.780	0.606	0.663	0.610
flan-t5-base	3	4	0.576	0.787	0.612	0.675	0.619
flan-t5-base	9	5	0.574	0.782	0.609	0.673	0.617
flan-t5-base	7	6	0.583	0.793	0.618	0.689	0.629

Table 6.1: Metrics@3 - Unstructured data

Model	Epochs	Prompt	Precision	Recall	F1	MAP	MAR
flan-t5-base	2	1	0.433	0.906	0.541	0.560	0.733
flan-t5-base	4	2	0.435	0.908	0.544	0.604	0.737
flan-t5-base	7	3	0.431	0.902	0.540	0.588	0.723
flan-t5-base	3	4	0.433	0.905	0.541	0.596	0.730
flan-t5-base	9	5	0.432	0.903	0.540	0.595	0.728
flan-t5-base	7	6	0.434	0.906	0.542	0.604	0.735

Table 6.2: Metrics@5 - Unstructured data

Model	Epochs	Prompt	Precision	Recall	F1	MAP	MAR
flan-t5-base	20	manual	0.473	0.932	0.592	0.658	0.825
flan-t5-base	9	automatic	0.474	0.934	0.594	0.658	0.827

Table 6.3: Metrics@3 - Structured data

Model	Epochs	Prompt	Precision	Recall	F1	MAP	MAR
flan-t5-base	20	manual	0.306	0.966	0.440	0.532	0.881
flan-t5-base	9	automatic	0.305	0.964	0.439	0.532	0.881

Table 6.4: Metrics@5 - Structured data

of the metric; this leads to a recommendation list which length is lower than k, the missing recommendations impact on the precision value. For the same motivations metrics computed at 3 are slightly higher than metrics computed

at 5. There is no difference between using a manual prompt and one obtained from Autoprompt, the final performances are the same. We may also draw useful conclusions analyzing how the choice of the prompt impact on the performances. Looking at the prompts list presented in the previous chapter we mainly notice two categories:

- **prompts 1, 2 and 3**: they share the same sentence structure, the difference between the three prompts is the inclusion of IDs:
  - in prompt 1 we do not have any ID
  - prompt 2 contains the ID of the user
  - prompt 3 contains the ID of the item and the ID of the user
- **prompts 3, 4 and 5**: this prompts share another sentence structure, slightly more complex than the one in the previous group. About the IDs we used the same approach as the previous group

The motivation behind the inclusion of IDs is to push, during the finetuning, the LM towards the concept of personalization, recognizing a mapping between IDs and words. In the results computed @3 we notice that for the first group of prompts the best results are obtained including only the user ID. While for the second group the best performances are in the setting with both IDs. The same situation is observed in the metrics computed @5.

Chapter 7

Conclusion

Our work attempts to carry out a first approach to recommendation based on prompting of language models. The results obtained are not very high especially for the precision, but we believe that new experiments could lead to higher performances. In order to improve performances it might be useful to experiment with different types of manual templates used on the structured dataset. As future developments it might be interesting to use the specialized model to create embeddings to be used in a neural recommender system like AMAR [14]. The embeddings generated by this model could be more semantically rich since the weights based on the input dataset have been fine-tuned, or they may be an additional input to AMAR, containing the necessary semantics to undersatnd users, items and their relation expressed through language. Thinking on this line, in which we add inputs to AMAR, prompting can be exploited to finetune a language model on a task different from recommendation, but somehow related. An example may be to use a prompt that presents to the language model the content of an item or a user and a sentence that explains that we are providing as input a textual content. If we mask some tokens of the content and we add to the prompt a sentence that asks to the language model to fill such masks, we fine tune the model in a way that makes it able extract knowledge about items and users, the aim of

this proposal is to involve in the embedding the latent knowledge about items and user that may be present in the LM, in particular after finetuning. An example prompt for this strategy is : "The item item_id has the following description: {description}, we need to fill the missing information".

References

[1] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[2] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[3] S. Geng, S. Liu, Z. Fu, Y. Ge, and Y. Zhang, "Recommendation as language processing (rlp): A unified pretrain, personalized prompt predict paradigm (p5)," in *Proceedings of the 16th ACM Conference on Recommender Systems*, ser. RecSys '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 299–315. [Online]. Available: https://doi.org/10.1145/3523227.3546767

[4] Y. Zhang, H. Ding, Z. Shui, Y. Ma, J. Zou, A. Deoras, and H. Wang, "Language models as recommender systems: Evaluations and limitations," 2021.

[5] P. Liu, L. Zhang, and J. A. Gulla, "Pre-train, prompt and recommendation: A comprehensive survey of language modelling paradigm adaptations in recommender systems," *arXiv preprint arXiv:2302.03735*, 2023.

[6] T. Shin, Y. Razeghi, R. L. Logan IV, E. Wallace, and S. Singh, "Autoprompt: Eliciting knowledge from language models with automatically generated prompts," *arXiv preprint arXiv:2010.15980*, 2020.

[7] T. Schick and H. Schütze, "Exploiting cloze questions for few shot text classification and natural language inference," *arXiv preprint arXiv:2001.07676*, 2020.

[8] X. Han, W. Zhao, N. Ding, Z. Liu, and M. Sun, "Ptr: Prompt tuning with rules for text classification," *AI Open*, vol. 3, pp. 182–192, 2022.

[9] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," *arXiv preprint arXiv:2104.08691*, 2021.

[10] F. Jin, J. Lu, J. Zhang, and C. Zong, "Instance-aware prompt learning for language understanding and generation," *arXiv preprint arXiv:2201.07126*, 2022.

[11] W. Li, X. Huang, Z. Zhu, Y. Tang, X. Li, J. Zhou, and J. Lu, "Ordinalclip: Learning rank prompts for language-guided ordinal regression," *arXiv preprint arXiv:2206.02338*, 2022.

[12] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 5485–5551, 2020.

[13] T. Gao, A. Fisch, and D. Chen, "Making pre-trained language models better few-shot learners," 2020. [Online]. Available: https://arxiv.org/abs/2012.15723

[14] C. Musto, C. Greco, A. Suglia, and G. Semeraro, "Ask me any rating: A content-based recommender system based on recurrent neural networks." in *IIR*, 2016.