Sorbonne Université — ISIR — EDITE de Paris

THÈSE

défendue par

AGNÈS MUSTAR

en vue de l'obtention du grade de Docteur

MODELING USER-MACHINE INTERACTIONS DURING THE INFORMATION RETRIEVAL PROCESS

Devant le jury composé de

Dr Jean-Pierre Chevallet	Rapporteur
Université Pierre Mendès France Grenoble II	
Dr Karen Pinel-Sauvagnat	Rapporteuse
Université Paul Sabatier Toulouse III	
Pr Alexandre Allauzen	Examinateur
Université Paris Dauphine-PSL	
Pr Catherine Pelachaud	Examinatrice
Sorbonne Université, CNRS	
Pr Sylvain Lamprier	Co-directeur
Université d'Angers	
Dr Benjamin Piwowarski	Co-directeur
Sorbonne Université, CNRS	

For Michka.

Remerciements

Ma thèse, à l'image de la recherche, repose sur la présence de nombreuses personnes. Seule ou sans le soutien et l'implication de chacune d'entre elle ce travail n'aurait pas pu voir le jour.

Je remercie d'abord mes directeurs de thèse Benjamin et Sylvain qui ont accepté de m'encadrer, merci de m'avoir proposé de nombreuses directions de recherche, d'avoir partagé avec moi votre expertise et expérience. Sans vos conseils, je me serais perdue dans les méandres de la recherche. Votre capacité à discerner les pistes prometteuses a grandement contribué à l'avancement de cette thèse.

Je suis également reconnaissante envers les membres du jury d'avoir lu et évalué mon travail Catherine Pelachaud, Karen Pinel-Sauvagnat, Jean-Pierre Chevallet et Alexandre Allauzen. Vos commentaires constructifs et vos précieuses suggestions ont enrichi cette thèse.

Merci à l'ensemble des doctorant.e.s du MLIA. Je n'ai pas eu l'occasion de travailler avec vous mais votre présence bienveillante et joyeuse a été un soutien moral d'une importance inestimable. Chaque partie de babyfoot gagnée a contribué à mon bien être ! Je te remercie évidement en particulier Marie, tu as été mon plus grand soutien et une amie en or. Je suis également reconnaissante envers les ancien.ne.s qui m'ont guidé Jean-Yves, Etienne, Clément, Jérémie, et Perrine, celles et ceux de ma génération Yuan, Tristan, Corentin, Frosso, Pierre, et Yannis, et les plus fraîchement arrivé.e.s Tanguy, Raphael, Armand et Lise.

Merci aussi à tous les permanent.e.s et en particulier à Christophe et Laure, qui m'ont aidée jusqu'au jour de la soutenance. Merci de prendre soin de nous pendant l'aventure du doctorat.

Enfin je remercie ceux qui comptent le plus, ma famille. Merci pour votre amour inconditionnel et vos encouragements qui m'ont permis de me surpasser. Merci à mes grands-mères qui sont des modèles, à mes parents, mes frères et ma belle-sœur et évidement à Romane, mon soleil.

Pour finir, je te remercie ma chère petite sœur, Pauline. Merci de m'avoir aidée à préparer toutes mes interventions orales et d'avoir organisé un mémorable pot de thèse ! Mais surtout merci d'avoir été

un soutien inestimable, notamment dans les moments qui ont été les plus difficiles, tu es mon ancre.

Abstract

While today's search engines work well for simple queries, there are situations where search results are not satisfactory. To cope with such situations, user-machine interactions have increased significantly since the early days of retrieval systems. The exchanges between users and retrieval systems during search sessions may contain information that is critical to the success of the information search.

Meanwhile, the Transformer-based architectures relying on the attention mechanism have led to great improvements in several NLP tasks, such as summarization or translation. The architecture was soon applied to other domains, including information retrieval. Several retrieval models have benefited from this architecture's ability to focus on document and query terms to estimate their relationships.

However, the majority of these works have focused on ad hoc retrieval. The goal of this thesis is to study user modeling and user-machine interactions with transformer-based models. The contributions of this thesis can be divided into two parts, those related to user modeling and those related to interactive systems. In the former, I analyze existing user models, and in particular the text generation process of transformer-based models for query suggestions (Mustar, Lamprier, and Piwowarski, 2020; Mustar, Lamprier, and Piwowarski, 2021). In the latter, I present a new user/machine interaction framework based on the studied user models (Mustar, Lamprier, and Piwowarski, 2022).

Contents

List of Figures

List of Tables

Chapter 1: Introduction

To retrieve information and access knowledge in large data collections such as Internet, people need to have a certain number of expertise and skills. Digital literacy (Reddy, Sharma, and Chaudhary, 2020) refers to the competence to use technologies in order to find, evaluate and communicate information. This competence depends on various user-specific factors, among which social and cultural background, access to collections, age, level of education, and individual technical ability. The ability to use these technologies and to obtain relevant information also depends on technological advances (computer, internet connection, software).

Access to knowledge does not only depend on the user literacy and IT tools. Another essential feature, less often mentioned and yet playing a crucial role, is the algorithm endowing users to access information, namely search engines. Their functioning is often opaque, at least from the users' point of view. Search engines are black boxes, as explained in the sociology of sciences (Pasquale, 2015), i. e. users do not need to understand their inner workings but nevertheless, the former determine what information users can access. Users are biased by the search system they are using, just as they are influenced by their level of literacy and their equipment, without necessarily being aware of it.

How to rank documents in Information Retrieval (IR) systems is the result of human choices that influence the information available to users. An important example dates back to 1998, when the founders of Google, Larry Page, and Sergey Brin, presented an algorithm, PageRank (Brin and Page, 1998) whose principle is based on the notion of citation, which comes from the scientific literature. It consists in evaluating the importance of a web page. The PageRank value of a page is calculated according to the number of pages pointing to it and the PageRank values of these pages. The higher the value, the higher the document is ranked in the result page. With PageRank, Google puts more emphasis on websites that are frequently linked to.

PageRank's way of evaluating pages has risks, for example, a page that is often quoted, because it is shocking or because it contains false

information, has a high value. There are other ways to evaluate the importance of a web page, for example with TrustRank (Gyöngyi, Garcia-Molina, and Pedersen, 2004), a score also taken into account in the Google algorithm, which evaluates the relevance of pages from a set of documents known as legitimate. The TrustRank algorithm relies on authority, while PageRank relies on visibility.

The importance given to TrustRank and PageRank is one of the many design choices that influence how documents are selected, sorted, and classified. The fact that search engines are black boxes makes this choice invisible. But in reality, algorithms either respond to explicit rules, or seek to optimize a function posed manually, or fit data labeled by humans. Thus, it is a human point of view that influences the documents put forward.

Given the crucial role of search engines in accessing information, it is essential to reduce these biases and to improve the relevance of their results. To this end, we can try to give back more control to users through interactions, for example, by asking them about their preferences. Interaction is a way for users to override the search engine biases discussed above, and to incorporate their personal expectations. They also make the choice of document ranking more transparent and less black-boxed for the users. In the next section, I present how users and search engines can interact together, and how these interactions can be leveraged to improve search results.

1.1 Information Retrieval

In this section, I briefly review the history of IR, trying to address the place given to user-computer interactions and user modeling. I point out the importance of IR by going back to its omnipresence in today's world and its multiple applications. I then expose the current challenges of IR.

1.1.1 The first IR researcher

In 300 BC, the Greek poet Callimachus had to organize the 500,000 papyri of the great Library of Alexandria so that readers could easily find them. He organized papyrus scrolls into six genres, which he then sorted by author in alphabetical order. For each genre, he created an index – called a Pinake – with the name of the documents as well as essential information to represent them, such as the author and a summary (Phillips, 2010). Historians now consider his work to be the first library catalog that existed (Harman et al., 2019). While Callimachus' work has now almost completely disappeared, the method he used is still relevant today in some places, like bookstores or libraries. In fact, as soon as collections with large quantities of documents existed, estab-

lishing systems to find the information sought in these collections was necessary. Obviously, Pinakes are far from the complex Information Retrieval systems of today, however, they testify the need of having methods to access information as early as 300 BC.

After Callimachus, different systems allowing a person to search for a document in a large collection were created, as for example directories or registers of voters of a city. But it is with the development of computers and then of the Internet that IR research has really advanced.

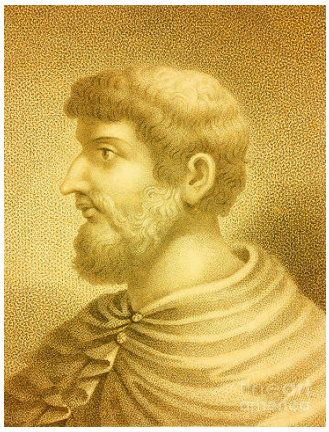


Figure 1.1: Callimachus, the designer of the first library catalog

1.1.2 The IR boom

In the 60s and 70s, we witness the emergence of communication networks, a key component of different technologies that accelerated the development of IR (Harman et al., 2019).

In the early 1960s, an MIT project aimed to allow computers, not located in the same geographical space and technically heterogeneous, to communicate and exchange information, instead of just being calculators. The technique is based on a communication method called packet switching used by the telephone industry. It has the advantage of allowing messages split into different packets to circulate through different routes, which makes them less vulnerable to failure. The project, called Arpanet, was set up in 1969 and was used by academics who progressively increased its capacities. With the support of American industrials, operating networks with various types of architectures was then made possible. Two protocols, that were then combined, played a decisive role in the circulation of packets: in 1973 TCP (Transmission Control Protocol) and in 1978 IP (Internet Protocol) (Ceruzzi, 2003; Abbate, 1999). To encourage the adoption of these protocols, DARPA funded their integration into the Unix operating system, which was then sold at a low cost to universities. Thus, the Internet was rapidly deployed in all American universities, and in the early 1990s, the Internet arrived in Europe, Asia, and Australia.

As the Internet evolved and gradually spread throughout the world, computers also evolved. In 1969, Bell Labs computer scientists Ken Thompson and Dennis Ritchie developed the first version of an operating system including an assembler, an editor, and a shell. This system, called Unix, is distinguished by its flexibility and simplicity compared to other existing systems, which partly explains its success. In addition, as mentioned briefly, in 1975, Unix and its source code were distributed to

universities in exchange for a very low-cost license, which popularized it in the academic world. At the end of the 70s, personal computers were being developed. One will speak later about the 1977 trinity: a series of computers, from the companies Commodore, Apple, and Tandy, which had an important public success.

In parallel with the development of networks and computers, the first works on data management appeared. In 1965, Charles Bachman designed the Ansi/Sparc architecture which is still used today. The MARC format – MAchine-Readable Cataloging – a format for the exchange of computerized bibliographic data was created in 1968. Then in 1970, Edgar F. Codd of IBM San Jose Research Lab laid down the principles behind relational databases. Last but not least, many search engines were created (JumpStation, Mosaic, Yahoo!, Infoseek, and Altavista the most popular in the late 1990s...), although they were quickly surpassed by Google created in 1998. Thus, in the 90s, with the increase in computer power and storage, the development of personal computers, and of course, the creation of search engines, applications of IR models took off. This development is noticeable, for example, by the expansion of the SIGIR conference (Harman et al., 2019). It is not the purpose of this thesis to review all of these works. However, to contextualize my thesis, in the next section, I present current IR research problems, as well as those that are emerging.

1.1.3 IR research: current issues

In this section, I present a non-exhaustive set of subfields of IR, and their current research issues.

Data While information retrieval is a relatively unknown field of research for the public, it has led to the development of some very popular technologies. First, of course, there are search engines, which are used on a daily basis by almost anyone who has access to the internet. IR methods are needed when a person uses a search bar, whether it is the one from a website, or a device, or an application. These techniques are essential when a large amount of data is available. Beyond the quantity of data, its very nature raises new research problems. Consequently, IR subdomains attempting to cater for specific domains with their own data have been developed. They deal with various types of data such as geographical data (Purves and Jones, 2011), legal texts (Van Opijnen and Santos, 2017), or chemical structures (Cooke, 2011), or with types of data that do not have the structure of a standard text document: image (Datta et al., 2008), speech (Singhal and Pereira, 1999), video (Gabeur et al., 2020) or 3D retrieval (Tangelder and Veltkamp, 2004). The emergence of new types of data leads to the rapid development of new domains.

Usages Other IR areas tackle the problem of improving user experience. Indeed, it is sometimes tedious for users to perform their search in the form of keyword queries, or they find it laborious to look for the answer to a question in a long text, or they would prefer to talk directly to the search device to answer their need for information. This corresponds to specific information access tasks, respectively question answering (Kolomiyets and Moens, 2011), automatic summarization (Scialom et al., 2019a), or conversational and interactive search (Radlinski and Craswell, 2017). The latter is now a very active research area and includes new search interfaces such as chatbots (Adamopoulou and Moussiades, 2020) or voice assistants. These assistants have become widespread in recent years with their commercial applications, i. e. Alexa, Siri, or Google voice assistant (Kiesel et al., 2018).

This shows the interest to further research on this constantly evolving field where new hardware technologies (devices, computers, phones, tablets, etc...) and new emerging types of data (social networks, 3d, virtual reality, NFT, etc...) offer new research opportunities.

Although in some cases IR systems are satisfactory, a part of the search sessions does not lead to relevant results (Carmel and Yom-Tov, 2010). Thus, even in the case of searches with more traditional data, text queries, and text documents, searching is not a solved problem. Solving these information needs requires looking at the nature of the search process itself, proposing new ways to search or to return information, i. e. to interact with users. The proposals for improvement concern all stages of the process, from the moment users perform their search, to the algorithm which ranks the documents, the processing of the results, as well as the interface.

Expression of need The first step that can be improved in the search process concerns the way users express their needs. The most common way for users to communicate their information needs to search engines is to submit a textual query using keywords or natural language. To allow users to add constraints on query words, some retrieval models (Muhammad, 2017) allow queries to be expressed as boolean strings, i. e. queries with logical operators (and, or, not, ...). However, formulating a query to express a need can be tricky for users without expertise, thus many conversational search frameworks have emerged (Adamopoulou and Moussiades, 2020; Radlinski and Craswell, 2017).

Instead of asking users to express their needs in words, retrieval methods require other types of data from the requester such as an image, a tag, or a document. These methods are respectively query-by-document (Abolghasemi, Verberne, and Azzopardi, 2022), query-by-tag (Wang et al., 2011) and query-by-image (Datcu and Seidel, 1999).

More generally, these methods are referred to as query-by-example (QBE) methods. They are particularly well suited in cases where users fail to express their needs explicitly. The newest methods allow users to submit more complex examples. For example, for audio, there are methods for finding a song by singing it (query-by-humming) (Alfaro-Paredes, Alfaro-Carrasco, and Ugarte, 2021) or playing its tune with an instrument (query-by-vocal percussion) (Delgado et al., 2021). For text, query-by-webpage (Geng, Chuai, and Jin, 2022) allows to find websites that are similar in terms of content and web design. The latter makes use of the fact that sources of the same type generally have similar web designs (e. g. government or academic sites).

Recently, Mysore et al. (2021) introduced the faceted query-by-example task: in addition to providing a document, users specify the aspect of the document that interests them. This task allows to work on more advanced systems than those trained for QBE, but also to users to get finer-grained control on the results.

Ranking After the user's initial query, arises the question of which documents to return. On this point, actual search systems work relatively well for needs that are considered simple. For example, to find the name of the capital of France, a query is enough. But the needs and the data are not always so trivial. Strohman et al. (2005) present a search engine that deals with complex searches, i. e. searches with queries requiring the combination of several documents to be answered or searches with strong constraints on the documents such as the proximity of terms, or the structure of the document.

A query can also be enriched by its context. Personalized ranking (Abri, Abri, and Cetin, 2020) consider users' data (age, gender, GPS location, ...) to refine the search results, while contextual ranking proposes to improve the relevance of search results by taking into account users' history such as past clicks and queries (Qu et al., 2020). I detail these domains in the next Section 2.3.

Finally, the Transformer models, which have enabled great advances in NLP, are now being used to design the new generation of IR systems thanks to their great semantic and cross-attention capabilities. We describe this in more detail in the next chapter, Section 2.2.

Filtering After the selection of the documents by a first retrieval algorithm, a processing can be done. Indeed, the information in the database is not always controlled, and information filtering attempts to exclude specific documents, such as fake news (Zhou and Zafarani, 2020) or spams (Aswani et al., 2021), from the results. Moreover, to eliminate redundancy and lack of diversity, that actually still appears in search results of current search engines, new metrics that take results diversity into account are introduced (An, Huang, and Wang, 2020).

Interface Finally, various works (Hearst, 2009; Negi et al., 2020) focus on the last stage of the search process: displaying the returned documents. They aim at improving the interface that presents the results. Sekaran et al. (2020) adapt the interface to the needs of users, with a presentation of results that summarize documents' concepts relevant to users' queries. Interfaces can also be adapted to the target audience. (Allen et al., 2021) design a search engine for children, showing the importance of interactive tools and icons. (Aqle, Khowaja, and Al-Thani, 2020)'s interface is adapted for visually impaired users by reducing the effort and the time needed to find the relevant information within documents.

To sum up, the information retrieval process can be improved at each stage of the process: from the beginning via the way users express their needs, to the ranking and filtering of the results, and finally via the displayed interface. We can also consider a more global approach of the problem. Rather than considering each step independently, the whole process or part of it can be modeled. In this way, the dependencies between the stages can be studied. For example, by modeling the users expressing their needs and the model returning the results, one can set the objective of optimizing both parts jointly.

1.2 User-Machine Interactions

Exchanges between users and retrieval systems are therefore essential at each of the stages described above. I now detail the explicit and implicit interactions between users and search engines, and expose the different problems for which there are no satisfactory solutions, and how the modeling of users could allow to improve retrieval systems.

1.2.1 Mutual influence between users and search engines

As long as they are unsatisfied, users seek to modify search results (Huang and Efthimiadis, 2009). They change the terms of their initial query to refine the results according to their needs. Users also influence search results in a more indirect way since engines' personalization mechanisms take users' behavior and personal data into account.

Some search engines, such as Google, highlight results related to actual events (Campos et al., 2014). They also take into account users' GPS coordinates to prioritize documents close to their geographic area (Tabarcea, Gali, and Fränti, 2017). Time and location play a role in the results presented, as well as the type of device used. For instance, mobile information retrieval (Tsai et al., 2010) focuses on information

searches from smartphones. All of the above define the search environment, which plays a key role on users' behavior and expectations, and on search engines' results.

While the environment, search engines, and users determine the search results, search engines influence users in their search behavior. First, in an obvious way, users conform to the displayed interface. For example, they have learned to express their need in the form of keywords with search engines, while they discuss when it comes to interact with a chatbot.

More generally, user behavior can be analyzed through the concept of *nudge*, crafted by the Nobel Prize-winning economist Richard Thaler and the philosopher and lawyer Cass Sunstein (Karlsen and Andersen, 2019). Their work shows that incentive is more efficient than coercion in achieving a desired behavior. Influencing users' behavior through a *nudge* is more efficient than coercing them. For example, most search engines are equipped with an automatic query suggestion tool: after users have typed the first few query characters, search engines suggest a list of queries. This list of suggestions influences users' subsequent search path.

To conclude, one cannot think about search systems without considering their users, and one cannot study these users without considering the algorithms and techniques that make search engines. The purpose of this thesis is precisely to study both users and machines when the former seek to acquire information. I investigate user models and interactive search process to improve the quality of search sessions.

1.2.2 Leveraging interactions

Today access to knowledge is partly done through the Internet and search engines. While for simple queries, search engines are fulfilling well their role today, there are situations in which the search results are not satisfactory. Let us take a look at the basic principles of search engine algorithms, before discussing their limits through two types of queries, for which search engines fail.

Initially, retrieval systems relied on statistics of query word occurrence in the documents. This method considers that the higher the number of occurrences, the higher the document rank. The limit of these models is that users must use exactly the same words as those present in relevant documents, this issue is called *vocabulary mismatch*.

Although search engines have significantly evolved today, query formulation still relies on the matching of similar concepts in the query and in the document, without always caring about the global meaning of the query. Consequently, this does not always satisfy the information need. For example, a user searching "What animal doesn't eat lettuce?" is returned exclusively pages about animals with lettuce in their diet.

This is because documents about animals mention what they eat but not what they don't eat.

As a second example of the limitations of search engines, consider the cases where the information need must be broken down into several queries. In those cases, search engines generally fail. For example, Google finds relevant results for the queries "movie hero in love AI?" and "color movie her", i. e. pages that discuss the movie *Her* in the first case, and pages that describe the preponderance of red in the movie *Her* in the second. On the other hand, when searching for "color movie hero in love AI?", the search engine only sends documents that evoke the movie and not the color red. In this example, to reach their goal, users must make two queries. For such information needs, users will probably try new queries to get the information they are looking for.

More generally, Carmel and Yom-Tov (2010) have established a taxonomy of situations that can cause a search system to fail, and thus lead users to reiterate the process. These situations are grouped into two categories, those where systems fail to identify and cover all aspects of the topic and those in which it incorrectly analyzes the meaning of the query. In the first category, systems emphasize an irrelevant aspect or miss a relevant one. This is the case of the query "color movie hero in love AI?" for which Google miss the aspect "color". The second category includes failures to identify relationships between terms, proximity relationships, or expansion of a general term (e.g., extending the word "Europe" to a specific country such as "France"). The query "what animal doesn't eat lettuce?" falls into this category: the considered search engine, Google, does not take negation into account. Interactions with the machine could allow users to show, in an indirect and non-explicit way, that they are still unsatisfied.

The level of interaction between the user and the machine has evolved greatly since the early days of search engines. In the beginning, users were quite passive and had very few interactions with the search engine. The only possible interactions were with the queries and the returned search results. With years, and the growing interest of researchers and computer scientists in information retrieval, tools to increase the quality and quantity of these interactions have been developed. This is for example the case of the query suggestion tool or the automatic query completion tool.

Inherently, the system transmits information – the returned documents – to users, but users transmit also a certain number of feedback or information in an indirect way to the system. For instance, reformulating a query constitutes an interaction whereby users modify their query in the hope of obtaining more relevant results. In fact, quite early in the history of IR, leveraging terms added or deleted between two consecutive queries have been considered (Bruza and Dennis, 1997).

Other users' actions can be taken into consideration to improve the system. For example, the links clicked during a search session help to understand the trajectory of the search session (Mei, Zhou, and Church, 2008), the documents clicked and ignored on the results page are a form of feedback (Ahmad, Chang, and Wang, 2018; Ahmad, Chang, and Wang, 2019), and even the computer mouse movement (Diaz et al., 2013), or the eyes' movement on the screen (eye-tracking process) (Cutrell and Guan, 2007) provide information.

1.2.3 Benefits of user modeling

As discussed in the previous section, machines, and users hence exchange information with each other, and this information can be decisive in improving the search process. We now show how user models can help to leverage those interactions.

During information search sessions, users start with an initial query, then perform a set of actions: they study the returned documents, click on some of them, might browse the internet further, formulate new queries, and reiterate until they are satisfied or abandon their search. Modeling users consists in building a model that predicts all or a part of these actions. Brusilovsky and Tasso (2004) justify the necessity of user modeling in this way: "the information retrieval system needs to follow over time the way the user understands and formulates her information needs". User modeling could allow the system to adapt to the specific needs of users. This modeling can be enriched by integrating other data specific to users such as their geographical position (Tabarcea, Gali, and Fränti, 2017), the device used (Tsai et al., 2010), their previous searches (Sordoni et al., 2015), the set of admissible languages, etc. Improving user modeling has many potential applications we detail below.

Enhance interactions Search engines and users are interacting during the search process. On one hand, users send queries and on the other hand, search engines answer with a list of documents. Still, it is users who initiate the interactions, through query reformulation and navigation in the results page, which can be used as implicit feedback. Although today search engines do not only rank documents, and have several strategies to improve the search experience such as response highlighting, query suggestions, query completion, or results aggregation. Richer interactions between the two parties could largely improve search experiences. The prediction of users' next queries, which is a form of modeling, is used by query suggestions and autocompletion tools (Sordoni et al., 2015; Dehghani et al., 2017; Mustar, Lamprier, and Piwowarski, 2021). These tools are particularly important in the case of complex searches to guide users and save time. One way to make interactions more relevant is to correctly model users during their search

session. It has been done, for example, by predicting users' intent and then asking them clarifying questions about their intent (Dhole, 2020).

Improve interfaces The classic search engine interface with results presented as a document list is challenged. For instance, search engines now propose vertical search results which consist of aggregating results of different types (text, image, video, news...) in an ergonomic interface that allows users to find the information they are looking for more quickly(Zhou et al., 2013).

Interfaces displayed to users can also be personalized thanks to user models. These are called Adaptive Web Applications. A representative work in this area is Lohmann, Kaltz, and Ziegler (2006) who propose an approach to take users' information into account in the way their graphical interface is displayed. However, they show that a failed adaptation confuses the user, which limits the adaptation proposals and the tasks for which the interface can be customized.

Training with simulated users Interactive IR systems are nowadays parametric systems, they can be trained with the simulation of users' decisions such as their queries, clicks, and overall satisfaction. Training such algorithms requires thousands of interactions, which is impossible to obtain. Even evaluating them on a sufficient number of real users is very expensive.

The question of training models with simulated users is becoming more and more important with the emergence of heavy neural network architectures for text processing as Transformer (Vaswani et al., 2017). It could be used, for example, to limit the number of interactions so that users reach their goals as quickly as possible.

Improve metrics IR methods that aim to improve some metrics such as accuracy, or recall, are called system-centric. Conversely, research that uses real users to evaluate their performance is called user-centric. A gap has been found between these two types of measures since there is not always a correlation between metrics and user satisfaction (Liu et al., 2019a). This gap shows that automatic measures are not always satisfactory. Consequently, training or evaluating a model by optimizing these same metrics is necessarily not very fruitful. In contrast, a perfect user model could predict if a user is satisfied or not. A simulated user could be used during training to define a reward, or during model evaluation at a cost lower than with expensive human evaluation (Dupret and Piwowarski, 2013).

In this thesis, I work on user modeling in order to improve user-machine interactions, which I consider the most promising angle of attack to improve the classical search process. User modeling in IR goes beyond search engines. It could be applied to every situation in

which humans and machines interact. For instance, in the context of task-solving modelling, such as automatic translation or accounting, software programs could benefit from these methods by anticipating user behavior or explicitly asking for clarification. Given the intensive use of machines to solve tasks today, it is interesting to model our interactions with them.

1.3 Towards user modeling in IR

User models can be leveraged to improve user search experience, as discussed in the previous section. We now discuss how the idea of modeling the user has slowly emerged in IR.

From the early days, researchers already understood that taking into account the users' behavior and/or their characteristics could improve search results. In this section, I describe the first IR methods which, although they aim at various objectives (system evaluation, results improvement, or user categorization) have in common to consider users to improve IR systems results through better user models.

1.3.1 Cranfield experiments

In the 1960s, Cyril W. Cleverdon, a British librarian and computer scientist, proposed a series of experiments to evaluate an indexing system (Cleverdon, 1960). While research systems have changed dramatically since the 1960s, the Cranfield evaluation paradigm has been followed for decades. The size of the Cranfield collections no longer corresponds to the amount of data in current problems, however, the principles of these experiments are still used in a number of current evaluation campaigns.

The first Cranfield experiments are based on a test collection that consists of a set of queries and documents with their associated set of relevance judgments, indicating which documents are relevant or non-relevant to which queries. Then various metrics – such as precision, recall, and NDCG – are used to compare the evaluated search systems. Cleverdon initially proposed an experiment in which a single document is deemed relevant to a query. But this assumption has been criticized. Indeed, it led to problems: a user might be happy to be presented with several sources of knowledge (Harman, 2010).

For this reason, Cleverdon proposed a second evaluation (Cleverdon, 1967; Cleverdon, Mills, and Keen, 1966) in which a party judges if the set of returned documents is relevant to the query. Moreover, in this second evaluation, queries were formulated in natural language, thus allowing to judge the correlation between real users' searches and their satisfaction given the documents, more than on their ability to generate keyword queries. In these second Cranfield experiments, the

evaluation is conducted more from the users' perspective. We notice how very quickly in the IR history users are placed at the forefront of the evaluation process.

Of course, Cranfield evaluations are limited, since they only consider one type of users and are restricted to sessions of one query. However, they are still very useful to advance ad-hoc search systems (Harman, 2010).

1.3.2 Premises of user modeling

At the end of the 70s, the term "user modeling" first appeared (Rich, 1979). User modeling can rely on two types of information: the user categorization using stereotypes (Rich, 1979; Brajnik, Guida, and Tasso, 1987) or a summary of their past interactions with the machine (Gershoum, 1981; Belkin, 1984). Stereotypes are characteristics that can be inferred about a person from initial information about them. In the 80s and 90s, the increasing evolution of the work on user modeling shows that it is an important concern. The aim of this section is not to make an exhaustive review of these works, but to describe some of them in order to understand their positioning and the conclusions that serve as a basis for interactive information retrieval.

In 1981, Gershman introduced the Automatic Yellow Page Assistant (Gershoum, 1981) whose objective is to return to users useful addresses for car repair based on their query about the problem with their vehicle. The program can interact with users to ask for more information. The method, based on rules and a knowledge graph, is done in several steps. In particular, one of them has the objective of finding the user's goal. Although the system responds to a specific need and is only functional in a specific domain, it constitutes one of the first "user model" during a search and a proposal for interaction between the user and the machine.

Belkin studies how a search process takes place in a more general way (Belkin, 1984). For this purpose, he analyzes telephone records of conversations between a user and an intermediary person working in an online IR service, who issues the queries for the user. He concludes with three essential criteria for a good search system: a model of the user's problem, and interactions, and that the user model and the search engine model can cooperate to evolve mutually. These criteria are still valid today.

Another work explores user modeling through the aggregation of users via stereotypes: the assumed characteristics of a person (see Section 1.3.2). (Rich, 1979) proposes to use both users' behavior and stereotypes to improve the search system. He presents Grundy, a system that plays the role of a librarian who makes book recommendations. To do this, he first asks users to describe themselves with adjectives and then asks about their literary tastes. Brajnik, Guida, and Tasso (1990) show

that the information about users is used to better respond to their needs, and that the interactions help to refine the recommendations.

In this section, I showed how the IR field evolved quickly to put the user at the center of this field. I then discussed how user interaction has been leveraged in early IR models. The presented works are based on systems restricted to very specific domains and needs, where user modeling is implicit. In the next section, I review the evolution of the IR paradigm which tried to better match the search process to users' needs.

1.3.3 Questioning the ad-hoc retrieval and the Cranfield paradigm

The Cranfield evaluation is based on a simplified conception of the information retrieval process. It considers that the information need is associated with a query and a set of relevant documents. In reality, this IR paradigm, called ad-hoc retrieval, is not sufficient to meet all information needs. It works well for particular cases such as simple searches, very specific tasks, or very long queries, but often users face more complex search situations, in which they perform several queries, and for which the search is done in several steps. In these cases, Liu et al. (2019a) have shown that ad-hoc evaluations following the Cranfield model are insufficient to measure users' satisfaction.

The Cranfield evaluation is said to be system-oriented because it is positioned from the perspective of the search system and not the user one. Therefore, more user-oriented systems, evaluations, and collections have been proposed such as the Okapi project (Robertson et al., 1994) or the MEDLARS tests (Borlund, 2009).

Donna Harman, an IR researcher who received the 1999 Tony Kent Strix award for outstanding contributions to the field of IR, initiated the TREC evaluation initiative (Harman, 1993) in 1992. The competition is co-sponsored by the National Institute of Standards and Technology (NIST) and the U.S. Department of Defense, but tracks are organized by the participant research groups. A track consists, in general, of documents, research problems, and the needed relevance judgments. This competition had a considerable impact on information retrieval by standing out mainly for its large text collections, its strong participation of the academy and industry, as well as the variety of its tracks, such as Complex Answer Retrieval (CAR), Precision Medicine Track, and Cross-Language Track.

Hal Varian the Chief Economist at Google wrote that "The TREC data revitalized research on information retrieval. Having a standard, widely available, and carefully constructed set of data laid the groundwork for

further innovation in this field"[1]. Starting from the first TREC track in 1992, there is a feedback track, an idea based on the Rocchio algorithm (cf 1.3.4.1). In this track, participants have to submit three types of results: without using user feedback, with one feedback iteration, and with any number of feedback iterations. Participants are free to use the feedback in their system to reformulate the next query automatically or to re-rank future results.

Stephen Robertson, an IR researcher, known in part for his famous BM25 model, said: "In fact, the idea has extended into theories and models; the notion that documents may be judged for relevance to the need becomes not just a mechanism for evaluating systems, but a basic concept in design" (*Information Science in Transition* 2009). Next, I present a set of works in which users are considered "by design".

1.3.4 User models

Users can be modeled either from their characteristics or from their behavior. The latter, based on users' past behavior, is one of the topics of my thesis. In this section, I review the evolution of user modeling. I do not make an exhaustive list of user modeling works, but to draw up a quick overview of the tracks pursued in the 2000s before the arrival of the Deep Neural Networks (DNN) on which most research efforts have since then been concentrated.

1.3.4.1 Rocchio algorithm

The Rocchio algorithm (Rocchio, 1971) proposes to improve a query based on the relevant and non-relevant documents associated with it. The new query is considered better than the initial one because, in a document and query space, it is closer to relevant documents and further away from non-relevant documents. An explicit way to get the set of relevant and non-relevant documents is to directly ask the user feedback, i. e. the so-called relevance feedback.

Asking users for feedback on the returned documents is probably the most direct user-machine interaction. Firstly because it is explicit feedback, and secondly because this feedback is directly targeted at the relevance of the results. But in reality, the Rocchio algorithm was mostly used without explicitly asking users for feedback, but using what is called pseudo-relevance feedback, which consists in using indirect signals to label documents as relevant (Robertson and Jones, 1976).

1.3.4.2 Explicit user models

Current user modeling methods are mainly based on deep learning – I present them in the state of the art in chapter 2. But before the arrival

1 https://googleblog.blogspot.com/2008/03/why-data-matters.html

of deep learning, various methods were used to obtain user models. I quickly review these works to illustrate the diversity of the methods on which researchers have relied for user modeling based on short or long-term behavior. All these works are not from the IR community, however, they could also benefit retrieval tasks.

First, Bayesian models have been frequently used to model users. A Bayesian network is a probabilistic model where probability relationships between variables are encoded by a directed acyclic graph. Bayesian networks have been used to predict user future actions (Kuenzer et al., 2001), to model users' search intent as a mixture of users' latent search interests and their results preferences (clicks) (Wang et al., 2014), and to investigate user models (Tedesco et al., 2006). Besides Bayesian models, decision trees have also been used in the context of works on users. They can be used for classification. For example, Beck et al. (2003) use classification trees to predict if users click on a specific word. Finally, neural networks (not deep) have also been used in this context. Bidel et al. (2003) use neural networks to classify user navigation paths.

Although various methods have been proposed for modeling users, today DNNs are the most common algorithms used for this task. I focus more on these methods in the next chapter, by first presenting the prerequisites in NLP and RL to understand these approaches.

1.4 Organization and contributions of the thesis

Organization The objective of my thesis is to improve user-machine interactions during a search process. The thesis is structured around four chapters. First, I discuss the evolution of user-centeredness in search systems. The latter has always been present, although it was initially very simplistic. I then present user model methods that came before the boom fostered neural networks and deep learning. In the second chapter, I first outline the NLP and RL background on which the state of the art and the proposed models lie. I analyze how the three domains (NLP, RL, IR) have evolved first separately, and then were able to share their methods. After that, I analyze the recent works on interactive IR and user modeling based on NLP or RL models. The last two chapters present my contributions, which I present next.

Contributions I now briefly describe my thesis's main contributions. I contribute to the study of RNN and transformers for user modeling during a search session, through the task of next query prediction (Chapter 3). Investigating long, complex, or noisy sessions, I compare the robustness of various models for this task. Finally, I analyze how

transformers generate text in the context of a user model. The results of this work have been published in the CIRCLE conference (Mustar, Lamprier, and Piwowarski, 2020). A more detailed version has been published in the TOIS journal (Mustar, Lamprier, and Piwowarski, 2021).

Then, in Chapter 4, I present a new user/machine interaction framework. It is based on a user model that is derived from my previous analysis of user behavior. More specifically, the search process is viewed as a series of moves in a latent semantic information need space, where the system agents drive the users toward relevant documents. This system is trained using self-supervision and RL techniques. The work on this framework has been published at the ICTIR conference (Mustar, Lamprier, and Piwowarski, 2022).

Chapter 2: Background and State of the Art

This Chapter has two objectives. First, to introduce the essential concepts of text representation in IR and the Reinforcement Learning (RL) background. Then, in the remaining sections, to give an overview of the state of the art in user-machine interaction (within IR).

2.1 Word, sentence and document representations

For many fields involving texts – such as RI, NLP, or text mining – representing texts in a latent space is essential. The objective is to represent texts in a vector space where similar text vectors are close to each other according to a distance (e.g. the Euclidean distance or the inner product). The produced representations can have different uses, whether to find similar texts or to calculate *distances* between texts. In this section, I expose methods from IR and NLP researchers. Although these methods were distant for a while, they have finally converged with neural networks.

2.1.1 Bag-of-words

Gerard Salton, known as "the father of Information Retrieval", presented in 1975 a vector space model for IR (Salton, Wong, and Yang, 1975). This space model is used for automatic indexing, where the stored entities are the documents and the keys are the queries. The method relies on the fact that similar documents and query vectors tend to be close in the vector space. Its novelty, at the time, was to use word frequencies in a corpus of text as a clue for semantic similarity.

For that, it relies on the term-document matrix based on a collection. The matrix columns represent documents as bag-of-words (BOW) – a simple way of representing text as a set of words, without taking into account word order or grammar.

Considering a vocabulary V, that is a set of $|V|$ terms e.g. $V = \{anaconda, ant, antelope, \dots\}$. Each document D_i is represented by a $|V|$-dimensional vector $D_i = (d_{i1}, d_{i2}, \dots, d_{i|V|})$ where d_{ij} is the weight of the j^{th} term. This value is zero if the term is not in the document, and otherwise, multiple methods (such as TF-IDF) can be used to compute it. The representations from the vector space model allow to compute the similarity between a query vector and a document vector as their inner product (or cosine distance), and so to retrieve documents from a query. The retrieval model was implemented on the SMART system (System for the Mechanical Analysis and Retrieval of Text) (Salton, 1971).

Vector space models were the theoretical foundation of the pioneering SMART search engine, and have been used by multiple NLP researchers (Turney and Pantel, 2010). However, they rely on simplistic BOW representations of text, which limits the semantic information that can be embedded in the vectors. In this space, documents with similar content but different terms can have completely different representations, while documents with common terms but different contexts can be close. The semantics of the texts are poorly represented because the order of the words and the dependencies between words are not taken into account. Another limitation is that the representation is at document level and not at lower levels (such as word or sentence level). For the latter, BOW representations are quite deficient.

2.1.2 Word Embeddings

IR researchers based early document representation methods on the assumption that documents with common words have similar content and should therefore have close embeddings. However, these embeddings lack semantics. Two steps can be taken to obtain word embeddings with a better sense of their meaning. First, the idea of term co-occurrences is introduced. Then, instead of considering whole documents, a more restricted context such as a paragraph, a chapter, or a predefined number of words is considered. Thus, word-context matrices are considered instead of a word-document matrix.

Co-occurrence et matrix factorization Dumais (1990) presents a method to obtain word embeddings that are dense vectors with a size smaller than the vocabulary. She relies on the frequency of occurrence of words in the same window of a predefined size. The matrix X of dimension $|V| \times |V|$ is called a window or a document-based co-occurrence

matrix. To reduce the size of this large matrix, Dumais (1990) uses a matrix factorization method. More precisely, a singular value decomposition (SVD) reduction, where $X = U\Sigma V^\top$. The first k columns of U correspond to the coordinates of each word in a k-dimensional vector space. The embedding matrix is of dimension $|V| \times k$. The use of truncated SVD for word similarity is called Latent Semantic Analysis (LSA), when used for document similarity it is called Latent Semantic Indexing (LSI). Unlike BOW representations, LSA and LSI methods have the advantage of giving representations of size k, where k is a fixed parameter that does not depend on the size of the collection. However, this method is computationally expensive: SVD-based methods do not scale well to large matrices, and to represent new words, the whole process has to be redone from scratch.

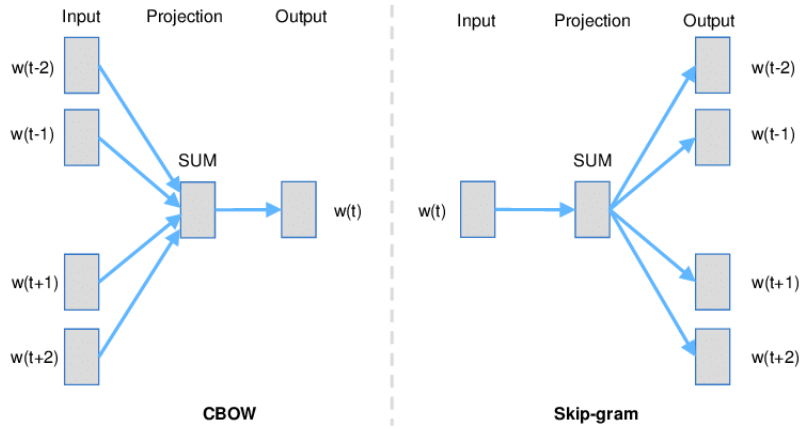
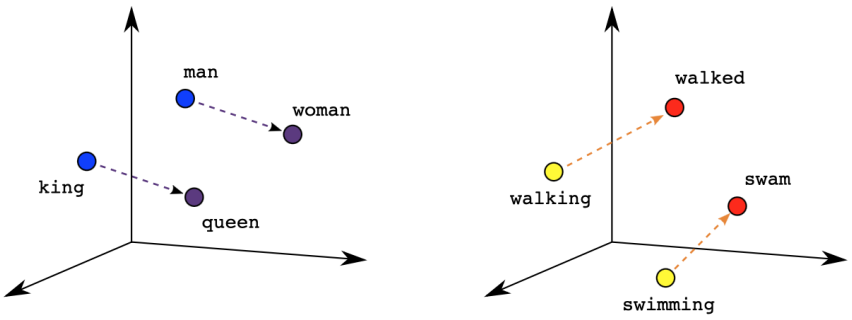


Figure 2.1: Word2vec models CBOW and Skip-Gram. Illustration from Landthaler et al. (2017).

Word2vec Models using matrix factorization are based on BOW-type representations of words. Word order, dependencies, or relationships are not taken into account. NLP researchers have been working on statistical methods that allow for better encoding of word meaning. In 2013, two models – CBOW and Skip-gram – were introduced, by a team of NLP researchers at Google, under the name Word2vec (Mikolov, Yih, and Zweig, 2013). For CBOW, the idea is to learn the probability of a word based on its context. Let us take as an example the sentence "`ignorance of the law is no excuse`", given the input "`ignorance of the ? is no excuse`" the goal is to predict "`law`". On the contrary, for Skip-gram it is to learn the context from a word, i. e. from "`law`" predict the whole sentence "`ignorance of the law is no excuse`". In both cases, a neural network is learned with backpropagation (Rumelhart, Hinton, and Williams, 1986). At the end of the training, the hidden layers of the network are used as word embeddings.

As LSI, Word2vec is able to capture the semantics of words. More interestingly, Mikolov, Yih, and Zweig (2013) have shown that Word2vec learns the relationships between words. For example, by performing the operation `king - man + woman` on the embeddings of the concerned words, the closest word to the result is `queen`. This phenomenon is illustrated Figure 2.2. This is also true for many relations such as singular/plural, present/past, or capital/country.



Male-Female Verb Tense

Figure 2.2: Examples of semantic relations captured with Word2vec. Source: https://developers.google.com/machine-learning/crash-course/embeddings/translating-to-a-lower-dimensional-space?hl=fr

Unlike LSI, which does not provide document representations, Doc2vec (Le and Mikolov, 2014), an extension of Word2vec, allows to learn sentence, paragraph or document representations of a fixed size. Doc2vec refers to two models, respectively Distributed Memory Model of Paragraph Vectors (PV-DM) and Distributed Bag-of-Words version of Paragraph Vector (PV-DBOW), which are equivalent to CBOW and Skip-gram. To extend Word2vec to documents, Le and Mikolov (2014) add a document-specific representation vector to the model's input. Once the model is trained, this vector holds a representation of the document. PV-DM's takes as input the document representation and the document with a hidden word and outputs the hidden word, while PV-DBOW takes as input the document identifier and outputs the words of the document. Although Doc2vec models can represent text, they do not take text structure and word order into account. This makes them less suitable for advanced tasks.

2.1.3 Recurrent Neural Networks

To improve the representation of text, more sophisticated methods based on neural networks have been proposed. In particular, RNNs make it possible to obtain fixed-size representations for sequences of variable length. They update hidden states, one token/word at a time, as the sentence is processed. Their output depends on the previous sequence elements. This process is illustrated Figure 2.1.3. They are mainly used for NLP, but can also be used to process time series or any sequential data.

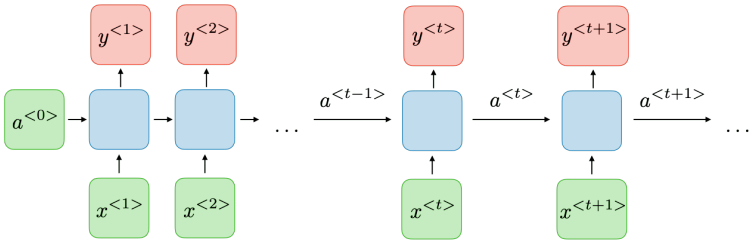


Figure 2.3: Architecture of a traditional RNN.
Source: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

Formally, let x be the input sequence of token embeddings $x^1, \dots x^n \in$. The hidden layer a^t and the output y^t are computed as follows:

$$\begin{aligned} a^t &= g_1(W_{aa}a^{t-1} + W_{ax}x^t + b_a) \\ y^t &= g_2(W_{ya}a^t + b_y) \end{aligned} \tag{2.1}$$

where g_1 and g_2 are activation functions (e.g. tanh or ReLU), and W_{aa}, W_{ax}, W_{ya}, b_a and b_y are the neural networks weights. Those weights can be learned through a language modeling objective (pretraining) and then fine-tuned on a specific task. The process is illustrated Figure 2.1.3.

While unidirectional RNN process sequences in a left-to-right direction, bidirectional RNN (Schuster and Paliwal, 1997) process them in two ways: left-to-right and right-to-left. This allows to take into account both previous and next tokens when encoding a token. This property is interesting for text, where words can have different meanings depending on the words that precede or follow them.

The weights W_{aa}, W_{ax}, W_{ya}, b_a and b_y are shared across time steps t, which make the network efficient. This is also interesting for text processing because they can process input of any length without modifying the network size. However, because of this recurrent architecture, the computation is not able to keep long-term information (Sherstinsky, 2020). In addition, backpropagation training can lead to vanishing gra-

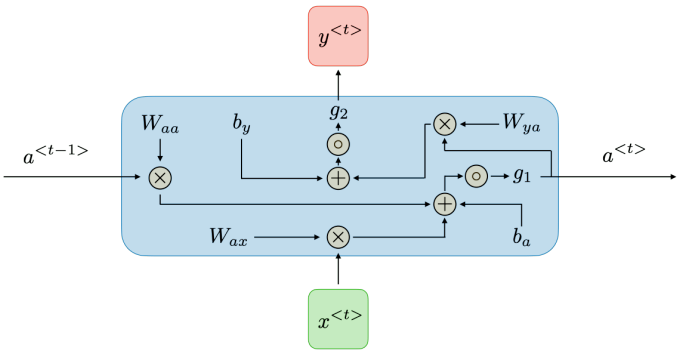


Figure 2.4: RNN operations at step t.
Source: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

dients[1] (Sherstinsky, 2020). This refers to situations where the gradient becomes smaller (vanishing) through the layers of the network and through the sequence's items or conversely increases to huge values (exploding). In both cases, the training is not successful.

To solve the short-term memory and training problems, alternative architectures such as GRU (Cho et al., 2014) and LSTM (Hochreiter and Schmidhuber, 1997) have been proposed. These problems can also be reduced with training tricks, as for example (1) truncated backpropagation through time (Jaeger, 2002) which consists of considering a moving window of the sequence, instead of the whole sequence, during training, (2) gradient clipping (Graves, 2013) which allows to control the magnitude of the gradient without changing its direction. These different works, although they provide improvements, do not entirely solve the issues concerning RNN short-term memory, gradient vanishing, and long training.

2.1.4 The Transformer architecture

RNNs have problems with long-term dependencies, while the newer Transformer architecture presented by Vaswani et al. (2017) does not.

Transformers (illustrated Figure 2.1.4) produce contextual representations of words. This is very interesting when dealing with text since the meaning of a word depends on its context. It has outperformed the state of the art in various NLP tasks, which I return to at the end of this section. This architecture consists of an encoder and a decoder that successively refine the representation of sequences. Unlike RNNs, transformers process the whole sequence at once and reintroduce word order information via positional embedding. In this thesis, I analyze

1 RNNs also lead to exploding gradients, but to a lesser extent.

the Transformer architecture and propose models based on it. I now describe it in more detail.

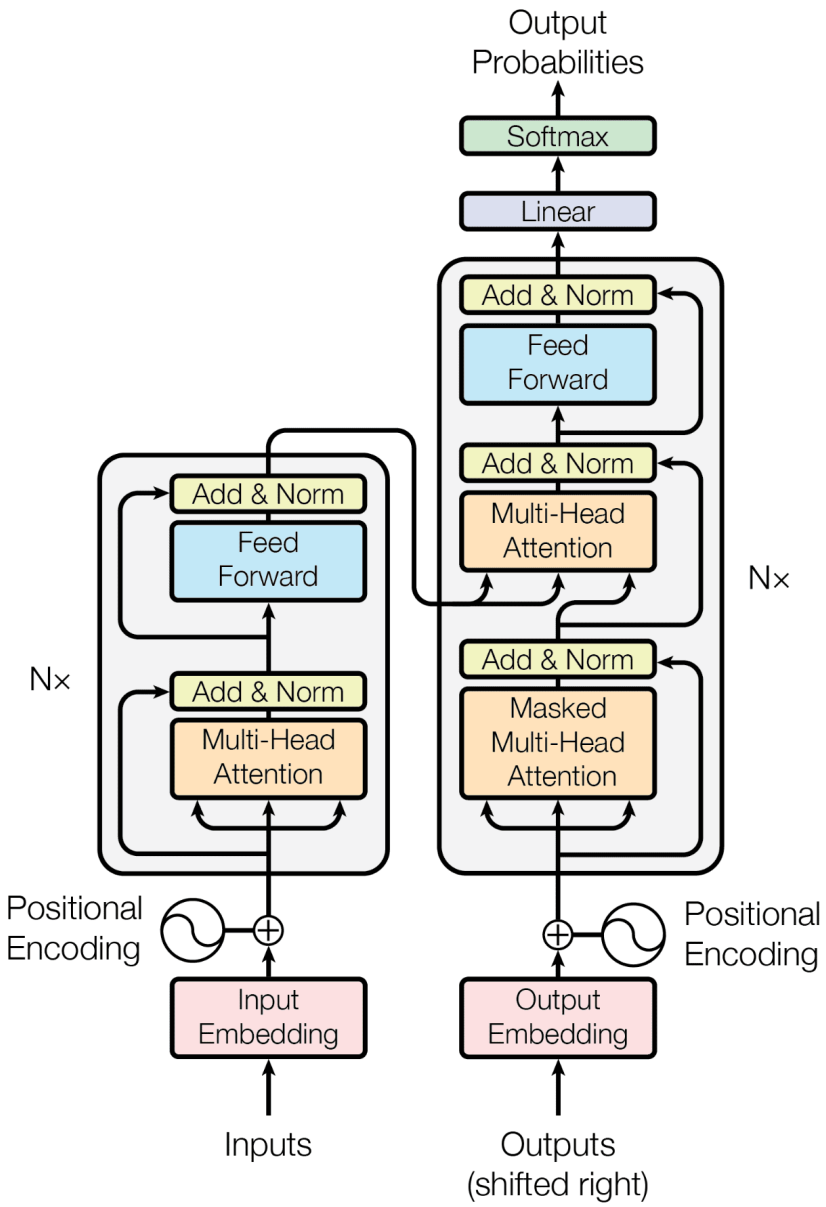


Figure 2.5: Transformer architecture. Illustration from the original paper (Vaswani et al., 2017)

Each layer of the encoder or of the decoder transforms a sequence x composed of n vectors $x_1, \dots, x_n$ into a sequence $y_1, \dots, y_n$ of the same length, through an attention over a context sequence c composed of n vectors $c_1, \dots, c_n$. Each time, the central mechanism is the use of an attention mechanism — other operations are performed to ensure a stable and efficient learning process, and are detailed in Vaswani et

al. (2017), but here we focus on the attention mechanism since it is important for the future analysis.

Attention heads and transformations At each layer of the encoder or the decoder, the transformation function T is based on the output of a series of H attention-based functions A_h (called *heads*).

For each head A_h, the attention mechanism (illustrated Figure 2.1.4) relies on:

- keys $k_h(c_j) \in \mathbb{R}^{d_k}$ computed for each element of the context c_j
- values $v_h(c_j) \in \mathbb{R}^{d_k}$ computed for each c_j
- queries $q_h(x_i) \in \mathbb{R}^{d_k}$ computed for each input token $x_i \in \mathbb{R}^d$, with $d = H \times d_k$.

Each input is decomposed in H parts of the same dimension d_k, i. e. $x_i = (x_{1i} \oplus \cdots \oplus x_{Hi})$ where $\oplus$ is a vector concatenation operation. Each x_{hi} is modified by a linear combination of the values $v_h(c_j)$ based on weights derived from the match between the query $q_h(x_i)$ with the different keys $k_h(c_j)$. More formally, we define a head A_h as:

$$A_{hi}(x, c) = x_{hi} + \sum_{j=1}^{m} \underbrace{\alpha_{hij} v_h(c_j)}_{\beta_{hij}(c_j)} \text{ with } \alpha_{hij} \propto \exp\left(\frac{1}{\sqrt{d_k}} q_h(x_i) \cdot k_h(c_j)\right) \tag{2.2}$$

where we can see that the attention mechanism only modifies the input if both the attention α_{hij} and the value $v_h(c_j)$ are not null. Each key, query, and value function is unique to a given layer and head. The output of the layer is given by $T(x, c) = (T_1(x, c), \ldots, T_n(x, c))$ with

$$y_i = T_i(x, c) = f(A_{1i}(x, c) \oplus \ldots \oplus A_{Hi}(x, c))$$

where f is a normalization followed optionally by a feed-forward layer.

The full transformation performed at layer l for a part $\bullet$ of the model is denoted as $T_l^\bullet$ in the following. The parameters of the corresponding heads (queries, keys, and values) are specific to each $T_l^\bullet$, where $\bullet$ is either the encoder self-attention e → e, the decoder self-attention d → d or the decoder to encoder attention e → d (see below).

Encoding When encoding, i. e. processing the input sequence $s^{(0)}$ of token embeddings $s_1^{(0)}, \ldots, s_n^{(0)}$, each layer transforms a sequence $s^{(l-1)}$ into $s^{(l)}$ using the transformation $T_l^{\text{e}\to\text{e}}(s^{(l-1)}, s^{(l-1)})$ based on the heads $A_{hi}^{\text{e}\to\text{e}}$ ($e \to e$ for attention from the **e**ncoder on the **e**ncoder).

Since the context is simply the input here, this is called a *self*-attention mechanism — i. e. each input item representation is transformed by looking at the whole input sequence. This is repeated L_e times until

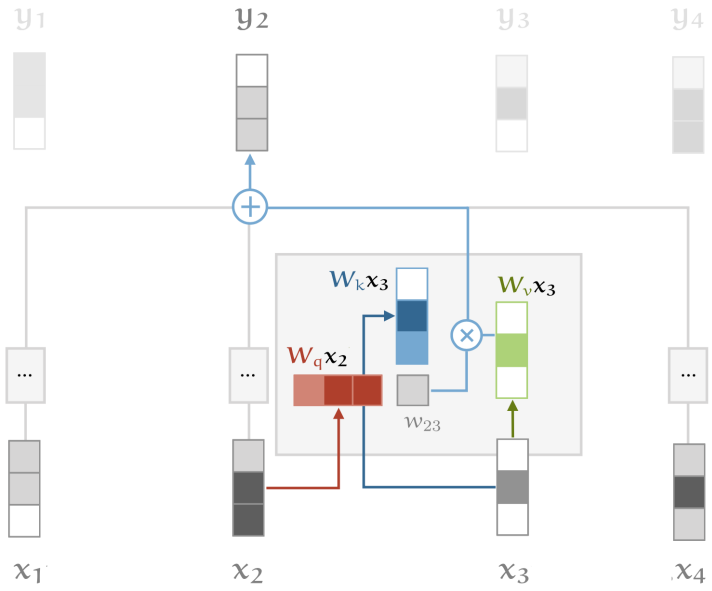


Figure 2.6: Illustration of the Transformer self-attention with key, query, and value transformations.
Source: https://peterbloem.nl/blog/transformers

obtaining the final representation of the encoded sequence $s^{(L_e)}$ which has the same length as the original input, but where each representation is *contextualized* depending on the other tokens of the input.

Decoding The generating process (called decoding) is based on the same principle — with a small twist since we take into account not only the already generated sequence but also the input. To compute the probability of generating a new token w given the sequence $w_0, w_1, \dots, w_{n'}$, whose embeddings are $t_0^{(0)}, \dots, t_{n'}^{(0)}$, the decoder uses two attentions: one self-attention $A^{d \to d}$ (decoder to decoder attention) followed by an attention on the encoded sequence $A^{d \to e}$ (decoder to encoder attention). The representation at layer l is based on the representation at layer $l-1$ and on the final encoded sequence:

$$d^{(l)} = T_l^{\mathrm{d}\to\mathrm{e}}\left[T_l^{\mathrm{d}\to\mathrm{d}}\left(t^{(l-1)}, t^{(l-1)}\right), s^{(L_e)}\right]$$

The process is repeated L_d times, giving rise to the representations $t_1^{(L_d)}, \dots, t_{n'}^{(L_d)}$. The distribution over the next token w (whose embedding is t) is then given by a parametric function applied to the representation of the last previously generated output $t_{n'}$ (so that to generate the first token, a special token [START] is used):

$$p(w|w_1, \dots, w_{n'}) = g(t; t_{n'}^{(L_d)}) \tag{2.3}$$

Sentence transformers The Transformer architecture proposed by Vaswani et al. (2017) is not suitable for computing the similarity between two sentences, nor for clustering. Indeed, Reimers and Gurevych

(2019) show that BERTrequires a large number of computations on each sentence of the database. This makes it very expensive, especially for IR tasks. For example, it takes 50 hours of computation on a V100 GPU to find the most similar question on the Quora site to another question with raw BERT. They propose an architecture that learns to predict the similarity between two sentences. The architecture is a Siamese network with mean pooling. With their method, it takes a few milliseconds to find the most similar Quora question.

2.1.5 Pretrained Transformers

Transformer models have a large number of parameters, which makes them costly to train. In addition, the attention mechanism is computationally expensive, especially for long sequences: it has a complexity of $O(n^2)$ with respect to the sequence length n (Wang et al., 2020). This makes them complex to train. But multiple pre-trained models trained on large datasets have been publicly released: BERT (Devlin et al., 2019), BART (Lewis et al., 2020), GPT-2 (Radford et al., 2019), T5 (Raffel et al., 2020), XLM (Conneau and Lample, 2019), RoBERTa (Liu et al., 2019b), and the famous GPT-3 (Brown et al., 2020) — whose parameters have not been made public. I now detail some of these models as they are used in the rest of this thesis.

2.1.5.1 BERT

BERT, the Bidirectional Encoder Representations from Transformers (Devlin et al., 2019) was released by Google in 2018 and remains the most popular pre-trained Transformer.

As it is usually done for large pre-trained Transformers, BERT is trained on a large dataset without any annotation, using self-supervision. In order to use the model for a specific task, transfer learning is typically used, i. e. some or all of the model weights are fine-tuned for the task with annotated data.

BERT consists of a transformer-based encoder. Its parameters are learned on two unsupervised tasks, namely Masked Language modeling (MLM) and Next Sentence Prediction (NSP). The former consists of randomly masking a percentage of the input tokens and then predicting these masked tokens. Although it allows the model to learn a representation of words in the context of a sentence, this first task does not take into account the relationships between sentences in the text. To overcome this problem, the NSP task requires the model to predict whether two sentences follow each other. The BERT authors show that, while simple, this task is crucial, and even more so for the Question Answering (QA) and Natural Language Inference (NLI) tasks.

BERT was trained on a large dataset: the BooksCorpus (Zhu et al., 2015) and all the text passages of English Wikipedia.

2.1.5.2 BART

BART, the Bidirectional and Auto-Regressive Transformer (Lewis et al., 2020), consists of an encoder and a decoder. Like BERT, it is trained on several tasks: token masking, token detection, text filling, sentence permutation, and document rotation. Because it has a decoder and is trained on these tasks, BART is better at text generation than BERT. The authors have also released fine-tuned versions of BART for other tasks. BART is trained on the same data as BERT.

2.1.5.3 T5

T5, Text-to-Text Transfer Transformer (Raffel et al., 2020), also has a transformer-based encoder and decoder as described in Vaswani et al. (2017) with minor architecture modifications in the attention mechanism. It is trained on numerous tasks such as machine translation, question answering, or text classification. The tasks are specified by adding their description as a prefix in the original input. Inputs and outputs are always text, even for tasks for which it is not so natural such as coreference resolution. Because inputs and outputs are always textual, the authors call it a "unified" framework. While often multitask models have a specific network for each task (Liu et al., 2020b), the T5 network is the same for all inputs. In our work on Query Generation (Chapter 3) this is however not problematic because queries are short.

2.1.6 Training Transformers

2.1.6.1 Supervised training

The transformer-based models presented above are trained with gradient descent. Their parameters are learned by minimizing a cost function, specific to the target task, called the loss. Let θ be the weights of the model to be trained, J the loss, and α a parameter called the learning step. The latter determines the step size performed to approach a minimum of the loss function J. Gradient descent is performed in this way:

$$\theta \longleftarrow \theta - \alpha \nabla J(\theta) \tag{2.4}$$

When training a language model, the most frequently used metric is perplexity. It measures the ability of a model to generate a sentence of the learned language. The lower it is, the better the model. With $y = (y_1, \dots, y_T)$ the target sequence of length T, and p_θ the probabilities according to the model, the perplexity is:

$$PPL(y) = \exp(-\frac{1}{T}\sum_{t=1}^{T} \log p_\theta(y_t|y_1, \dots, y_{t-1})) \tag{2.5}$$

When ground-truth sequences are used as model inputs during training, this is called teacher forcing. But when the model is used for predictions, the training sequences can no longer be used, thus the model inputs are the model's own predictions. The gap between training and inference inputs is problematic: the model may never have seen this kind of input (Goyal et al., 2016). This phenomenon is known as exposure bias. The bias increases with the length of the sequences to be generated, the gap increasing over the course of the generation. This is problematic when studying long sequences such as long-term research sessions.

2.1.6.2 Training for complex objectives

In addition to the difficulties related to sequence lengths, the proposed supervised training is not suitable for complex tasks. Indeed, for the latter, it is not always possible to design an appropriate cost. For example, to train an IR ranking model whose goal is to satisfy users during a search session, search systems, and users perform actions that are not immediately obvious to evaluate. We can only determine if the strategy used was successful or not at the end of the search session, for example, when users share their feedback. For this complex objective, it is impossible to derive a loss function.

To tackle such problems, Reinforcement Learning tries to take into account the effect of actions on the state of the world, even when actions cannot be evaluated immediately. RL attempts to learn strategies, called policies, which are evaluated by value functions. These reward functions estimate the reward of the last action. The goal of RL is to learn a policy that maximizes these cumulative values in the long run. For these various reasons, I use reinforcement methods in this thesis. In the following, I provide the RL formalism to understand them.

Reinforcement Learning framework I now formally set up the Reinforcement Learning framework and show how well the RL framework fits the search session task.

In RL, the problem to be solved is posed in the form of an MDP (Markov Decision Process). It relies on:

- the Markov states S of the world. A state has to be able to describe the system without the history (i. e. the previous states)
- the actions of the agent A
- the transitions of the system, i. e. the probability of being in a state s_{t+1} starting from a previous state s_t and an action a_t: $P(s_{t+1} = s' | s_t = s, a_t = a)$

- the reward function R which assigns a reward to a state and an action

During a search session, a state could be a set of user queries observed during a session. The actions of the search engine would be the documents presented to users. The reward is the explicit or implicit feedback from users. The optimal policy would then be the best ranking of documents during the session. More generally, the MDP is defined by the tuple (S, A, R, P) describing the set of states S, set of actions A, reward function R, and transition probability P.

We use the notion of cumulative reward which is the sum of the rewards obtained with a discount term $\gamma \in (0, 1)$ which penalizes the rewards obtained late. Let τ be a sequence of states and actions $\tau = (s_0, a_0, s_1, a_1, ...)$, and r_t the reward obtained at step t of that sequence.

$$R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t \tag{2.6}$$

The goal of RL is to find the policy that maximizes the expected return. The optimal policy is defined by:

$$\pi^* = \arg\max_{\pi} {}_{\tau \sim \pi}\mathbb{E}[R(\tau)] \tag{2.7}$$

I detail RL algorithms in Chapter 4.

2.1.6.3 Use of RL for various NLP tasks

RL has been used in a wide variety of NLP tasks, such as question answering, text generation, summarization, or translation. In those tasks, the actions correspond to the chosen word at each step of the generation process. The reward is a measure of the goodness of the generated text and task-related metrics. For example, for text summarization, a common metric is Recall-Oriented Understudy for Gisting Evaluation (ROUGE). ROUGE regroups an ensemble of metrics, one of the most common being ROUGE-N, which accounts for the overlap of n-grams between the generated summaries and the reference summaries. However, the ROUGE metrics are not differentiable, which prevents them from being used for supervised training. To overcome this difficulty, Paulus, Xiong, and Socher (2017) propose to use a dual objective: perplexity, which is the classical measure for language models and to optimize ROUGE using the self-critical policy gradient training algorithm.

In the remainder of this section, I describe examples of the use of NLP in a framework closer to our own, namely conversational agents. For more examples of NLP tasks learned with RL algorithms, please refer to the recent review from Uc-Cetina et al. (2022).

RL for conversational systems The use of RL for conversational systems is becoming more and more common (Yang et al., 2021a; Liu et al., 2020a). Through the combined use of deep learning and RL, conversational systems have made significant progress (Gao, Galley, and Li, 2018).

As an example, let us present the recent work of Yang et al. (2021a). Their dialog system learns to maximize 3 rewards related to topic coherence, semantic coherence, and grammatical correctness. The first reward computes the similarity between the topic representation of the history and that of the generated response. It is therefore particularly interesting because it has a direct impact on representations.

ChatGPT Recently, a conversational system trained with RL, ChatGPT, has been in the news because the sentences it generates are almost indistinguishable from sentences that would have been written by a human. ChatGPT model interacts with users in the form of a dialogue. It is capable of answering complex questions, although some of its limitations have already been identified (Wenzlaff and Spaeth, 2022).

ChatGPT is trained using Reinforcement Learning from Human Feedback (RLHF). The authors use the pre-trained Transformer GPT-3.5 as a starting point. Training is then performed in two steps. First, the model is trained in a supervised way on conversational data, and in a second step, the model is fine-tuned using PPO (Schulman et al., 2017), the RL algorithm. The reward model has been trained on manually rated dialogs, where annotators had to rank different generated answers for the same question.

2.2 Transformer for Information Retrieval

The use of Deep Learning in IR has led to state-of-the-art results. Especially, the attention-based text representation methods described in the previous section have enabled great progress in IR. In this section, I present this hot topic, called NeuIR (Neural Information Retrieval), and in particular the Transformer-based models for IR.

Early statistical IR methods were often based on bag-of-words representations (see Section 2.1.1), in which retrieved documents are returned directly to users. More recent work divides the retrieval process into two steps 1) retrieving documents and 2) re-ranking these documents. The first step retrieves a large number of results using sparse representations of queries and documents. The second step uses more accurate models to rank these documents. The works presented are divided according to these two stages: those based on enhanced representations to improve the retrieval stage of the system, and those proposing transformer-based architectures for the ranking stage.

2.2.1 Enhancing representations with Transformers

Many IR models depend on the representations of queries and documents. Hence, better representations can improve retrieval systems. Two groups of models are particularly dependent on the latter: sparse retrieval models and dense retrieval models. The former are based on sparse representations followed by indexing, while the latter propose to generate queries and documents low-dimensional representations used during the second stage of the pipeline, namely the re-ranking.

2.2.1.1 Sparse retrieval models

The first stage of the pipeline, which consists in retrieving a large number of documents, is critical; relevant documents that are not selected have no chance of being presented to users, and the performance of the re-ranking stage is limited by this selection. Sparse retrieval models propose improved query and document representations, which are then used by an inverted index.

Term weighting Instead of calculating the weight of terms based on their frequencies (as done in the standard bag-of-words methods Section 2.1.1), terms' importance can be calculated according to their context (Zheng and Callan, 2015; Frej et al., 2020; Dai and Callan, 2020). Embeddings from pre-trained models can be used directly or fine-tuned during ranking training to compute the importance of terms.

Dai and Callan (2020) estimate the importance of terms in passages and in documents by projecting each word's contextual representation from BERT into a single term weight. Instead of relying on term-level weights, Mallia et al. (2021) optimize the sum of query term weights to maximize the score difference between relevant and non-relevant documents for a query in their framework DeepImpact.

Document expansion Another method for improving the results of the first step of the pipeline is to add terms to the documents to reduce the vocabulary mismatch. The hypothesis is that by adding semantically related terms, we reduce the number of documents that are ignored during this first step. Again, various pretrained transformers have been used (Yan et al., 2021; Nogueira, Lin, and Epistemic, 2019; Nogueira et al., 2019). One of these interesting works is that of Yan et al. (2021) who use an encoder-decoder transformer where the encoder manages the re-ranking and the decoder manages the generation of terms to expand the document. Thus, the expansion of the document must be done in favor of the ranking of relevant documents.

Sparse representation learning While dense retrieval models have shown good performances, they are still combined with BOW

models because they are not able of explicit term matching. Thus, they are suffering from the same mismatch vocabulary problem than BOW methods. On the other hand, there has been a growing interest in learning sparse documents and query representation.

Jang et al. (2021) use the hidden representations of several layers of the model in a winner-take-all (WTA) model (Makhzani and Frey, 2015; Ahmad and Hawkins, 2015) that sparsifies the vectors. Formal, Piwowarski, and Clinchant (2021) combined term weighting and document expansion to produce sparse representations.

2.2.1.2 Dense retrieval models

Pre-trained Transformer models for NLP tasks provide a representation per token. Many works propose adaptations of these models to obtain a representation of a whole query or document that is useful for ranking. The following text representation methods are used to compute the similarity score between a document and a query.

Zhan et al. (2020) propose RepBERT, a BERT-based model of query and document representations where the token representations are averaged to produce a single vector.

Instead of trying to pool the token representations, other methods (Khattab and Zaharia, 2020; Gao, Dai, and Callan, 2021; Luan et al., 2021) use a multi-vector representation to compute similarity. One of the most popular models is ColBERT (Khattab and Zaharia, 2020). ColBERT, illustrated Figure 2.7, is considered as a *late interaction model* because query and document are encoded independently, as opposed to *all to all interaction model* where they are encoded at the same time (e.g. MonoBERT model). Note that here the term *interaction* is used to describe the relationships between queries and documents in ranking models.

ColBERT encodes queries and documents with two independent BERT encoders. The similarity is computed with the MaxSim operator which computes the summation of maximum similarity. The model parameters are learned by minimizing a pairwise softmax cross-entropy loss. Thus, with the ColBERT architecture the document representations can be pre-computed upstream.

2.2.2 Ranking with Transformers

Instead of working only on the representations of queries and documents, or on the first retrieval step of the IR pipeline, one can also improve the re-ranking step. Ranking models can be divided into two groups: discriminative and generative. The former learn to classify whether a document is relevant to a query, while the latter predict the

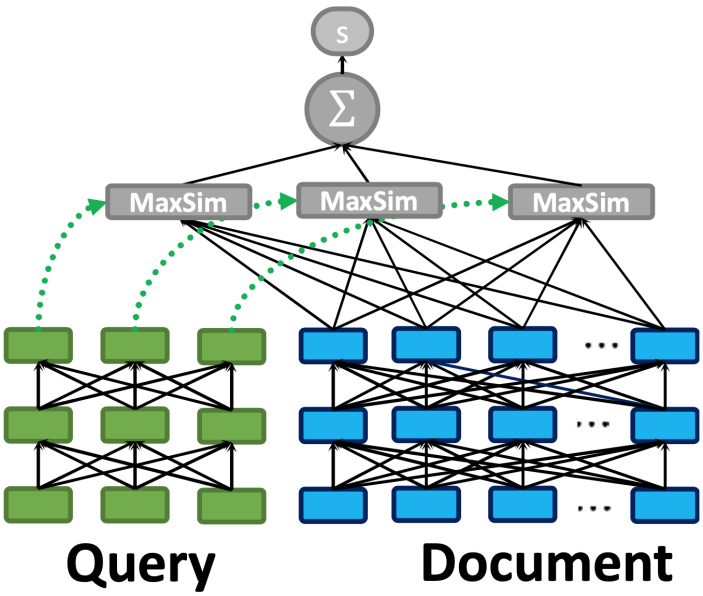


Figure 2.7: ColBERT architecture. Illustration from the original paper (Khattab and Zaharia, 2020)

probability of relevance by modeling the generative process between documents and queries.

2.2.2.1 Discriminative methods

Transformer-based models are interesting for ranking tasks because their attention mechanisms allow them to take query-document relationships into account. Thus, the MonoBERT model (Nogueira and Cho, 2019), which simply takes the concatenation of a query and a passage as input and learns to predict the relevance score from the embedding of the first token in the sequence, has achieved results far beyond the existing state of the art. Other models with variations in the architecture or training exist (MacAvaney et al., 2019; Pradeep, Nogueira, and Lin, 2021). However, MonoBERT remains the most popular of the discriminative models. Let's look at the model in detail.

MonoBERT The model is used in the second step of the classical retrieval pipeline, after having retrieved a large number of documents with BM25, the objective is to rank them.

The model takes as input the concatenation of the classification token [CLS], the query q, and the document d. The sequence is used as input to BERT, which allows to obtain a representation for each of the input tokens. The representation of the token [CLS] passes through a feed-forward network with two outputs corresponding to the classes: relevant/not relevant. The model weights are learned by minimizing the cross-entropy loss. Since the loss only considers the [CLS] token representation among the set of token representations output by the

transformer model, the representation of the [CLS] token captures the relationships between queries and documents.

Although this approach is simple, it achieves very good performance. However, queries and documents have to be processed simultaneously, making it impossible to precompute the document representation offline to make the whole process more efficient.

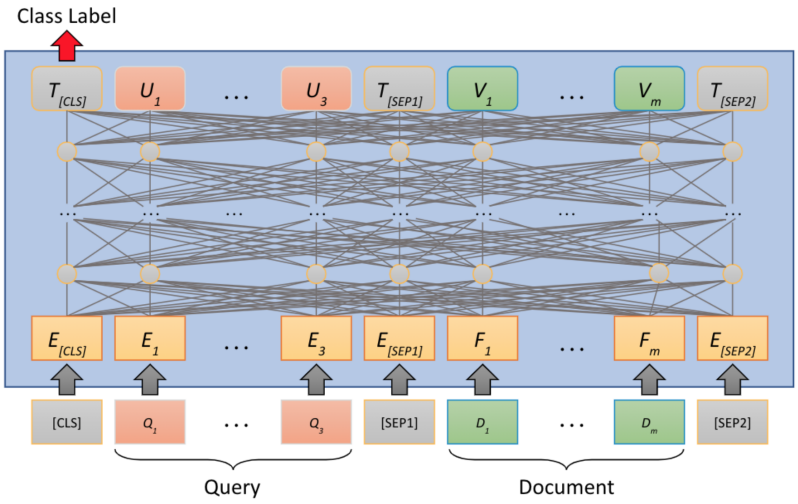


Figure 2.8: MonoBERT architecture.

2.2.2.2 Generative methods

Generative methods focus on the generation process between queries and documents, rather than learning the relevance score directly. Some transformer models are well suited to this task, in particular, the encoder-decoder transformers learned from language generation tasks. There are two types of generative methods: the ones that generate a relevance token from documents and queries, and the ones that generate queries from documents.

Relevance models generate a token (usually "relevant"/"irrelevant") from a document and a query. The probability of generating the relevant token is considered the relevance score. Nogueira et al. (2020) uses the T5 model to generate the relevance token. By comparing the performance of the complete model (encoder-decoder) with the encoder alone, the authors show that the decoder is crucial for this task. Furthermore, when changing the relevance tokens (true/false) to other words (yes/no) the results are less good. This shows the importance of pretraining: even when fine-tuned, the model uses the knowledge learned during pretraining on a large dataset.

Relevance generation methods estimate the probability of generating a query from a document to determine its relevance. Recently, a hybrid method of relevance and query generation has been proposed

by Liu et al. (2021). Their multi-task learning approach (ranking, query generation, and question answering) allows to obtain a more complete model. It is common to learn transformers on several tasks in order to make them more robust.

2.3 User Modeling

Having described state-of-the-art IR models, we now go back to the problem of user modeling, focussing on IR-related models.

A landmark work in Information Retrieval proposes a taxonomy of user intents (Broder, 2002). The authors study the behavior of search engine users and show that there are different types of search implying different behaviors. They define three types of search: navigational, transactional, and informational. This classification can be used to improve search results, e.g. Tsukuda et al. (2013) who use it to diversify search results after classifying the user's intent, while (Santos, Macdonald, and Ounis, 2011) propose to use different IR algorithms depending on the type of intent of the initial query. These different works show two essential points. There are different types of search, and taking into account the type of search can improve the relevance of search results. These findings lead to a search for more refined user modeling.

Among user modeling techniques, representative-based techniques are the most suitable because they allow for great flexibility in terms of modeling. The modeling of users with machine learning methods consists in learning a latent representation from the users' past actions or personal characteristics (age, gender, preferences, location, ...). In this thesis, I am not interested in modeling users based on this latter type of personal data called personalization (Abri, Abri, and Cetin, 2020), but instead, I follow the first direction that relies on user logs datasets.

User simulation requires complete user modeling, while other tasks such as intent prediction or next-click prediction can be tackled with partial modeling. I present user modeling methods in two parts: first, user simulation work, and then tasks that allow to obtain a user model indirectly. The methods discussed are not all from standard textual IR, but also come from conversational search, recommendation, and reinforcement learning. I present them because I am convinced that these methods can be as useful for Interactive Information Retrieval as NLP methods have been before.

2.3.1 User simulation

User simulation in an unconstrained setting such as a search engine is complex: the data are large and unstructured, and users have multiple

modes of action (clicks and queries). Thus, there are more works on user simulation in more restricted settings, such as recommendation systems. In this section, I present work on user simulation in these different settings.

Classic search Maxwell and Azzopardi (2016a) propose to simulate a user during a search. The simulated user can write a query, click on a document, and decide to continue or stop the search. The simulation is based on a model representing the user's cognitive state. The cognitive state has multiple components: prior background knowledge, user information needs, lists of previous interactions, and several models: a query generation model, a model evaluating the attractiveness of a snippet, and a document relevance model. The attractiveness of a snippet and the relevance of a document depend on language models computed on sets of attractive snippets and relevant documents.

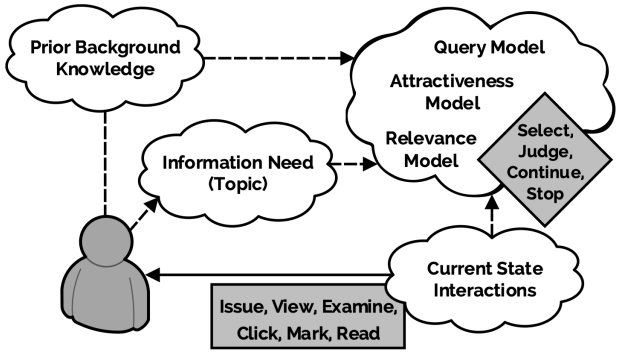


Figure 2.9: Proposed framework of Maxwell and Azzopardi (2016a)

The model follows a framework based on a set of strong assumptions and does not allow to obtain a latent representation of the user. However, it is interesting since it represents the user as a set of components ranging from their information needs to their interactions in the context of a search on a classical search engine.

The authors extended their work with subtopic selection strategies (Câmara, Maxwell, and Hauff, 2022), which allows a decomposition of complex search tasks into simpler ones, parametrized with tunable parameters (learning speed, exploration, tolerance, and subtopic switching). Although this extension makes the simulation more realistic and allows to simulate different types of users, it still makes the framework more complex and does not allow to obtain a latent representation of the user, which restricts the use of such system to simulation.

The next papers of the section address more restricted settings.

Commercial search engine The AESim (Gao et al., 2021) platform allows to train and evaluate ranking models for commercial search engines via a simulated user. In this simulation, a virtual user performs a query that is part of a finite set of categories, the ranker proposes a list of products, and then the user gives feedback on these products. The learned feedback concerns the clicking and purchasing of the proposed products. They show that their simulated user has a purchasing behavior similar to real users. The virtual user is learned from a set of real users logs. It is composed of two modules: a user module, and a feedback module. The first module learns to represent the user via a set of features. It is composed of a generator and a discriminator. The generator generates user-query pairs and the discriminator learns to distinguish true from false pairs. The models are learned with the WGAN-GP algorithm (Arjovsky, Chintala, and Bottou, 2017; Gulrajani et al., 2017). The second module simulates user feedback, i. e. the prediction of clicks and purchases per presented product. It is learned with the GAIL algorithm (Ho and Ermon, 2016).

This platform is interesting because it allows to learn and evaluate models without using a search engine, or real users. However, it takes place in a restricted search framework, since it is a commercial search engine, with a finite number of initial queries and a limited number of actions for the user (a single query and the following clicks). The other major difference with a classic textual search is that we have very implicit feedback on the user's satisfaction, which is the purchase of one of the presented products.

Conversational search First works on simulated users in conversational search are agenda-based user simulators: authors define a set of hand-crafted rules named an *agenda* (Schatzmann and Young, 2009; Li et al., 2016; Schatzmann et al., 2007). These rules-based approaches have weaknesses. They require feature engineering work to define precisely and exhaustively the various users' needs. For complex situations, a complete taxonomy cannot necessarily be established. Moreover, even if the agenda is sophisticated, it only includes the situations present in the training set, thus the simulated user is stationary and cannot learn to adapt to new scenarios.

Setting such an agenda is restrictive. Other works are based only on conversational data from real users, without using any preconceived notions about the structure of the dialogue. These works can be classified into two groups: those learned only through supervision, and those in which reinforcement algorithms are used.

First, there are works based on real conversation logs learned in a supervised way. These sequence-to-sequence methods require a large amount of data (Asri, He, and Suleman, 2016; Crook and Marin, 2017). Asri, He, and Suleman (2016) seq-2-seq user simulator model takes as input

the dialogue context. They propose a method for encoding the conversation history by using one-hot-encoded vectors containing information about the system response types (such as "confirm", "request", and "inform") and their consistency with previous answers. Crook and Marin (2017) use a RNN model that takes as input the raw dialogues without any processing. Instead of using annotated data, they propose an architecture with an extra RNN encoder that embeds the history context. These models are learned by optimizing cross-entropy, but this loss is not always associated with a good user model.

To optimize a metric that better fits the problem, other work uses reinforcement learning to train their models. (Gür et al., 2018; Chandramohan et al., 2011; Kreyssig et al., 2018). They follow the RL formalization, with an MDP in which the agent is the simulated user. Generally, the main difficulty of these methods is to define a reward. Moreover, RL-based training tends to lead to degeneration in the generated conversations with repetitions or irrational texts (He et al., 2018). Gür et al. (2018) and Kreyssig et al. (2018) use a common reward in task-oriented dialog for which the simulated user goal is known: at each new utterance there is a small penalty, and at the end, if the dialog is successful, there is a larger reward. Since RL algorithms are being used more and more for text generation tasks (as discussed in Section 2.1.6.3), we can expect that they will allow great improvements in user simulation for conversational search in the next few years.

Simulating a user during a search on a classic textual search engine is quite complex, especially when one tries to avoid strong assumptions like the agendas (Asri, He, and Suleman, 2016). Thus, there is more work concerning user simulation on more restricted applications such as chatbots. Indeed, on a search engine, user actions are quite varied (clicks, query reformulation, time spent on a document, ...) feedbacks are implicit and not easy to decode, and data are very large. To model users, instead of trying to simulate them, we can rather go through intermediate tasks like predicting their intention, their next request, or their next click. This is what I propose and study in Chapter 3.

2.3.2 Partial user modeling

Users have multiple ways of interacting with search systems, making their simulation very complex. Full user simulations follow a structured framework that does not allow for a latent representation of users. In this section, I present modeling work that addresses modeling a single dimension of search, namely intent, queries, and clicks modeling.

2.3.2.1 Intent prediction

In their article, Ruotsalo et al. (2014) claim that "Interactive intent modeling can improve task-level information-seeking performance by over 100%". This is true for several reasons. First, users do not always know how to express their information need correctly, especially in the case of a complex search task. Secondly, because their intent may change throughout the research process, for example, if they learn new information about their initial intent.

Thus, several works focus on modeling users' intent. Some of them seek to split a search task into subtasks, or a multitask search into different tasks (Liu and Belkin, 2010; Kotov et al., 2011). Yang et al. (2020) use an intent taxonomy (such as follow-up question, positive feedback, clarifying question, etc...) and add to a classical transformer an intent-aware attention mechanism. The latter learns to classify the intent of each sentence in the conversation. This model provides users' representation at each stage of the search process.

Intent models are of great interest in this thesis, especially the modeling of the user's goal, which can be updated during the search session.

2.3.2.2 Query suggestion

Query suggestion is a tool that proposes a list of queries to users to help them with their search. One of the methods to generate suggestions is to predict the user's next queries using search logs, in order to suggest a query that might accelerate the search process (Sordoni et al., 2015; Mustar, Lamprier, and Piwowarski, 2021; Dehghani et al., 2017). Seq-2-seq models (Section 2.1.3), such as RNNs or transformers, allows to obtain a latent representation of users, by using the encoder output. There are also other methods for query suggestion which I review in the next chapter.

2.3.2.3 Click model

Click models aim to predict how users interact with the list of returned results. Clicks can be seen as implicit feedback from users. Click models can be used to learn ranking models (Dai et al., 2021), to cluster users (Punera and Merugu, 2010), or as a metric to compute the relevance scores for query-document pairs (Chen et al., 2020).

Several of these methods are based on a latent representation of users. Neural click model (NCM) from Borisov et al. (2016) represents users as a sequence of hidden states, while the click sequence model (CSM) (Borisov et al., 2018) uses an encoder-decoder architecture, where the encoder computes contextual embeddings of the documents and the decoder predicts the next clicked documents.

Many ways of modeling user behavior have been identified, although no fully satisfactory model of users performing their searches on a general search engine exists to date. However, while not perfect, these models can be used to work on interactive search systems, which I discuss in the next section.

2.4 Personalized systems

A promising avenue to increase users satisfaction in IR is the use of interactive systems. In this section, I present interaction systems based on user models. I first review interactive systems that require explicit feedback from users, before I describe systems that exploit implicit feedback, such as clicks or queries, to improve results.

2.4.1 Interactive systems based on explicit feedback

During a search, users' actions (clicks, queries, etc.) are not always interpretable. For example, a click on a document does not necessarily imply that they are satisfied. One way to deal with the problem may be to directly ask users to answer questions, either to clarify their needs or to find out if they are satisfied.

Intent prediction in structured spaces The Q20 game consists in guessing the object or the person the player is thinking about by asking a maximum of 20 questions. The game is studied (Burgener, 2006; Hu et al., 2018; Wu et al., 2018b) because it allows working on interaction models in a small and controlled framework. Indeed, the search space is much smaller than the one of a search engine. Each item is described by a set of attributes which is easier to leverage by such systems.

Differently, data from search sessions are much less structured, which makes the work on interactions more complex. Hu et al. (2018) learn a probability distribution over objects, which is updated after each response from the user. Their method requires a large amount of data because each object and each question requires multiple answers to learn an accurate distribution.

Yu et al. (2019) work in a similar environment, the agent asks users clarifying questions to guess their goal using an information gain criterion to determine which question to ask. This criterion, calculated for each question, is based on the entropy of goals distribution before and after asking the question. However, as the work on the Q20 game, a knowledge base of attributes by goal is needed.

Recommendation system Recommendation systems, in which the characteristics of the products to be recommended are usually available, rely on explicit user feedback.

For example, Christakopoulou et al. (2018) use the fact that they know the topic (e.g. "sketch comedy" or "horror") of the recommended videos. Thus, to improve their system, they propose to first ask a question to users about their topics preferences: the system proposes a list of topics likely to interest them, and this feedback is taken into account by the recommendation system. The system has two components: a question generator Q and a recommendation model R. Q takes as input the sequence of topics seen up to t and is trained to learn to predict the topic of the video at $t + 1$. While R takes as input the history and the topic chosen by users when asked and learns to predict the video watched at $t + 1$.

Similarly, product recommendation systems have taken advantage of the fact that they know a set of attributes about products to query the user (Zou and Kanoulas, 2019; Lei et al., 2020). Zou and Kanoulas (2019) proposes a new interactive method for users to find the right products. After an initial query, the system asks users about the attributes of the searched product to discriminate it from others. When it considers that it has acquired enough information about the user's need, it proposes a list of products. The model is learned with a reward which is the evolution of the rank of the target product after asking the question.

Systems can try to use their feedback after returning an initial list of results. Zhang et al. (2020) propose to ask directly users to give their feedback in natural language on the proposed products. Their model uses this feedback to update the recommendations. To do so, they use a discriminator that evaluates whether the recommendations match the feedback given by users. The score of the discriminator is used to compute the reward of the recommendation model: $R_t = r_t - \lambda_k c_t$, where c_t is the score of the discriminator and r_t is the score of the initial recommendation model.

Recommendation systems, like search systems, seek to satisfy users. But unlike the latter, they have a structured space (e.g., with attributes) and explicit metrics to judge user satisfaction (e.g., a purchase at the end of a session). Thus, most of the methods are not applicable in the context of this thesis, which is restricted to Interactive Information Retrieval.

Conversational search Conversational search systems are a special case of explicit feedback, as by design, users can provide feedback in natural language. In various conversational search systems, the search agent and the simulated user are learned jointly (Liu and Lane, 2017; Shah et al., 2018; He et al., 2018). These works are placed in the framework of an RL problem with a MDP with two agents to learn. They can be difficult to train because the deficiency of one agent (divergence) can lead to the deficiency of the other.

Furthermore, it is necessary to manually define a reward for each agent. In a restrictive system, it is easier to reward the agent. For example, He et al. (2018) compare three rewards for simulating a user in a price negotiation dialogue. The first is a linear function of the final price negotiated, the second encodes a notion of fairness between the two agents, and the last encourages dialogue length. The authors note that although the intended rewards objectives are achieved, the conversations are less natural than with supervised learning. This questions the appropriateness of the proposed reward.

For more general systems, a simple heuristic cannot work. Chandramohan et al. (2011) overcome the difficulty of manually setting a reward by using Inverse Reinforcement Learning. This type of learning consists of imitating the traces of an expert, without knowledge of expertise. They learn a reward function to maximize the distance between the expert and the simulated users. This reward is then used to train the simulated user.

To improve further conversational systems, user feedback can be requested even more explicitly by asking users questions. Several studies have investigated ways of asking questions to disambiguate users' intent (Zamani et al., 2020; Dhole, 2020; Cao, Rao, and Daumé III, 2019). Dhole (2020) disambiguates users' initial query by asking a question to discriminate their goal. The question is chosen according to the predicted distribution of an intent classification model.

Conversational systems are becoming increasingly popular, as we have seen with ChatGPT. Search engines are getting closer to these systems by being able to answer queries in natural language. Both types of systems have benefited from advances in NLP (with transformers) and RL, so the boundary between them is shrinking.

2.4.2 Interactive systems based on implicit feedback

Explicit feedback systems require actions from users. This changes their behavior and requires additional time and effort on their part. It is therefore interesting to study weak signals of users satisfaction, such as query reformulation, clicks on documents, time spent reading them, ignored documents, etc.

A sequence of queries can be used to find the user goal. For example, a user who starts with the query "jaguar" and then after considering the results specifies a second query "jaguar animal", sends a signal to the search engine. Yang, Guan, and Zhang (2015) study users behavior by focusing on the syntactic query changes during a session, such as deleting or adding words. Two agents are learned, the user whose actions are query changes, and the search engine whose actions are term weights adjustments.

Another type of signal is ignoring the first returned documents and clicking on links further in the results list, users send a signal to the search engine. Using eye tracking, Joachims et al. (2005) show that while clicks are not perfect relevance judgments, they provide good estimates of relative preferences between documents. Thus, instead of taking into account exclusively word changes in submitted queries, various work focus on clicked or not documents (Zhao et al., 2018; Levine, Roitman, and Cohen, 2017; Li et al., 2019). Levine, Roitman, and Cohen (2017) consider not only the syntactic changes, but also the returned documents, and those clicked in order to adjust the weights given to each of the terms of the last query. Zhao et al. (2018) propose an architecture with two different RNNs to encode separately clicked and ignored items of their recommendation system. The outputs of these RNNs are then concatenated to recommend new items. The model is learned via Reinforcement Learning rewarding clicks and purchases.

Let us end this section with an article that is not directly related to Information Retrieval, but whose framework is inspiring for interactive systems designs. Li et al. (2020) propose a framework in which a system and a user interact. They consider that as long as the user continues to interact with the system, the latter is valuable. In the framework, the user and the system are two agents. They are learned sequentially: at one learning iteration the agent policy is learned, and at the next one the user is updated with the newly learned policy. A particularly interesting aspect of this work is that rather than manually defining rewards, which is, as seen above, perilous for complex goals, they propose to infer goals from observed interactions. To estimate the rewards, the authors use inverse reinforcement learning and assume that rewards can be estimated by a linear combination of the state features. They assume that the reward can be estimated from the state.

Although the proposed MDP is interesting, it is not directly applicable to the framework of this thesis. In fact, the simplification that allows to represent a state in a vector without any learning and the fact that it rewards the longest interactions do not correspond to a realistic or desirable search system.

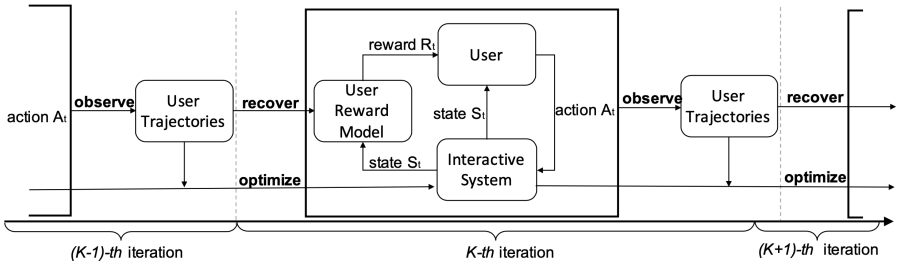


Figure 2.10: Li et al. (2020) framework

In most of the papers presented above, interactions between a user and a search system are based on a model learned by reinforcement learning. Reinforcement environments require a simulated user who usually has knowledge of the taxonomy of products that can be recommended to them, and thus always answers the system's questions. In the context of a classical search engine, it is impossible to establish such a taxonomy of needs. In the next chapter, I focus on user modeling through the prediction of their next query.

Conclusion In recent years, NLP has made great progress with the powerful transformer models. The latter have been used in many other tasks, including IR, as we have seen with the monoBERT and colBERT models. But while they have been used for sparse and dense retrieval models, they have been little used for user modeling in a search task. Moreover, they have been shown to outperform the existing state-of-the-art in many domains, but little analysis has focused on their functioning.

On the other hand, existing work on interactive systems is promising, but it is still at an exploratory stage, and there are still many challenges to overcome. These are in limited or simplified settings. Explicit user feedback is difficult to obtain, but implicit feedback is difficult to make explicit. For these reasons, we believe that interactive information retrieval can be greatly improved.

Chapter 3: Self-Attention Based Query Prediction

User modeling is essential for improving IR systems. As seen in Section 2.3, one way to build user models is to predict users' behavior based on their previous actions. Indeed, if a model is able to correctly generate queries, it has (at least partially) captured the user's intent. In this chapter, I explore generative language models for modeling users during a search session. Working with representation-based models - such as neural networks - is particularly interesting for the next query prediction task because the learned representations can be useful for models exploiting user sessions, such as interactive IR models that rely on a semantic representation of the user state.

Furthermore, when performing a search task, users may find it difficult to articulate their needs, especially when the task is complex. To help them complete their search, search engines typically provide query suggestions. While a good query suggestion system requires modeling user behavior during the search session, user models based on queries can serve as query suggestion models. The models studied can serve two purposes, namely user modeling and query suggestion.

This chapter is an adaptation of two papers published during my thesis: Mustar, Lamprier, and Piwowarski (2021) and Mustar, Lamprier, and Piwowarski (2020) that investigate this issue.

3.1 Motivations

To explore the space of potentially relevant documents, users interact with search engines through queries. This process is particularly valuable when users are accomplishing a complex search task (Liu et al., 2019a). Among the different ways to help users in exploring the informa-

tion space, modern search engines provide a list of query suggestions, which help users by either following their current search direction—e.g. by refining the current query—or by switching to a different aspect of a search task (Ozertem et al., 2012a). Another use of query suggestions is to help search engines by providing ways to diversify the presented information (Song, Zhou, and He, 2011).

There are two ways to approach the task of query suggestion. Either in a direct way, seeking directly to improve user experience. This involves searching for the most suitable queries so that users access the most relevant information as quickly as possible (Bhatia, Majumdar, and Mitra, 2011). Such an approach requires a mean to assess what constitutes a relevant suggestion, or data on whether or not suggestions are relevant. The second approach consists in modeling the average user (Broccolo et al., 2012; Sordoni et al., 2015; Dehghani et al., 2017; Ahmad, Chang, and Wang, 2018; Ahmad, Chang, and Wang, 2019; Wu et al., 2018a). The goal is to predict the next query based on the current search session where learned systems can leverage huge datasets (i. e. query logs, etc.). The hypothesis is that by proposing their future queries to users, they skip search steps and reach their goals faster, thus the system would help them in their search. In the absence of a public dataset allowing to train and evaluate models on the first type of approach, this latter type of approach is usually pursued.

To suggest useful queries, most models are built upon web search logs, where the actions of users (queries, clicks, and timestamps) are recorded. User sessions are extracted by segmenting web search log. The first query suggestion models exploit query co-occurrence graph extracted from user sessions (Huang, Chien, and Oyang, 2003; Jain, Ozertem, and Velipasaoglu, 2011a): if a query is often followed by another one, then the latter is a good potential reformulation. However, co-occurrence based models suffer from data sparsity. For instance, when named entities are mentioned, they lack of coverage for rare or unseen queries. Moreover, these models are difficult to adapt when using a broader context than the last submitted query (Dehghani et al., 2017).

More recently, recurrent neural network-based (RNNs) methods have been proposed to exploit longer dependencies between queries (Sordoni et al., 2015; Dehghani et al., 2017; Ahmad, Chang, and Wang, 2018; Ahmad, Chang, and Wang, 2019; Wu et al., 2018a). RNNs do so by keeping track of users in a representation/vector space which depends on all the previous actions performed. Such models have improved the quality of suggestions by capturing a broader context, but are limited by the relatively short span of interaction that RNNs are able to capture. Moreover, these networks are black-boxed: the aspects on which the suggestion generation mechanisms are based are unknown (Sussillo and Barak, 2013).

Among all the models exploited in NLP and IR, most (Vaswani et al., 2017; Tan et al., 2018; Liu et al., 2018; Scialom et al., 2019b; Qiao et al., 2019; Yang, Zhang, and Lin, 2019) have benefited from the Transformer architecture (Vaswani et al., 2017). Transformer networks, such as BERT (Devlin et al., 2019), capture long-range dependencies between terms by refining each token representation based on its context before handling the task at hand. They are thus a particularly interesting architecture for query suggestion since query terms are often repeated throughout a session (Sloan, Yang, and Wang, 2015) and their relationships need to be captured to build a faithful representation of the user's current state. Recently, Garg, Dhillon, and Yu (2019) presented a hierarchical transformer for query suggestion, with a two-level encoder. Their model outperforms the hierarchical recurrent-based models (Dehghani et al., 2017; Sordoni et al., 2015), and shows that recurrence is not essential for the query suggestion task. In opposition to this type of hierarchical transformers, we refer afterward to the classical transformer networks as flat transformers.

However, the authors Garg, Dhillon, and Yu (2019) do not provide a full analysis of whether the hierarchical architecture is important, especially for complex user sessions which are particularly interesting in the context of interactive IR. In this chapter, I study transformers for the query suggestion task—and more generally, for user models. The outline is as follows:

- A reproduction of RNN-based models experiments (Dehghani et al., 2017; Sordoni et al., 2015). The use of techniques from transformers that segment queries using subword units, which allow models to avoid the problem of out-of-vocabulary tokens which limit the usefulness of RNN for query suggestion.
- A reproduction of the hierarchical Transformer architecture (Garg, Dhillon, and Yu, 2019), with word and sub-word units, and a comparison with flat transformers.
- A comparison of the flat transformers with three pre-trained transformers: BERT, BART and T5, fine-tuned for the next query prediction task. The encoders of these pre-trained models are also integrated to the hierarchical transformer.

The analysis is structured into three research questions detailed below. First, the study of the transformers' performance from a global point of view.

Q1. How well the various presented transformers generate query suggestions compared to the usual baselines?

When a user performs a complex search, it is more difficult to capture the intent of the user. However, such sessions are of particular interest

for nowadays IR research, and in particular for interactive IR. Particular attention is thus paid to the robustness of the different models on sessions corresponding to so-called "complex" search tasks. This raises the question of whether all transformers have the same ability to handle long, complex, or noisy sessions, or whether, on the contrary, the results are impacted differently depending on the pretraining or the architecture of the transformer:

Q2. Which model is the most robust?

a) to complex sessions

b) to noisy sessions

c) to long sessions

Following the analysis conducted to answer **Q2.**, we conclude that flatten pre-trained transformers are more resilient to noise, length, and complexity of sessions. The understanding of the reason for the robustness of this model leads to the final research question:

Q3. How does the flat transformer generate queries?

a) On which context's queries does it focus its attention?

b) On which context's tokens does it focus its attention?

c) How does it choose the next token to generate?

The analyses and answers to these questions aim to better understand the behavior of various Transformer architectures for user modeling.

3.2 Query suggestions methods

3.2.1 Formalization

Before presenting the different methods of query suggestion, let us formalize the problem.

Let us consider a session $S = (Q_1, \dots, Q_{|S|})$ as a sequence of $|S|$ queries, where every query $Q_i = (w_{i,1}, \dots, w_{i,|Q_i|})$ is a sequence of $|Q_i|$ words. The goal of query suggestion is to propose the most relevant query for the user intent, which is represented by the session. However, no perfect ground truth can be easily established for such problems. Defining the perfect query for a given need, given a sequence of past queries, is an intractable problem. It requires to consider very diverse (in nature and complexity) search tasks, and it depends on the user state, the IR system, and the available information in the targeted collection. Thus, the chosen task consists in predicting the next query within an observed session.

We suppose that our dataset is composed of pairs $(S, \breve{Q})$ where $\breve{Q}$ is the query following a sequence of queries S.

For the generative methods, the aim is to find the parameters θ that maximize the log probability of observing the train dataset:

$$\mathcal{L}(S;\theta) = \sum_{(S,\check{Q})} \log p_\theta(\check{Q}|S) = \sum_{(S,\check{Q})} \sum_{t=1}^{|\check{Q}|} \log p_\theta(w_t|Q_1,\dots,Q_{|S|}) \quad (3.1)$$

where $(w_1,\dots,w_{|\check{Q}|})$ are the token of the query $\check{Q}$.

3.2.2 Co-occurence, graph and similarity

A large number of works have focused on the task of query suggestion (Ozertem et al., 2012b), and related tasks such as query autocompletion (Mitra and Craswell, 2015), based on search logs to extract query co-occurrences (Huang, Chien, and Oyang, 2003; Jain, Ozertem, and Velipasaoglu, 2011a). From a given single query formulated by a user, the goal is to identify related queries from logs and to suggest reformulations based on what follows in the retrieved sessions, assuming subsequent queries as refinements of former ones (Sadikov et al., 2010). These works rely on several methods, such as using term co-occurrence (Huang, Chien, and Oyang, 2003), users click information (Mei, Zhou, and Church, 2008), or word-level representation (Bonchi et al., 2012); capturing higher order collocation in query-document sub-graphs (Boldi et al., 2009); clustering queries from logs (Sadikov et al., 2010); or, defining hierarchies of related search tasks and subtasks (Hassan Awadallah et al., 2014; Mehrotra and Yilmaz, 2017).

Methods based on similarities with existing queries in the logs encounter difficulties when confronted with queries never seen before, they are not always able to generalize. The approach in Cao et al. (2008) attempts to alleviate this sparsity problem by relating the user session to paths in a concept tree. Other works prevent query sparsity via reformulations using NLP techniques (Ozertem et al., 2012b). For instance, Jain, Ozertem, and Velipasaoglu (2011b) present an end-to-end system to generate synthetic suggestions based on the removal of non-critical terms in the available text resources. Broccolo et al. (2012) propose to alleviate the sparsity issue by creating a knowledge base from query logs. This database contains synthetic documents produced from train log queries. When users perform a query, a function measures the similarity between this query and the documents in the database. The closest document's title is then used as a suggestion.

However, even for the latter methods, such log-based methods suffer from data sparsity and are not effective for rare or unseen queries (Sordoni et al., 2015). In addition, these approaches are usually context-agnostic, focusing on matching candidates with a single query. When the query comes in a session with some previous attempts for finding relevant information, it is crucial to leverage this context for capturing the user intent and understanding its reformulation behavior.

Instead of trying to predict directly a query, it is possible to learn how to transform it. Most approaches operate at a high level, with term retention, addition and removal as the possible reformulation actions (Levine, Roitman, and Cohen, 2017; Sloan, Yang, and Wang, 2015). Levine, Roitman, and Cohen (2017) consider these actions as feedback from the user – e. g. a term that is retained during the whole session should be considered as central for the user intent. Depending on the previous sequence of user actions, these methods seek to predict their next action. They are interesting because they model users' behavior in a session. However, they focus on the syntactic changes in queries and do not capture the relationships between words and thus the overall meaning of queries (Jiang and Wang, 2018).

3.2.3 RNN based methods

To cope with the limitations of log-based and action-based methods, some works propose using probabilistic models for next query prediction (He et al., 2009). Due to their ability to process sequences of variable sizes, Recurrent Neural Networks (RNNs) have been widely used for text modeling and generation tasks. They are composed of an encoder that processes an input sequence by updating a representation in $\mathbb{R}^n$, and a decoder that generates the target sequence from the last computed representation. Some works have adapted these ideas to a sequence of queries (Dehghani et al., 2017; Jiang and Wang, 2018; Sordoni et al., 2015).

3.2.3.1 HRED

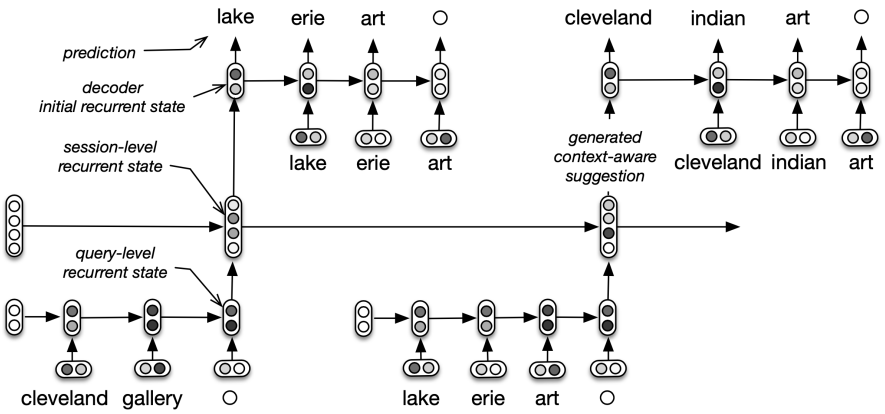


Figure 3.1: HRED architecture

The model HRED (Sordoni et al., 2015), illustrated Figure 3.1, proposes to use two encoders: a query-level encoder GRU_{enc}, a session-level

encoder GRU_{ses} and a decoder GRU_{dec}. GRU_{enc} encodes each query of the user session independently, the representation of the n-th word of the i-th query is noted $h_{i,n}$. While GRU_{ses}, the session-level encoder, deals with the sequence of query representations, it takes as input the query encoding and updates its own recurrent state, the representation of the i-th query is noted s_i. This session-level encoder should learn a summary of the user history so that its output is used as input by GRU_{dec} to predict the next query. When computing the probability of the $n+1$-th word of the i-th query, the decoder uses its recurrent state $d_{i,n}$.

The hidden states of the three networks GRU_{enc}, GRU_{ses} and GRU_{dec} are respectively:

$$\begin{aligned} h_{i,n} &= GRU_{enc}(h_{i,n-1}, w_{i,n}), \quad n = 1, \dots, |Q_i| \\ s_i &= GRU_{ses}(s_{i-1}, h_{i,|Q_i|}), \quad i = 1, \dots, |S| \\ d_{i,n} &= GRU_{dec}(d_{i,n-1}, w_{i,n}), \quad n = 1, \dots, |Q_i| \end{aligned} \tag{3.2}$$

The networks parameters are learned by maximizing the log-likelihood of a session S:

$$\begin{aligned} \mathcal{L}(S) &= \sum_{i=1}^{|S|} \log P(Q_i \mid Q_{1:i-1}) \\ &= \sum_{i=1}^{|S|} \sum_{n=1}^{|Q_i|} \log P(w_{i,n} \mid w_{i,1:n-1}, Q_{1:i-1}) \end{aligned} \tag{3.3}$$

This architecture has the advantage of taking into account the hierarchical structure of the data by encoding each query before integrating the information at the session level. It also allows to have a representation of the user for each new query via the session-level encoder GRU_{ses}. On the other hand, the tokenization of word queries does not allow the integration of out-of-vocabulary words (OOV), which are common in IR, especially for complex searches.

3.2.3.2 ACG

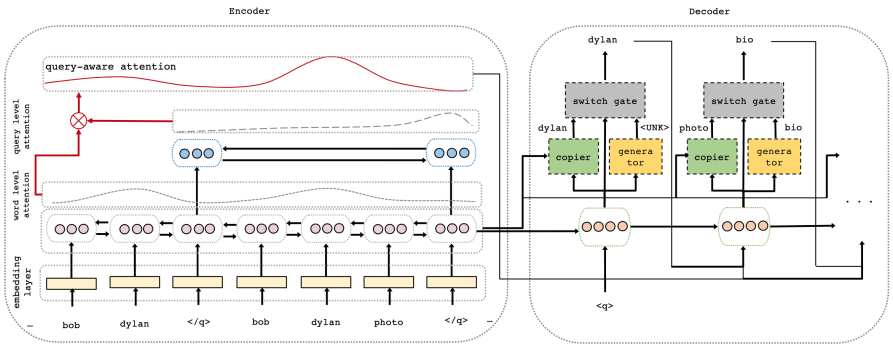


Figure 3.2: ACG architecture

Contrary to HRED which uses hierarchical representations, ACG (Dehghani et al., 2017) encodes all the queries with a RNN at the same time by concatenating them as a long sentence. However, to keep the notion of queries sequences it relies on a hierarchical attention process and a copy mechanism.

The attention mechanism has two levels of attention, a word-level one and a query-level one. The word-level attention attributes weights to the hidden states of the encoder which correspond to each word of the input queries. The query-level attention is calculated according to a query-level encoder. This second encoder takes as input the hidden state of the last word of each query. It is used to calculate the attention given to each query in the session. The two attention mechanisms are then combined to get a weight for each input token at each decoding step.

The authors assume that successive queries in sessions are reformulations that usually share words. Thus, they introduce a copy mechanism able to pick tokens from past user queries rather than generating them using a fixed-size vocabulary. It also allows the model to deal with Out-Of-Vocabulary (OOV) words if they have been used in the session before. This mechanism is made of a switch gate that decides if the next word should be copied or generated and of a copier that attributes a probability for each input word to be copied.

3.2.4 Feedback

Other RNN-based approaches have also been recently proposed, such as (Wu et al., 2018a), which leverages user clicks and document representations to specify the user intent (Ahmad, Chang, and Wang, 2018; Ahmad, Chang, and Wang, 2019), or (Jiang and Wang, 2018) which integrates click-through data into query embeddings to capture semantic reformulations. Some works have explored the use of long-term search history of users (Chen et al., 2018), using a RNN-based hierarchical architecture, to score query suggestions. In this thesis, we restrict to queries in sessions as input data, but other sources of information can be added to such models.

3.3 Transformers for query suggestion

This thesis proposes a thorough analysis of transformers for query prediction. Many variants of the architecture have been proposed. This section first describes how transformers, called here "flat" transformers can be used for this task. We also consider hierarchical transformers that might leverage the session structure (as HRED). In addition to their architecture based on powerful attention mechanisms, transformers have enabled leaps in NLP thanks to their multi-task pretraining on

large corpora. Several pre-trained models are described in the last part of this section.

3.3.1 Flat transformer

Let us first formalize how the classical transformers of Vaswani et al. (2017) can be directly adapted to our task. The presented model is illustrated by Figure 3.3.

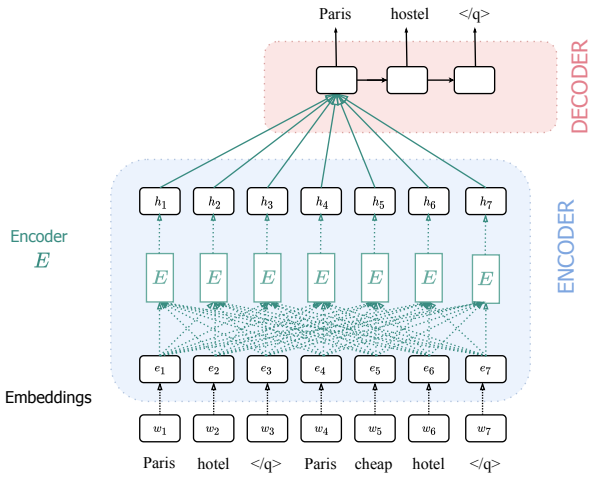


Figure 3.3: Flat Transformer for Query Suggestion

Input For a session, the input of the transformer is simply the concatenation of all the words of all the queries separated by a token [SEP], i. e. the [SEP] is used to mark the beginning of a new query in the session:

$$S = [[SEP] \underbrace{w_{1,1} \dots w_{1,|Q_1|}}_{Q_1}[SEP] \dots [SEP] \underbrace{w_{|S|,1} \dots w_{|S|,|Q_{|S|}|}}_{Q_{|S|}}[SEP]]$$

This sequence is then transformed by using the token embeddings added to positional embeddings (one per distinct position) — this is how transformers recover the sequence order (Vaswani et al., 2017).

The encoder E gives a contextualized representation for each token of the session:

$$\mathrm{E}(S) = (h_0, \dots, h_n) \tag{3.4}$$

where n is the number of tokens in the whole session: $n = \sum_i |Q_i|$.

We train models with various encoders E described in the next sections (in Section 3.4.2 and Section 3.4.3). The decoding part is the same for all and has been described in Section 2.1.4.

3.3.2 Hierarchical transformer

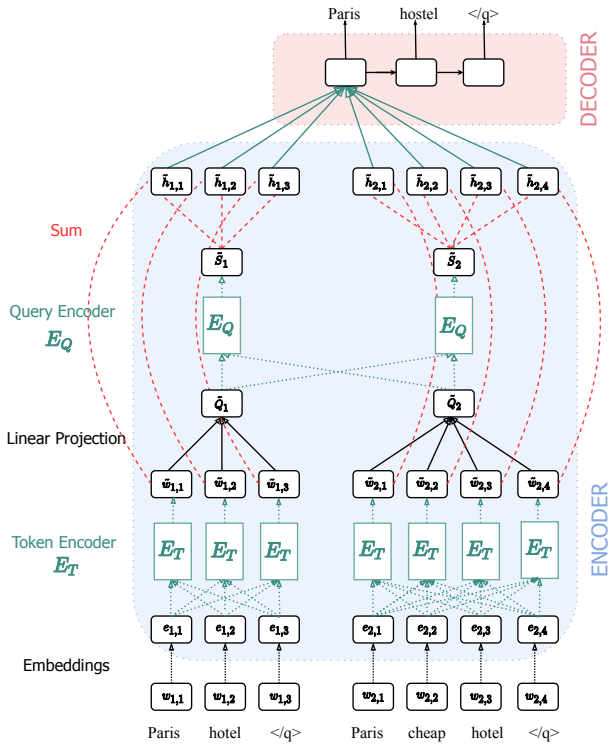


Figure 3.4: Hierarchical Transformer for Query Suggestion

For the query suggestion task, Garg, Dhillon, and Yu (2019) present a hierarchical transformer that outperforms RNN-based model, and thus show that recurrence is not crucial for the task. An illustration is given in Figure 3.4. Their model is composed of two levels of encoding: a token-level E_T and a query-level one E_Q, each following the same contextualization process as a standard encoder in a transformer model. The first encoder gives a contextualized representation of each token that depends on the other tokens of the query, while the second one outputs a contextualized representation of each query depending on the other queries of the session. Our work extends this paper by providing a thorough analysis of the behavior of (hierarchical) transformer models, as well as experimenting with various pre-trained transformer models.

First, the token-level Encoder E_T produces a contextualized representation $\mathrm{E}_T(Q_i) = (\tilde{w}_{i,1}, \dots, \tilde{w}_{i,K})$ of each token of a given query Q_i. Since queries might have a different length, padding is used (e. g. a special [BLANK] token), so that each query is of length K. This representation is then summarized into a query representation $\tilde{Q}_i$ using a linear transformation:

$$\tilde{Q}_i = \mathrm{E}_T(Q_i)\mathrm{W}_P \tag{3.5}$$

The transformation matrix is $W_P \in \mathbb{R}^{K \times d}$ where d is the output dimension of each token of the encoder. In our experiments we use $K = 12$, which is enough to cater for most of the queries of our dataset – the remaining tokens are truncated.

The session-level encoder takes these vectors $\tilde{Q}_i$ as input to transform them into final query representations $\tilde{S} = (\tilde{S}_1, \dots, \tilde{S}_{|S|})$ that embed context from neighbor queries, using positional encoding following Vaswani et al. (2017).

$$\tilde{S} = E_Q\left(\tilde{Q}_1, \dots, \tilde{Q}_{|S|}\right) \tag{3.6}$$

where $|S|$ is the number of queries in the session.

We then obtain the final representation of a query token by summing its query-wise representation $\tilde{w}_{i,j}$ with the contextualized representation of its corresponding query $\tilde{S}_i$:

$$\tilde{h}_{i,j} = \tilde{w}_{i,j} + \tilde{S}_i \tag{3.7}$$

Finally, the decoding part is exactly the same as for other transformer models (Section 2.1.4).

3.4 Experimental settings and results

In this section, we report experimental results comparing the various flat and hierarchical transformer-based models, as well as other baselines.

We first describe the datasets, the compared models, and the metrics (Sections 3.4.1 to 3.4.4), before presenting our main results in Section 3.4.5. In Section 3.4.6, we present some queries generated by a selection of models. Finally, in Section 3.5.1, we pursue our analysis by studying how the models perform when exposed to noise, by altering the sessions (filtering or concatenating). In both cases, we show that hierarchy does not help as much as good pretraining.

3.4.1 Datasets

Some datasets allow a fine evaluation of query suggestions. They consist of queries grouped by user sessions and associated with relevant documents. These datasets are the TREC Session dataset (Carterette et al., 2016) which contains the names of the tasks and relevant documents associated with the user sessions, the conversational dataset SCSdata (Trippas et al., 2020) segmented by task and containing the documents read by the user, and the Webis-SMC-12 dataset (Hagen et al., 2013) which is a subset of AOL for which the sessions have been manually split and annotated into missions. However, these three datasets contain a few sessions, respectively 1300, 1000, and 2200 sessions, which is insufficient to train the models we want to compare.

To the best of our knowledge, there is no dataset of sufficient size better suited to the task of suggesting queries than the two query logs datasets: the real dataset AOL web search log and the artificial dataset, MS Marco Conversational Search (Nguyen et al., 2016). In both cases, the queries are processed by removing all non-alphanumeric characters and lowercasing following Sordoni et al. (2015).

MS Marco is an artificial dataset, built from real queries. The authors filtered queries by removing navigation, bot, junk, and adult sessions and merged users' queries with a nearest-neighbor search based on their embeddings to create artificial sessions. The MS Marco dataset is provided in two parts. We use 80% of the first part as the training set, the remaining 20% as the validation set, and the second part of the dataset as the test set. Each set contains respectively 540 267, 135 066, and 75 193 sessions.

The AOL dataset consists of 16 million real search log entries from the AOL Web Search Engine for 657,426 users. Following Sordoni et al. (2015), we delimit sessions using a 30-minutes timeout for both datasets. The queries submitted before May 1, 2006, are used as the training set, the remaining four weeks are split into validation and test sets, as in (Sordoni et al., 2015). After filtering, there are 1 708 224 sessions in the training set, 416 450 in the test set, and 416 450 in the validation set. As the real-world AOL dataset is not filtered, it contains typos and noisy sessions. It is made of 860 155 unique words, whereas the artificial dataset MS Marco has 28 968. Sordoni et al. (2015) build a vocabulary by listing the 90k most common words in the training set. For AOL, it is counted that 8.9% of the words from the dataset are not in the vocabulary. Whereas when building a vocabulary in the same way with MS Marco, all the words in the dataset are in the vocabulary.

3.4.2 Transformer trained from scratch (TS)

When the encoder and decoder parameters are learned from scratch, these models are designated as fully trained transformer (TS), in opposition to the pre-trained transformers discussed in the next section. TS architecture is described in 2.1.4, with $L_d = 6$ layers, with $H = 12$ heads each and a dropout $p = 0.1$. On the top of the decoder, we use a feedforward network with a hidden size of 2048. For the input tokens, we use the same embeddings for the encoder and the decoder to reduce the number of parameters and to regularize the network following Vaswani et al. (2017).

In the following, we experiment with three different pre-trained models. (1) BERT because it is the most used transformer, (2) BART because it has an encoder-decoder architecture with very good performance in generation, and especially in summarization, and finally (3) T5 because it was one of the last transformers that has been published at the time of writing.

3.4.3 Compared models

The models compared are the following:

3.4.3.1 Non-tranformer models

Inverted Index We use a co-occurence based method as a baseline: the Inverted Index (Broccolo et al., 2012) described in Section 3.2.2.

RNN The RNN models are HRED (Sordoni et al., 2015) and ACG (Dehghani et al., 2017), which we described in Section 3.2.

3.4.3.2 Fully trained transformer (TS)

The fully trained transformer, hereafter referred to as TS, is composed of an encoder and a decoder presented in Section 3.3.

3.4.3.3 Pre-trained transformers

The pretrained models that we fine-tune are BERT (Devlin et al., 2019), BART (Lewis et al., 2020) and T5 (Raffel et al., 2020). To leverage these pre-trained models, which is especially important since the number of parameters in transformer models is high, we use their parameters to initialize the parameters of our corresponding models. More precisely, for the flat architecture, the encoder parameters are either initialized to those of the BERT model, the BART or T5 encoder. The models are named respectively BERT, ENCBART and ENCT5. Since BART and T5 are not only an encoder as BERT, we also consider a version with both encoder and decoder parameters initialized with pre-trained BART and T5 parameters, that we refer respectively to BART and T5.

BERT We use the pre-trained model BERT (Devlin et al., 2019), and extract each hidden layer of the model. We sum the last layer, with the average and the max of these layers based on CLIP-as-service library [1], and our own preliminary experiments. The intuition behind the use of multiple hidden layers is that the last layer might be too close to the specific tasks on which BERT was trained, while deeper layers could be more representative of the general meaning of words. Thus, for each token of the input, we have a contextualized embedding of size 768 given by BERT. For the decoding part, we use the same transformer decoder and feedforward network as the ones described in 3.4.2. At the beginning of the training the encoder is frozen and the decoder is trained. We then use a "gradual unfreezing" of the encoder layers as recommended by Howard and Ruder (2018): when the loss stabilizes,

1 https://github.com/hanxiao/bert-as-service

we unfreeze the last frozen layer of the encoder, until all the layers are fine-tuned.

BART and ENCBART BART architecture is made of both an encoder and a decoder. We also use gradual unfreezing to fine-tune the model but starting from the last layer of the pre-trained decoder. We analyze the results of the complete BART model fine-tuned for our task, but also the ones of the BART encoder ENCBART followed by a fully trained transformer decoder. This allows a fair comparison with the BERT model which has no decoder, and a measure of benefits of the pre-trained decoder.

T5 and ENCT5 T5 is a transformer with a pre-trained encoder and a pre-trained decoder. As we did for BART, we compare two versions of the model: T5, a version for which we fine-tune the entire pre-trained model, and ENCT5 the encoder-only version, with a fine-tuned encoder and a decoder trained from scratch. We use the training protocol described for BERT and BART.

3.4.3.4 Hierarchical transformers

H_TS The hierarchical transformer H_TS with the two-level encoder described in Section 3.3.2.

H_BERT, H_BART and H_T5 We also compare hierarchical architectures (H_TS) for which the Query Encoder E_T parameters are initialized with those from BERT, BART and T5 encoders, and the rest of the architecture remaining are trained from scratch. We refer to such models as H_BERT, H_BART and H_T5.

Tokenizer The two RNN-based models and the fully trained transformers TS and H_TS use a fixed vocabulary composed of words, but BERT, BART and T5 employ subword tokenizers, denoted as word piece tokenizer (WPT), that segment the text into n-grams of varying lengths (Sennrich, Haddow, and Birch, 2016). For instance, the query "Robert Mitchum" is segmented as `[robert, [UNK]]` with a word tokenizer while the WPT returns `[robert, mitch, ##um]`. Hence, for the latter, there is no out-of-vocabulary problem which was handled inadequately with the word tokenizer using the special OOV token. Besides, the vocabulary size is kept below a predefined threshold (31k tokens for BERT, 32k for T5, and 50k for BART), which in turns speeds up learning.

To analyze the importance of the tokenizer, we consider variants of HRED, ACG, TS and H_TS based on the BERT tokenizer, named HRED_wp, ACG_wp, TS_wp and H_TS_wp.

Training For all models involving pre-trained transformers, the training procedure is the same: we use the "gradual unfreezing" method, as recommended by (Howard and Ruder, 2018) and described in 3.4.3.3. Models optimization is performed on the training sets of sessions with the ADAM optimizer (Kingma and Ba, 2015). All hyper-parameters are tuned via grid search on a validation dataset.

3.4.4 Metrics

As with many other tasks in IR, evaluating the quality of the models is problematic since the evaluated models can generate arbitrary queries in response to a session — and there is no principled way to evaluate their quality. In the following, we describe the metrics that were reported in previous works to compare models, and which try to capture the quality of the system responses:

- **Perplexity**. All compared models generate probability distributions over the sequences. This enables to check how surprised the model is by the target query. However, the perplexities of some pairs of models cannot be compared because the vocabulary size is different (90k tokens for models without a WPT, 31k tokens with WPT, 50k for BART's tokenizer, and 32k for T5). An option would be to normalize the sentence likelihood by the number of characters, but it would have made the results less interpretable. Moreover, former versions of HRED, ACG, TS, and H_TS can generate OOV words, which strongly biases the results. Thus, perplexity is not reported for these last methods.

- **BLEU**. As a metric to evaluate generated queries compared to the target ones, we first use the classical metric BLEU (Papineni et al., 2002), which corresponds to the rate of generated n-grams that are present in the target query. We refer to BLEU-1, BLEU-2, BLEU-3 and BLEU-4 for 1-gram, 2-grams, 3-grams and 4-grams respectively.

- **EM**. We also calculate the exact match EM, which is equal to 1 if the predicted query is exactly the observed one, and 0 otherwise.

- **Sim**$_{extrema}$. As EM can be too harsh, we also use a metric, Sim$_{extrema}$ (Forgues et al., 2014), which computes the cosine similarity between the representation of the candidate query with the target one. The representation of a query q (either target or generated) is a component-wise maximum of the representations of the words making up the query (we use the GoogleNews embeddings, following (Sordoni et al., 2015)). The extrema vector method has the advantage of taking into account words carrying information, instead of other common words of the queries.

- **Sim**$_{pairwise}$. However, this component-wise maximum method might excessively degrade the representation of a query. As an alternative, we propose to compute Sim$_{pairwise}$ as the mean value of the maximum cosine similarity between each term of the target query and all the terms of the generated one.

- **New Words**. We also want to have models that can suggest words and queries that have not yet been used during the session. We use the ratio of new words, calculated by counting the number of unique words that appear in the suggested query but were not in the past queries of the session, divided by the count of unique words in this query. This metric is an indication of the model's ability to suggest new terms. The higher, the more new words.

- **Repetition Rank**. A model that suggests queries that have already been issued by the user is pointless. To evaluate this, we introduce the Repetition Rank metric, which computes the rank of the prediction in the beam search if the predicted query appears in the context (or 10 if it doesn't). The lower, the more repetitive.

Finally, as discussed in Section 3.3, there is no ground truth on what the best queries to suggest are. For each generation metric, we consider the maximum performance of the top 10 queries generated by the models. More precisely, for each model, we first generate (through a beam search with $K = 20$) 10 queries to suggest to the user given the context [2] The reported value for each metric (BLEU, EM, Sim$_{extrema}$ and Sim$_{pairwise}$) is the maximum score over the 10 different generated queries. This is usually employed for assessing the performance of a probabilistic model w.r.t. a single target (see e. g. Kumar et al. (2020)) and corresponds to a fair evaluation of models that try to find a good balance between quality and diversity.

3.4.5 Query Suggestion Performance

In this section, we aim to answer our first question: **Q1. How well the various presented transformers generate queries suggestions compared to the usual baselines?**

Tables 3.1 (generation scores), and 3.2 (perplexity) report results obtained by all the models.

We first note the difference between the two datasets. As expected, being synthetic, MS Marco is a much easier dataset — more restricted vocabulary and more regular sessions, as acknowledged by the fact that all the metrics are higher for MS Marco.

2 As we want to encourage the models trained with a word tokenizer to generate tokens present in the vocabulary, we follow (Kai, Hirose, and Nakagawa, 1998) and apply a penalty on the "OOV" token in the beam search.

Table 3.1: Results on the MS Marco (a) and the AOL dataset (b). We report different metrics, along with two quality indicators. Best results for a metric are reported with a bold font.

	II	Acg	Acg_wp	Hred	Hred_wp	TS	TS_wp	H_TS	H_TS_wp	Bert	H_Bert	EncBart	Bart	H_Bart	EncT5	T5	H_T5
EM	0.173	0.044	0.041	0.139	0.129	0.174	0.197	0.164	0.170	**0.223**	0.182	0.184	**0.226**	0.183	0.175	0.203	0.121
BLEU 1	0.584	0.435	0.416	0.572	0.555	0.579	0.596	0.574	0.589	**0.617**	0.597	0.591	**0.618**	0.592	0.598	0.576	0.565
BLEU 2	0.369	0.200	0.182	0.341	0.320	0.372	0.377	0.363	0.371	0.402	0.378	0.385	**0.419**	0.383	0.379	0.375	0.335
BLEU 3	0.218	0.092	0.087	0.193	0.176	0.223	0.248	0.218	0.224	**0.274**	0.234	0.238	**0.275**	0.236	0.230	0.238	0.174
BLEU 4	0.202	0.073	0.068	0.175	0.161	0.213	0.239	0.201	0.206	**0.268**	0.217	0.222	**0.266**	0.221	0.212	0.231	0.149
$sim_{extrema}$	0.835	0.798	0.780	0.828	0.817	0.833	0.840	0.834	0.837	**0.846**	0.839	0.837	**0.848**	0.839	0.837	0.837	0.830
$sim_{pairwise}$	0.677	0.579	0.543	0.635	0.616	0.671	0.682	0.665	0.670	**0.697**	0.677	0.672	**0.697**	0.678	0.675	0.659	0.661
New Words	0.950	0.138	0.354	0.594	0.604	0.886	0.880	0.902	0.899	0.870	0.902	0.902	0.858	0.911	0.879	0.910	0.895
Repetition Rank	8.618	8.767	9.429	8.974	9.141	6.926	6.689	7.055	7.022	6.424	6.755	6.985	5.586	7.098	7.116	6.913	7.318

(a) MS Marco dataset

	II	Acg	Acg_wp	Hred	Hred_wp	TS	TS_wp	H_TS	H_TS_wp	Bert	H_Bert	EncBart	Bart	H_Bart	EncT5	T5	H_T5
EM	0.018	0.017	0.010	0.029	0.036	0.037	0.048	0.046	0.081	0.061	0.085	0.055	**0.119**	0.087	0.052	0.082	0.053
BLEU 1	0.438	0.417	0.388	0.409	0.422	0.439	0.454	0.447	0.493	0.460	0.495	0.455	**0.552**	0.494	0.452	0.519	0.435
BLEU 2	0.148	0.128	0.098	0.122	0.135	0.162	0.178	0.178	0.238	0.194	0.241	0.186	**0.316**	0.240	0.183	0.275	0.166
BLEU 3	0.067	0.037	0.026	0.052	0.059	0.071	0.089	0.102	0.146	0.110	0.150	0.104	**0.231**	0.144	0.098	0.192	0.090
BLEU 4	0.033	0.006	0.004	0.018	0.023	0.027	0.040	0.055	0.086	0.063	0.093	0.058	**0.174**	0.084	0.051	0.148	0.043
$sim_{extrema}$	0.751	0.668	0.687	0.710	0.713	0.729	0.723	0.742	0.762	0.741	0.763	0.739	**0.792**	0.762	0.731	0.776	0.723
$sim_{pairwise}$	0.484	0.408	0.390	0.404	0.415	0.447	0.457	0.462	0.501	0.466	0.504	0.459	**0.558**	0.499	0.454	0.537	0.435
New Words	0.996	0.119	0.588	0.679	0.740	0.916	0.941	0.849	0.881	0.927	0.880	0.919	0.682	0.934	0.902	0.593	0.940
Repetition Rank	9.711	7.138	9.128	7.841	7.157	8.683	8.300	6.830	4.970	6.668	4.203	6.132	2.204	3.665	6.203	1.468	6.324

(b) AOL dataset

From a high-level point of view, we see that transformers are better performing than the baseline Inverted Index (II) and that the RNN-based models, Hred and Acg. Among transformers, more complex and pre-trained models perform better, with the flat architecture with a pre-trained encoder and decoder Bart performing the best. Contrarily to (Garg, Dhillon, and Yu, 2019), we do not observe a real difference between hierarchical and non-hierarchical Transformer architectures: The main factor of variation is on what task and dataset the model was pretrained.

We note that models have different tendencies to copy one of the queries in the session. This is a standard behavior: 3% of queries for MS Marco and 6% for AOL are among the previous queries of the session. So it is not surprising that more powerful models learn to copy — transformer models have a tendency to repeat a seen query compared to Acg or Hred (lower Repetition Rank). We explain this tendency by their ability to retrieve information at arbitrary positions in the input.

Perplexity We only compare perplexity for models based on the same tokenizer, since otherwise the problem of evaluating prediction with OOV tokens, or of vocabulary with different sizes makes comparisons impossible. We observe that the transformers obtain a much better perplexity than Acg and Hred with a Word Piece Tokenizer. The likeli-

Table 3.2: Perplexities for Word-Piece Tokenizer-based models

	Acg_wp	Hred_wp	TS_wp	H_TS_wp	Bert	H_Bert	EncBart	H_Bart	Bart	EncT5	H_T5	T5
AOL	1 175	1 101	721	486	492	473	557	209	173	92	215	37
MS Marco	242	111	56	56	47	64	52	40	39	22	58	21

hoods of target queries with these last two methods are both about half the one of the transformer model TS_wp. This shows that transformers better explain users' behavior in search sessions. Among transformers, we observe that while the hierarchy is beneficial on the AOL dataset, it is not the case on the MS Marco dataset. We discuss this behavior in more detail later.

Word Piece Tokenizer Among RNNs, using a WPT is sometimes beneficial for Hred but not for Acg. We explain this because the copy mechanism already allows Acg to produce rare tokens. This ability appears lowered when using word pieces, as assembling unknown words from smaller tokens is much more difficult than copying a whole word for such architectures. For Hred, the Word Piece Tokenizer improves the scores on the AOL Dataset, while it degrades them on the MS Marco one. This is explained by the fact that for the MS Marco dataset, there is no OOV and hence using a WPT is not useful anymore.

For transformers trained from scratch (TS, TS_wp, H_TS and H_TS_wp), the Word Piece tokenizer is always beneficial. It could be due to the use of positional embeddings, which makes the copy of consecutive tokens easier. Moreover, the use of this tokenizer reduces the vocabulary size.

Pretrained models First, Bart (flat transformers with a pre-trained encoder and decoder) outperforms all the models on all metrics. This shows the value of pre-trained models on large datasets and on generative tasks. When observing the flat pre-trained models scores, we note that they outperform the version trained from scratch: Bert, EncBart, Bart, EncT5 and T5 are better than TS_wp on the AOL dataset.

For the MS Marco dataset, while Bert and Bart have better scores than TS_wp, EncBart and EncT5 are similar to TS_wp. We think that because the vocabulary used in the MS Marco dataset is more restricted, and the dataset more regular, the use of large pre-trained models is less beneficial. While T5 largely outperforms Bert on the AOL dataset, Bert is much better than T5 on the MS Marco dataset. The unified framework — consisting of training simultaneously the model for various tasks — used to pretrain T5 is useful on a complex dataset, as it probably allows the model to acquire more language knowledge, but it is less efficient on simpler data. Finally, for both datasets, Bart performs the best for all metrics.

On the AOL dataset, BART improvement is particularly important on BLEU 3 and BLEU 4 — which are calculated by considering 3-gram and 4-gram sequences. It indicates that when comparing longer word sequences between target and predictions, BART is the best model, it is better at generating longer queries. We think this is because BART has been trained on a summarization task, it performs better than the other models at generating comprehensive sequences. Its similarity scores $sim_{extrema}$ and $sim_{pairwise}$ are also significantly better on the AOL dataset, which means that TS is the best model to capture the word semantic.

The Hierarchy On the AOL dataset, the hierarchical models perform better than their flat version: TS vs H_TS, TS_wp vs H_TS_wp, BERT vs H_BERT, ENCBART vs H_BART except for T5 for which ENCT5 outperforms H_T5. This could be due to the fact that T5 uses relative positional embeddings, while other models use absolute positional embeddings. H_T5 would have more difficulties to find the exact position of words within queries. Note that for fair comparison H_BART and H_T5 are compared to ENCBART and ENCT5 rather than BART and T5 because BART and T5 decoders are pretrained while H_BART and H_T5 decoders are trained from scratch. This shows that with a suitable encoder the hierarchy is beneficial for the query suggestion task, the two-levels encoder allowing to have a more complex representation of the session.

The conclusions are different for the MS Marco dataset. For the fully trained model TS and TS_wp, and for BART, the hierarchy does not help significantly, while with BERT and T5, the hierarchy decreases the results. We explain this because the queries and the sessions of the MS Marco dataset are longer, and the model has difficulty to focus its attention on the important queries. We discuss the behavior of the hierarchical models on longer and more complex sessions more in detail below.

3.4.6 Generated queries

In Table 3.3, we give examples of query suggestions for three sessions, and multiple models: HRED_wp(which is the best among the RNN baselines), the fully trained transformers TS_wp and H_TS_wp and the pre-trained ones ENCBART, H_BART, and BART.

First, we note that the RNN-based model HRED_wp generates the same word several times in a row. This behavior is very common for HRED_wp. For the session presented in the first column, it suggests "divorce groups groups", for the second "maryland hotel hotel ocean" and for the third "disney resorts resorts". Note that this is something the transformer models never do. Moreover, HRED_wp doesn't introduce new words, it reformulates the queries of the context by mixing words

Table 3.3: Generated queries for three sessions. The two first queries of the sessions are given in the top of the table (Q1 and Q2), and the first 5 suggestions of each model are reported below.

	Q1. divorce chat rooms Q2. divorce support groups	Q1. maryland ocean city Q2. marylandocean vity hotel	Q1. carobean cruises Q2. spa resorts Q3. disney world
HRED_wp	- divorce support groups	- maryland hotel hotel	- disney world resorts
	- divorce chat groups	- maryland hotel ocean	- disney resorts resorts
	- divorce divorce groups	- maryland hotel hotel ocean	- disney world
	- divorce groups groups	- maryland hotel	- disney vacation resorts
	- divorce support	- maryland hotel ocean ocean	- disney resorts
TS_wp	- chat room listings	- ocean city maryland	- disney world
	- ebay	- ocean city md	- travelocity
	- aol chat	- mapquest	- disney world hotels
	- chat rooms	- ocean county maryland	- disney world cruise
	- divorce chat room	- expedia	- disney cruise
H_TS_wp	- divorce support groups	- maryland ocean city	- disney world
	- free divorce support groups	- ocean city maryland	- sea world
	- divorce	- hotels in maryland	- disneyworld
	- divorce chat rooms	- hotel ocean city	- carnival cruise
	- divorce support	- mapquest	- spa resorts
ENCBART	- divorce chat rooms	- maryland hotel	- disney world
	- divorce chat room	- maryland hotels	- disney world cruises
	- divorce support group	- mapquest	- disney world texas
	- divorce support	- maryland	- disney world hotels
	- divorce chat	- maryland beach hotel	- disney world resort
H_BART	- divorce support groups	- maryland ocean city	- disney world
	- divorce	- marriott hotels	- spa resorts
	- free divorce chat rooms	- marylando ocean city	- disney world cruise
	- divorce help	- marriott	- disney world resorts
	- free divorce help	- mapquest	- ebay
BART	- divorce chat rooms	- maryland ocean city hotel	- disney world
	- divorce support groups	- maryland ocean city	- spa resorts
	- free divorce support groups	- maryland ocean city hotels	- disneyworld
	- divorce chat room	- maryland ocean town hotel	- disney world cruise
	- free divorce chat rooms	- maryland ocean city resort	- disney world hotels

order. On the contrary, the transformer models propose more diverse suggestions.

We note that the hierarchical models have a greater tendency to copy words from the context compared to their flat version (we study this behavior in the next section). H_TS introduces only one new word

Table 3.4: Human evaluation on 100 queries for MS Marco and AOL. Each cell is the % of times model in row is better than model in column vs the reverse (and the remaining % is equality)

	HRED	HRED_WP	ACG	ACG_WP	TS	TS_WP	BERT
				MS Marco			
HRED_WP	19% vs 18%						
ACG	26% vs 29%	22% vs 22%					
ACG_WP	20% vs 22%	17% vs 21%	22% vs 24%				
TS	32% vs 11%	33% vs 13%	38% vs 16%	36% vs 13%			
TS_WP	37% vs 10%	35% vs 10%	42% vs 15%	39% vs 11%	15% vs 10%		
BERT	41% vs 10%	38% vs 8%	42% vs 15%	43% vs 11%	25% vs 15%	22% vs 18%	
BART	43% vs 9%	42% vs 11%	45% vs 11%	44% vs 9%	27% vs 15%	21% vs 16%	19% vs 16%
				AOL			
HRED_WP	23% vs 16%						
ACG	13% vs 24%	10% vs 29%					
ACG_WP	4% vs 24%	6% vs 32%	7% vs 15%				
TS	34% vs 17%	31% vs 20%	35% vs 3%	43% vs 5%			
TS_WP	28% vs 15%	24% vs 18%	32% vs 5%	38% vs 5%	13% vs 18%		
BERT	34% vs 13%	31% vs 18%	41% vs 9%	44% vs 6%	28% vs 24%	28% vs 19%	
BART	38% vs 17%	35% vs 20%	41% vs 11%	45% vs 8%	30% vs 28%	31% vs 24%	26% vs 24%

("free") in the suggestions of the first session, while TS_wp proposes several new themes ("listings", "ebay", "aol"). The second presented session contains a typo: "marylandocean" instead of "maryland ocean" with a blank space. The hierarchical H_BART didn't succeed to correct this typo, it proposes "marylando ocean city" because it is more willing to copy words from the context, and thus a part of this typo, while the flat transformer models didn't.

The pre-trained models BART and H_BART propose more diverse suggestions compared to the fully trained models. In the session of the third column, the user performs queries on several topics of the same subject. While the various models succeed in integrating the diverse themes in the suggestions, the pre-trained models introduced more new topics: "texas", "hotels", "ebay". Finally, we notice that the suggestions of BART tend to be longer than the ones of the other models, confirming the experimental results shown earlier.

3.4.7 Human evaluation

To further investigate the ability of the flat models, we conducted a human evaluation by comparing 100 queries predicted for AOL and MS Marco by all the models. The judges were presented with complete sessions and corresponding suggestions predicted by each model. They had no knowledge of the ground truth or the user's goal. In our user modeling framework, we seek to evaluate whether suggestions make

sense to annotators based on the user's session, not only whether they are syntactically correct. That's why judges were asked to evaluate the suggestions that were most likely to meet the user's need in the session by answering the question "is this query likely to follow in the session?". They were asked to rank the predictions from most to least suitable. Annotators are supposed to be able to infer the user's purpose from the session. Indeed, no more can be expected from an optimal policy that only has the user session at its disposal, and this is what we are trying to assess. Giving the user's purpose to the annotators could have biased the evaluation by leading the annotators to evaluate too negatively many suggestions, even though they corresponded to average user behavior. We further asked the annotator to rank exact repetitions and generic queries (e. g. "google") as bad predictions. We report in Table 3.4 the % of times a model is judged better than another one.

The evaluation confirms the results obtained with the other metrics. The models ACG, HRED and the different transformers are increasingly better (e. g. on AOL, 27% of predicted queries are better for BART than for HRED, and 17% for the other way around). Among transformers, pre-trained models perform better (5-10% gap), with BART doing slightly better than BERT. Regarding Word Piece tokenization, they do perform better except for transformer on AOL, and for ACG.

3.5 Analysis of transformer for query suggestion

This section is an in-depth analysis of transformers in response to questions Q2 and Q3.

3.5.1 Robustness of the transformer models

We now look in more detail at how the models behave regarding different types of sessions to answer the second question **Q2: Which model is the most robust to complex sessions (a), to noisy sessions (b), and to long sessions (c)?** For each type of session, a section is dedicated to the answer.

3.5.1.1 Results on complex sessions

Focusing on the real-world dataset AOL, which contains many very short and simple search sessions typical of web search, we were interested in how transformer models could handle complex sessions. To identify those, we used a simple heuristic: a complex session (1) consists of at least three queries; (2) contains queries with more than

one word; and (3) should not contain spelling corrections. For (3), we used the following heuristic: each of its queries must be sufficiently different from the previous one, i. e. its editing distance (in characters) should be greater than 3.

Figure 3.5a reports the relative results obtained on this subset of 193 336 complex sessions. In particular, we want to compare the results of the flat and of the hierarchical models.

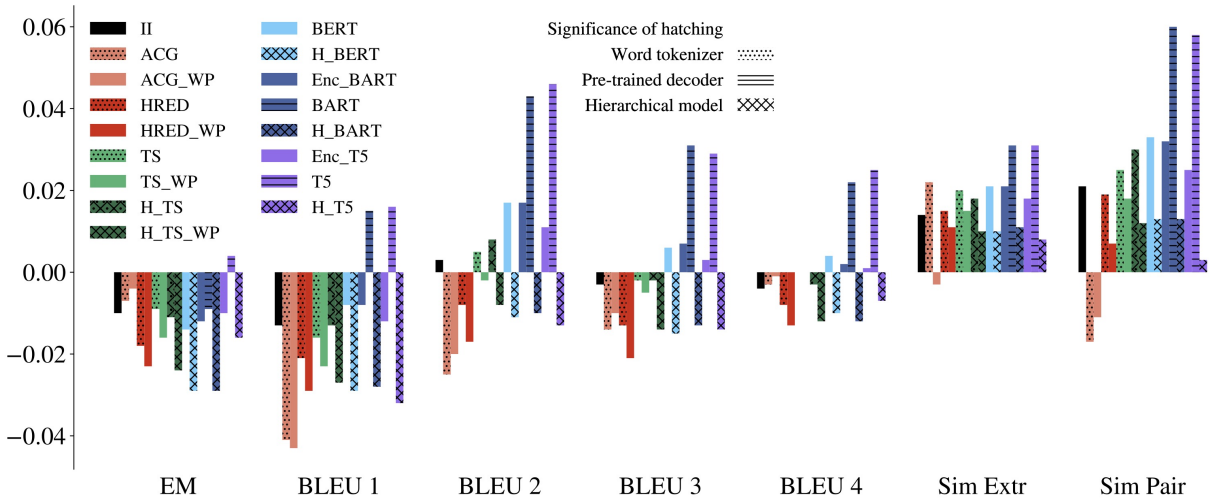
We note the good behavior of pre-trained flat transformers for query suggestions for the complex search task, while it emphasizes the weakness of the pre-trained hierarchical models on these sessions. The flat models improve the results on these sessions over the corresponding hierarchical model on all metrics: BERT is less deteriorated than H_BERT, and likewise, BART and ENCBART are less impacted than H_BART by the complexity of the sessions, and the same is true for T5 models. For the fully trained models, TS_wp is also less impacted than H_TS_wp on this subset of sessions on all metrics. This shows again the robustness of flat models.

3.5.1.2 Results on noisy sessions

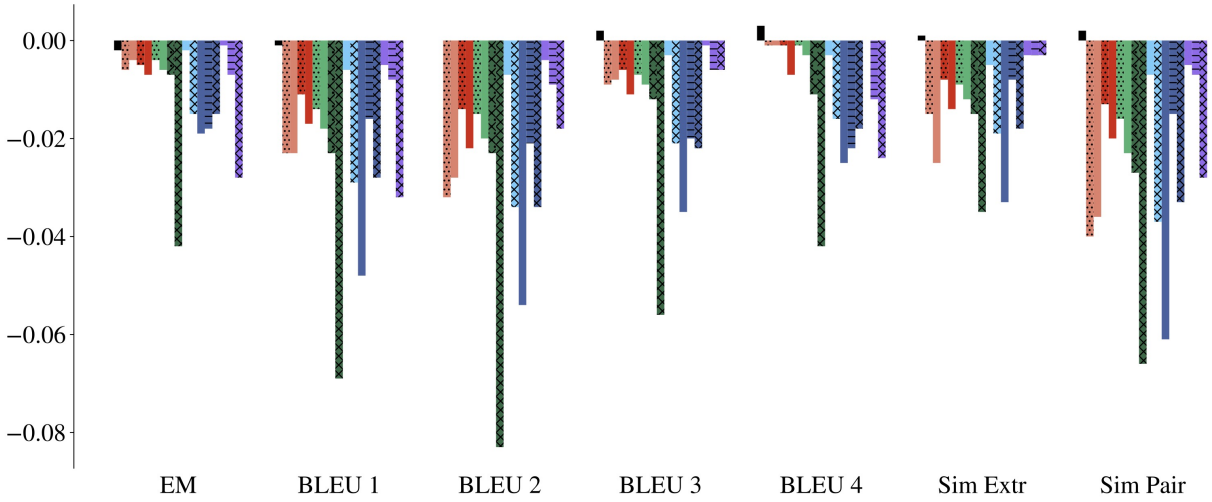
To assess the robustness of the approaches, we add one random session at the start of each session of the test set. Since the intent of these added sessions is not the same (on average) as the intent driving the user's behavior when formulating test queries, models must have learned to identify thematic breaks, and to ignore this noisy information. Figure 3.5b shows percentages of performance loss for every metric. We can see that for all models, the flat architectures are much less impacted than their corresponding hierarchical counterparts. This is an important result since the test sessions were arbitrarily split according to a 30-minute timeout, which might not correspond to users' intent changes. It shows that with the hierarchy, the transformers lose their ability to focus on the relevant part, and so to adapt themselves to longer sessions.

3.5.1.3 Sessions lengths

We study the impact of the session lengths on the two pre-trained models BERT and BART (flat and hierarchical versions) on the AOL dataset. Results are reported in Figure 3.6. Whatever the metric, the hierarchical models (in green) perform better than the flat ones (in red) for short sessions. However, for longer sessions (above 7 queries), it is the other way around. The flat models scores remain stable while the scores of the hierarchical models decrease. The hierarchical architecture of Garg, Dhillon, and Yu (2019) is adapted to short and more simple sessions search, but for longer and complex tasks the flat transformers



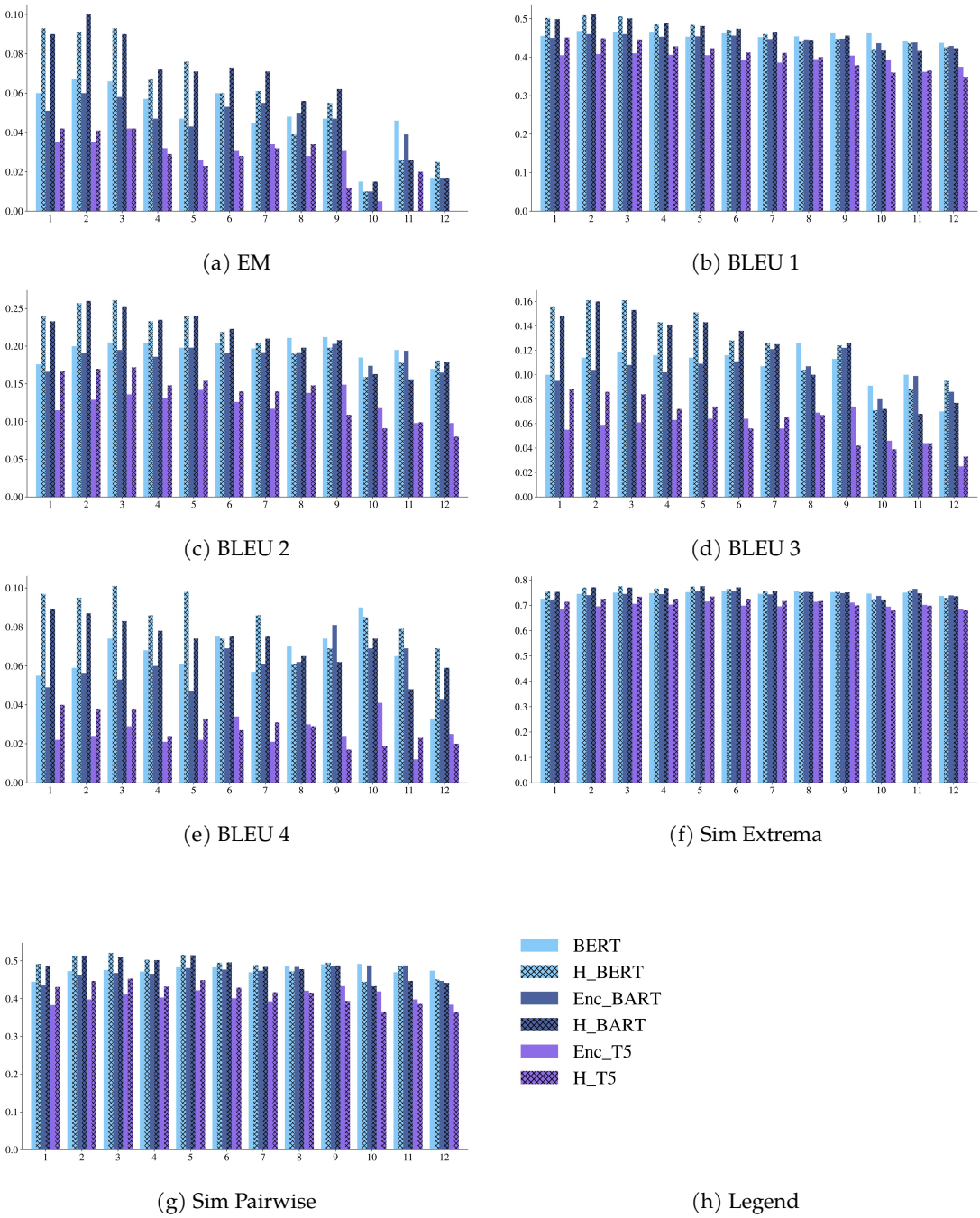
(a) *Complex* sessions



(b) *Concatenated* sessions

Figure 3.5: Difference between the performance on all the AOL sessions and on the noisy version (filtered/concatenated). Negative values indicate a degradation.

are more suitable. We believe that this is due to the fact that hierarchical transformers cannot focus reliably on the relevant parts of the session.



(a) EM (b) BLEU 1

(c) BLEU 2 (d) BLEU 3

(e) BLEU 4 (f) Sim Extrema

(g) Sim Pairwise (h) Legend

Figure 3.6: Models scores depending on the length of the sessions

3.5.2 Query generation

Having studied the robustness of the different models, we know that BART is the most suitable model for our task. We now investigate the behavior of this latter and design experiments to answer the last question **Q3: How does the flat transformer generate queries?**

Several papers propose to analyze transformers to check which information is learned or used (Clark et al., 2019; Jawahar, Sagot, and Seddah, 2019; Brunner et al., 2020) through either probing different parts of the layer or by looking at the attention towards the input (Clark et al., 2019). In this section, we follow this latter line of work, focusing on specific properties of transformers for query generation.

To do so, we focus on the attention of the decoder towards the encoder output (see Section 2.1.4), i. e. the attention weights computed for $A^{\mathrm{d}\to\mathrm{e}}$. When generating the $(t+1)^{\text{th}}$ token, we denote $\alpha^{(t)}_{lhij}$ the attention from the i^{th} decoder token into the j^{th} encoded token for each layer l and attention head h. To summarize this information, we (1) average the attention over the different heads – following (Clark et al., 2019); and (2) only look at the attention of the j output token when generating the $j+1$ output token. The rationale for the latter is that the generated token at step $j+1$ mostly depends on the final representation $t_j^{(L_d)}$ of the decoder token j, as shown in equation(2.3). Moreover, we observed that the attention did not vary much during the generation process, and hence those values are close to their average. We denote those averaged and picked attentions of token i on token j at the layer l as $\tilde{\alpha}_{lij}$.

Finally, as shown in (Brunner et al., 2020), the attention weight might not be a reliable indicator in all cases, since the actual modification of the representation depends on the value $v_h(s_i^{(L)})$ as shown in equation(2.2). To cater for this problem, we define the *importance* (of an attention) $\beta^{(t)}_{lhij}$ as $\alpha^{(t)}_{lhij}\|v_{lh}(s_i^{(L)})\|$. As for the attention, we summarize those values as $\tilde{\beta}_{lij}$. Unless specified, we focus on results for BART — but most of the behavior is shared by the different versions of the transformers we analyzed.

3.5.2.1 The growing importance of queries

In this section, we answer the first sub-question **Q3. (a) On which context's queries does the flat transformer focus its attention?** (Sordoni et al., 2015) claim that the last query — which they called the anchor query — plays a crucial role in queries suggestions. We verify this claim by assessing whether more attention was paid to the last queries in a session or not. For long enough sessions (≥ 5 queries), and for each query, we first sum the importance $\tilde{\beta}_{lij}$ over its tokens, and normalize the value by dividing it by its maximum value, so that we can average sessions of varying length. For the same reason, we normalize the index of each query by the length of the session, i. e. $i/|S|$. In Figure 3.7, we plot the boxplot of the importance given the normalized index of the query in the session. The x-axis corresponds to the position of the query in the session (from left to right: from the beginning to the end of the session), and the y-axis to the importance of the query.

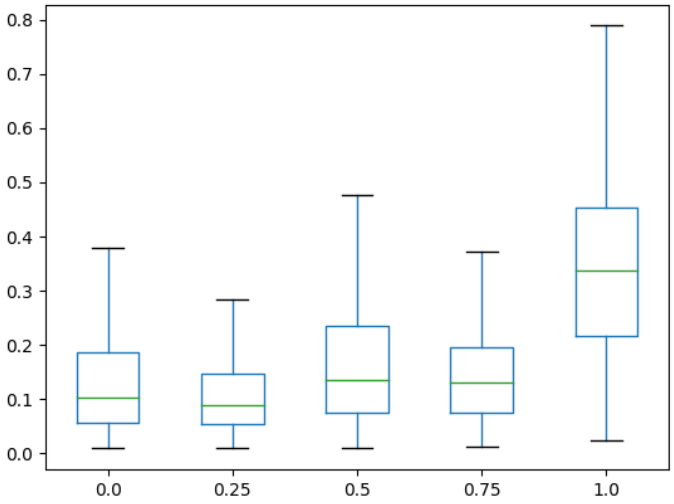


Figure 3.7: Importance of the queries depending on their (normalized, and using quantiles) positions in a session (average over layers)

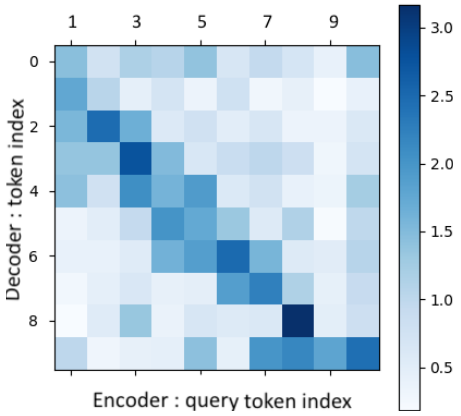
We see that there is a trend showing that last queries are more important for the prediction of the transformers since they have more impact on the vector used for predicting the output. It also explains the robustness of BART on concatenated sessions 3.5b.

3.5.2.2 The importance of the context's tokens

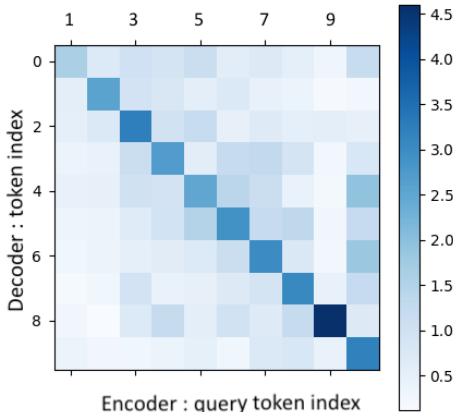
We now answer the second sub question of **Q3. (b) On which context's tokens does BART focuses its attention?**

For each decoded token (including the special token START numbered 0), we first look at the importance assigned to encoded tokens. In Figure 3.8, each cell (i, j) in the grid gives the importance of the j^th token (of each query in the session, e. g. the second token of each query in the session is numbered "2") when decoding the i^th token of the target query.

We only plot the importance for two representative layers (1 and 12), as we can distinguish two layers groups that behave similarly (not shown here: 1 to 4, and 8 to 12). We can observe that at layer 1 to 4, the importance focuses on tokens that match the same position (e. g. the first tokens of each query and the first decoded token). For the decoder token START (numbered 0), the importance is more broadly distributed — which is sensible since nothing has been generated so far. On layers 8 to 12, the importance focuses on tokens that match the *next* token position (e. g. the first tokens of each input query for START, the second tokens of each input query for t_1, etc.). This shows that transformers



(a) Layer $l = 1$



(b) Layer $l = 12$

Figure 3.8: Importance of the tokens depending on their position in the queries (attention of the decoder on the encoder), for layer 1 (a) and layer 12 (b) of the encoder. The X-axis corresponds to the context — i. e. the encoder tokens (averaged over all queries), while the Y-axis corresponds to the decoder — i. e. the decoder tokens. For the decoder, 0 corresponds to the START token. For instance, from (a) we see that when generating the 3rd token (row of index 2), the attention is focused mostly on the second token, and also (but less) on the first and third ones. This is different for the same token at layer 12 (b), where most of the attention is focused on the third token of every past query. Results are averaged over 20 000 sessions.

first focus on the matching encoded token before selecting the next token to generate.

The figure also underlines that BART, even without explicit hierarchical architecture, is able to capture the basic structure of sessions, the attention being in average more focused around the matching tokens (i. e. same position) of the queries present in the context session (as shown by the diagonal in both graphs).

3.5.2.3 Generating a new token

Finally, we answer the last sub question **Q3. (c) How does the model choose the next token to generate?**

This brings interesting questions in terms of the generative process of the transformer-based architectures. For the START decoder token, we observe that they first focus on the "[SEP]" encoded tokens, and then shift their attention to the next ones — relying on the position embedding that is added to the encoded token representations. For the next tokens to be generated, this is less obvious since the model could simply focus on a matching token (e. g. the decoder token "cat" matches the encoded tokens "cat"). As queries are often repeated within a session with small variations, the tokens might be in the same positions (in average) in the session queries and in the generated query. Consequently, to generate the next token, there are two possibilities: either the transformer shifts the attention towards a token to the right (position-based decision), or, the (query) language model of the decoder proposes a direction in the token space, which is then matched if an encoded token lies in this direction in the representation space.

To look into this, we used sub-sessions of the form

... ... | ... A B C ... | ...A B

for which the next query to be predicted (in red) contains a bi-gram of tokens (A,B) that exists in the past queries, followed by a different token C. For example, the target query contains "black/A cat/B" and the session contains a query with tokens "black/A cat/B sold/C". We calculate the probability of generating after "black/A" in the target:

- the target token ("cat/B") with a probability $P(B|S, A)$
- the token following the bi-gram in the context ("sold/C") with a probability $P(C|S, A)$

We do this for two settings: 1) using the original context session as S and 2) using a modified context session S for which we swapped tokens B and C in the context (i. e., substituting "black/A sold/C cat/B" to "black/A cat/B sold/C"). The goal is to assess whether the model favors a language model (LM) that captured that B usually follows A, or rather a copy mechanism that mainly considers positions from the context session (POS). Following this process, the average probabilities are computed over a set of 20000 sessions and are reported in Table 3.5 for the different transformers.

First, when position (in the context session) and language model agree (first and second columns), the probabilities are high for the real target and low otherwise. Among the different models, we note that

Table 3.5: Probabilities on mixed and unmixed sessions. For each original and swapped sessions, the preference of the model is highlighted in red (for differences above 0.01)

Session S	original ...A B C ...			B/C swapped ...A C B ...		
probability favors	$p(B\|S,A)$ LM/POS	$p(C\|S,A)$	total	$p(B\|S,A)$ LM	$p(C\|S,A)$ POS	total LM/POS
TS_wp	0.19	0.03	0.22	0.19	0.03	0.22
H_TS_wp	0.67	0.01	0.68	0.37	0.22	0.59
BERT	0.46	0.01	0.47	0.17	0.23	0.40
ENCBART	0.51	0.00	0.51	0.21	0.20	0.41
ENCT5	0.57	0.02	0.59	0.21	0.26	0.47
BART	0.70	0.03	0.73	0.35	0.28	0.63
T5	0.80	0.02	0.82	0.36	0.36	0.72
H_BERT	0.63	0.01	0.64	0.20	0.34	0.54
H_BART	0.72	0.01	0.73	0.29	0.28	0.57
H_T5	0.68	0.01	0.69	0.31	0.27	0.58

the best performing models (Section 3.4.5) have a very high probability of generating the token B (between 0.7 and 0.8).

When position (in the context session) and language model disagree (fourth and fifth column), the behavior of the architectures is quite different. Apart from the TS_wp (and to a lesser extent its hierarchical version) which mostly follows the language model (0.03 vs 0.19) and ignores the context session, we see that all the other models assign balanced probabilities to position and language in these swapped sessions.

Sufficiently powerful flat models such as BART appear sufficient to capture the query organization of sessions, while keeping enough flexibility to adapt to perturbations. We indeed observe that BART has both high probabilities of either following the language model or the position-based prediction (total probability of 0.63), which is nearly as high as when the context session and language model match (0.70). This difference with the other models might explain why BART is performing so well: it leverages both the copying mechanism and its powerful language model.

3.6 Conclusion

Inspired by the success of transformer-based models (Vaswani et al., 2017) in various NLP and IR tasks, we looked at the various architectures that could be applied to query generation. We compared tokenizers, architectures, and different pretraining methods. We show that while hierarchical models permit to obtain better performance than corresponding flat architectures, they are not adapted

for long and complex sessions. We conducted a deeper analysis of the flat models to understand why they are better at handling these sessions. We analyzed their generation process and found that the flat transformer is, on one hand, a position model that is able to recover the structure of a web search session (input queries are concatenated), and on the other hand, a good (query) language model.

This work could be extended in various directions. It would be worthwhile to look for ways to improve the hierarchical architecture, so the model could handle more complex search tasks, and could incorporate signals of various natures (longer history, clicked documents) into transformer-based architectures. This study is limited to query-based search sessions, but the hierarchical structure of data is also present in conversational searches (Aliannejadi et al., 2019; Zamani et al., 2020). However, while for the query suggestion task users can be modeled according to their own past actions only, the setting of conversational search requires considering external data such as the IR system's answers or available documents in the collection, to drive the user toward their target documents. It could also focus on working on architectures able to cope with long sessions, potentially all the user history, using other recently introduced transformers (Dai et al., 2019; Kitaev, Kaiser, and Levskaya, 2020; Beltagy, Peters, and Cohan, 2020) that overcome the limitations of the maximum context length.

The studies conducted on various models allow a thorough comparison of the ability of different models to predict the next user query in a more or less complex search session. As discussed in Section 2.3.2, a model with this ability can be used as a user model. The models studied can easily be enriched by incorporating other actions, such as clicks, into the model's input or prediction, in the manner of Borisov et al. (2016) and Borisov et al. (2018). Transformers are thus excellent behavioral models or user simulators. However, it would be useful to get a unique user representation, which could be obtained by using the [CLS] token representation during training to make it carry more semantics (as is done by (Liu and Shao, 2022)).

The work carried out in this chapter, beyond the query suggestion tool, allows us to work on new interactive systems with fine user models. This is the subject of the next chapter.

Chapter 4: Interactive IR

To tackle complex IR tasks, where users cannot precisely define their needs, interaction is paramount. Both query-reformulation approaches and chatbots are limited for this type of task since the former only learn to mimic users, while the latter are bounded by the domain they have been trained on. To take a first step towards truly exploratory and interactive IR, we introduce a framework, where users navigate document collections by expressing their preference among sets of queries proposed by the system at each step – thus refining the knowledge about the user's information need. Our training approach, based on self-supervised and reinforcement learning techniques, aims at minimizing the amount of interactions required to reach relevant queries, and thus documents, for users. We experimentally show that the introduced framework enables efficient learning from interactions with simple user bots, that are demonstrated to generalize well in real-world settings.

4.1 Introduction

For complex search tasks, when user needs cannot be precisely defined from a single query, interaction with session-based Information Retrieval systems is essential. Different session-based IR models have been proposed (Yang, Guan, and Zhang, 2015; Luo, Dong, and Yang, 2015; Luo et al., 2015), but they focus on biasing the document ranking process, thus preventing the user from truly interacting with the system. More direct interactions can be provided using query suggestions approaches (Sordoni et al., 2015; Dehghani et al., 2017; Mustar, Lamprier, and Piwowarski, 2021), that help users by reformulating their needs from interactions during the session. Most of them (see Section 3.2) are based on behavior models to predict the next queries of search sessions. Finally, chatbots for Information Retrieval, while ambitious in their goals, are usually ad-hoc systems, that are restricted to simple

dialogues for the specific domain they have been trained for (Chen et al., 2019).

Going further supposes IR systems able to *anticipate* user behavior so that they can proactively help users in their search tasks, as well as systems that can consider various possibilities in the evolution of the search process.

In this chapter, I first present a brief overview of our framework, IRnator, which allows a user and a system to cooperate to achieve the user's end goal. I then compare the system to existing work in interactive search that attempts to obtain insights from users to improve results. In the remainder of the chapter, I describe the framework and the experiments conducted to test it. This work was published at the ICTIR conference (ACM SIGIR International Conference on the Theory of Information Retrieval): Mustar, Lamprier, and Piwowarski (2022)

4.2 IRnator overview

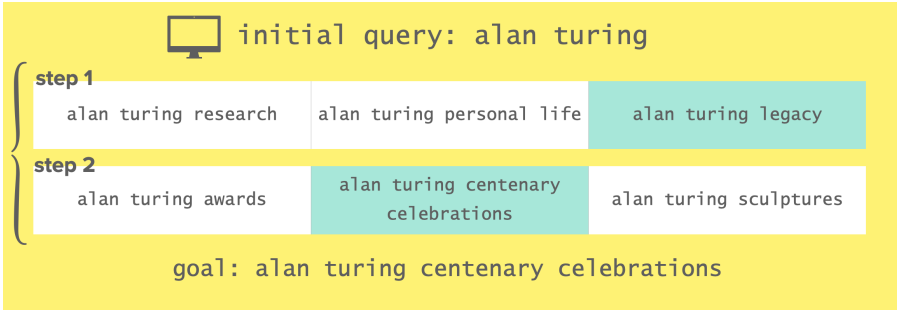


Figure 4.1: IRnator: the proposed framework

Rather than directly attempting to answer the user need, which is usually ill-defined for complex needs, or trying to have a conversation with the user about its interests, which is very difficult to efficiently drive and interpret, we introduce a new kind of interaction methodology inspired by Akinator-like systems (Groza and Coroama, 2019), i. e. systems that find a user's intent by asking questions about it. Our system (see Figure 4.1) successively proposes K query suggestions among which users can choose their favorite.

We think that this task, while simple, if successfully conducted, can be the basis of more ambitious conversation-based IR models because it (1) supposes refining the system's knowledge about users' needs to guide them more quickly toward relevant queries until it uncovers the user intent; and (2) requires that the system proposes different paths the user can follow.

4.3 Positioning

Our work is at the crossroads of many: asking clarifying questions, query suggestions, interactive IR systems, and user simulation. In this section, I position our work in relation to existing interactive systems.

Query Suggestion As discussed in Chapters 2 and 3 query suggestion works (Sordoni et al., 2015; Dehghani et al., 2017; Mustar, Lamprier, and Piwowarski, 2021; Garg, Dhillon, and Yu, 2019; Wu et al., 2018b) model users' sessions so as to predict their next query, which is then used as a suggestion. Most of these works do not take into account user feedback, except (Wu et al., 2018b) who use clicked (or not) documents. However, all of these works consider that users' future queries are relevant suggestions. In contrast, we view query suggestions as a way to uncover users' intent.

Our method differs from query suggestion in three other ways. First, query suggestion focuses on one or a few steps (Sordoni et al., 2015; Dehghani et al., 2017; Mustar, Lamprier, and Piwowarski, 2021; Garg, Dhillon, and Yu, 2019; Wu et al., 2018b) of a search session. In contrast, we aim at helping users to fulfill their information needs.

Second, while query suggestion works mostly focus on behavioral cloning methods, wherein the agent learns to mimic the user by predicting future actions, we aim at explicitly shortening the user efforts. We argue that this is necessary since users do not necessarily know the best course of actions to reach relevant documents.

Finally, the data needed to train query suggestion models are based on search session logs. These logs are expensive to obtain and raise serious questions about user privacy. They are dependent on the search engine used by the user at the time of extraction and do not allow the model to generalize if new goals or queries arise. It is thus interesting to develop models that do not rely on this type of data.

Clarification questions Several works have proposed to ask questions about users' goals to infer them. In particular, (Burgener, 2006; Hu et al., 2018; Wu et al., 2018b) study the Q20 game, and (Yu et al., 2019) the Akinator game, where the agent asks questions about the goal. In the proposed framework, the main difference is that the search space is much higher, and there are no predefined attributes that can guide the search. More IR-related, (Dhole, 2020) disambiguate an initial query by asking a question to discriminate the most likely intent, but in a one-step interactive process that can only be applied when the number of intents is small.

Some tasks such as product recommendation (Yang et al., 2021b; Bhattacharya et al., 2017) bear some similarities with our work since they aim at predicting user intents. These works are generally based on item (category, etc.) and user metadata (gender, age, location, etc.) and

interaction logs. Our research direction is orthogonal since we focus on a session-based single intent prediction – beside not using any metadata and/or interaction log.

Interactive IR models Closer to our work, interactive search sessions have already been modeled (Yang, Guan, and Zhang, 2015; Luo, Dong, and Yang, 2015; Luo et al., 2015) as a MDP (Markov Decision Process), in which the search engine plays the role of the agent. These works focus on ranking documents, and not on the interaction with the user, which could provide a better understanding of the user's goal. For instance, (Yang, Guan, and Zhang, 2015) studies user behavior by focusing on syntactic query changes during a session and doesn't provide the user with additional information. While (Pallagani and Srivastava, 2021) uses a setting closer to ours, it learns a strategy to reach the user's goal as quickly as possible. However, it works with structured data (with a hierarchy) and requires conversational data.

User Model Finally, Maxwell and Azzopardi (2016b), Baskaya, Keskustalo, and Järvelin (2013), Thomas et al. (2014), Maxwell and Azzopardi (2016b), and Câmara, Maxwell, and Hauff (2022) attempt to simulate users, based on a more or less complete description of the user's need. While simulating IR users in an interactive setting is a crucial topic to developing better interactive IR systems, such models are still difficult to use and not so reliable. In this work, we rely on a simple user heuristic, that allows to get a large number of simulated sessions needed for training our model and leave for future work the use of more sophisticated models.

4.4 Problem formalization

Let us consider a session S composed of S interaction steps between a user ξ with a goal g and an IR system π. We suppose that the session starts with an initial query q_0, which follows a distribution $\xi_0(g)$ of initial queries for the user ξ having a need g. Each interaction step t corresponds to $S_t = (Q_t, u_t)$, where $Q_t = \{q_t^1, .., q_t^K\}$ corresponds to a set of K query suggestions, and u_t is the index of the user's preferred suggestion amongst the K. A complete session is denoted as $S = (q_0, S_1, \dots, S_{|S|})$.

Any user choice u_t of a given session follows a conditional distribution about preferences of the user given the goal and the session up to step t, i. e. $u_t \sim \xi(u_t|g, S_{<t}, Q_t)$, where $S_{<t}$ denotes all interactions before step t in the session. Successive sets of questions suggested by the system also follow a conditional distribution $\pi(Q_t|S_{<t})$ given the previous interactions of the session $S_{<t}$ at step t. Finally, a session S with a goal g follows a distribution $\delta_\pi^\xi(g)$, depending both on the user model ξ and the suggestion system π.

The aim is to suggest query sets Q_t that allow to increase information about g as much as possible at each step, to help users achieve their goal as soon as possible. We introduce an interactive IR system whose aim is defined as the following maximization problem, given sessions with a maximum number of interactions T:

$$\pi^* = \arg\max_{\pi} \mathop{\mathbb{E}}_{g\sim\mathcal{G}} \mathop{\mathbb{E}}_{S\sim\mathcal{S}^{\xi}_{\pi}(g)} \left[\sum_{t=0}^{T} \gamma^t Achieved(g, S_{\leq t}) \right] \tag{4.1}$$

where $\mathcal{G}$ is the distribution of goals and $Achieved(g, S_{\leq t})$ is a binary function that returns 1 if goal g can be directly completed given information from $S_{\leq t}$, and 0 otherwise. The precise definition of *Achieved* depends on the considered IR system. The proposed framework can consider complex goals, which implies for instance the interrogation of a document retrieval system given the last selected query (or the full past session) and the inspection of the corresponding returned documents to assess the completion of g. In practice, it should be set to the propensity of users to continue their search. In Equation 4.1, $\gamma \in]0;1[$ is a discount factor that pushes to prefer sessions that complete goal g as soon as possible. For the sake of simplicity, and to avoid the dependence on a document collection with its specific retrieval system, we consider that goal g can be expressed as a query q_g and that g is achieved at step t if the system π proposes a set of suggestions that includes q_g i. e., $q_g \in \{q_t^1, .., q_t^K\}$.

4.5 Challenges

The problem as defined in Equation 4.1 is however particularly difficult to directly solve using standard Reinforcement Learning algorithms, as it involves the following challenges:

Query space size Ideally, given a vocabulary V of $|V|$ tokens and a max query length L, any suggested query q lives in V^L. This is huge, even for reasonably-sized vocabularies, and includes many sequences that do not correspond to human-readable queries (e.g., with token sequences that form words that do not exist in the user's language). While a prior query-language model could be used, in this work we simplify the task for π, by only restraining suggested queries to a set Q of pre-defined ones, from which the system samples sub-sets at each step, which allows to greatly restrict the search space. Dealing with more complex (generative) strategies are left for future works.

Combinatorial action space Even with the reduction of the query space as proposed above, the action space remains particularly large, because of its combinatorial aspect: an action for π corresponds to select K queries from Q, inducing an action space of size Q^K. While a policy π

composed of a main network (e.g., transformer) with K heads on top of its output would be an option, this still implies a complex search space, involving a hard credit assignment problem, well known in the multi-agent RL literature (Foerster et al., 2017). As detailed in the following section, we assume a well-structured semantic representation space of queries, that reduces the choice of Q_t to a single point in the space, from which the set of K suggestions can be deterministically determined (here, by clustering queries).

User model ξ unknown Modeling users of interactive IR systems is a particularly difficult task (Câmara, Maxwell, and Hauff, 2022). Beyond the lack of training IR session data, especially when considering innovative systems, behaviors of users are very difficult to precisely predict in many settings, due to the implication of many confounding factors. While it is well known that behaviors are not stationary during IR sessions, we assume here that past interactions do not modify users' preferences during the search. Moreover, rather than modeling complex user behaviors, as done for instance in classical – short term – query suggestion (Mustar, Lamprier, and Piwowarski, 2021) as discussed in Chapter 3, we assume in the following a simple user bot as ξ, hard-coded with pre-defined heuristics shared across sessions, though possibly hidden from the system agent π to be general enough for application of the model in real-world settings (where minds of users are not accessible).

Very sparse reward problem As defined in Equation 4.1, system π must succeed in generating a target query in less than T steps to expect a non-null reward. Thus, in the first steps of learning, no improvement direction of π is given to the learner, preventing it from completing the task. Reward shaping (Ng, Harada, and Russell, 1999) is a popular way to densify rewards for such hard problems, where advisories about states to visit are given as potential functions $\phi : \mathcal{S} \rightarrow \mathbb{R}$, with $\mathcal{S}$ the set of reachable states in the environment[1]. In addition to a self-supervised learning process to initiate the learning process, we consider in the following a learned model of user intent prediction ϕ_g, based on the partial user sessions, to drive the learning of π following directions which minimize the uncertainty of g with respect to this model.

Note that, assuming a well-known user that deterministically selects the closest suggestion to its goal in its own Euclidean representation space ψ^{ξ}, the problem as defined in Equation 4.1 could be greedily optimized by choosing each step t the set of queries that minimizes the number of admissible goals regarding $S_{\leq t}$. For a probabilistic user, the optimal solution could be approximated by suggesting at each step t the set of queries $Q_t = \{q_t^1, .., q_t^K\}$ that minimizes the conditional entropy $H(G|U_t)$ with:

1 In our setting, $\mathcal{S}$ corresponds to the full set of possible search sessions that can be built for any user from the set of all possible needs.

$$H(G|U_t) = \sum_{u=1}^{K} \xi(u|Q_t, S_{<t}) H(G|S_{\leq t}) \tag{4.2}$$

with $\xi(u|Q_t, S_{<t})$ the marginal probability that the user selects the query of index u given Q_t and the past of session $S_{<t}$, and $H(G|S_{\leq t}) = -\sum_g \phi(g|S_{\leq t}) \log \phi(g|S_{\leq t})$ the entropy of goal distribution given session $S_{\leq t}$. However, while this can be considered for instance for interactive classification with restricted sets of labels and closed questions, such as in (Yu et al., 2019), this is completely intractable in our setting.

4.6 Learning to drive users towards goals

This section first presents the considered suggestion architecture π, before describing self-supervised and reinforcement learning techniques used to solve the task.

4.6.1 Query suggestion process

Let us consider that the set of all possible queries $q \in Q$ belong to a continuous representation space, i. e. $\psi(q) \in \mathbb{R}^d$. Figure 4.2 depicts the proposed suggestion process, where π is implemented as a Transformer architecture (Vaswani et al., 2017), which takes as input the session S and outputs a set of K suggestions (in the figure, $K = 3$).

To provide a diverse set of suggestions, we rely on a clustering process based on a point $\pi(S)$ predicted by our model. The N closest queries from Q (queries are represented by crosses in the figure), depending on Euclidean distances in the continuous space ψ, are selected and clustered into K groups. Finally, the K medoids of clusters are used as the set of queries Q_t proposed to the user at step t. The user selects their preferred query, depending on g and ξ (q_3^3 in our example), which is the closest suggestions to g. This feedback u_t defines S_t that is used for the next suggestion step.

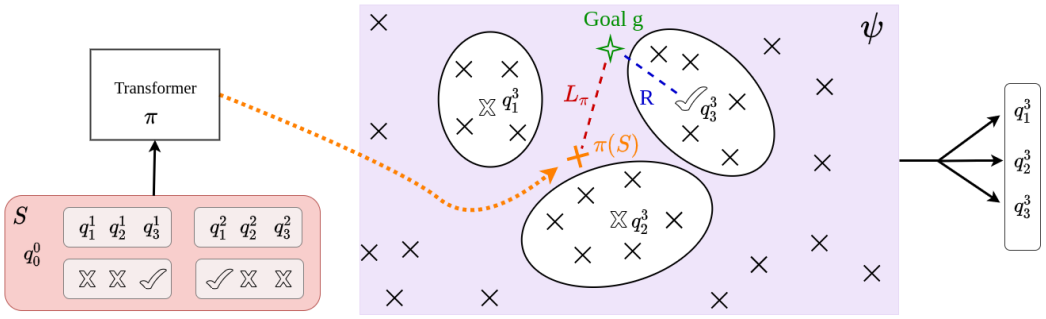


Figure 4.2: Query suggestion process

The assumption behind the use of a clustering method (a simple K-means approach in our experiments) is that the neighborhood $\pi(S)$ in ψ contains the main aspects that can specialize $\pi(S)$, which can be partitioned in relevant sub-topics to present to the user. We argue that, while the use of hyperbolic representation spaces (Tay, Tuan, and Hui, 2018) could allow to even improve accuracy (which we leave as a possible extension of our work), the representation space ϕ we consider, which results from a pre-trained sentence-transformer designed for semantic search (Reimers and Gurevych, 2019), presents a structure that fits well with this assumption, with general queries tending to occupy central positions in the representation space.

The suggestion model π corresponds to a Transformer architecture (Vaswani et al., 2017), which takes as input sessions concatenation of the initial query with all past interactions, each S_t being encoded as the sum of three representations:

- Query embeddings are obtained by encoding the set of Wikipedia queries with a pre-trained sentence-transformer (Reimers and Gurevych, 2019) designed for semantic search. These embeddings are normalized as suggested by the authors.
- Positional embeddings allowing to retain the temporality of interactions
- User's action embeddings corresponding to the user's choice (1 for selected queries, 0 for the others)

We use a FAISS index (Johnson, Douze, and Jégou, 2019) to search quickly for the top-K nearest neighbors of a query.

This model π is trained using two learning modes: an iterative supervision and a training based on reinforcement learning. These modes are described in the next two sections.

4.6.2 Iterative Supervision

We don't have any existing sessions, which makes classical supervised training impossible. Instead, we propose an iterative self-supervision training.

The iterative process starts with synthetic trajectories generated with a random model π^0. The suggestion model learns to infer the goal g on these sessions. We obtain a suggester π^1 which is better than the random model π^0, because it proposes suggestions closer to the goal. New trajectories can then be generated with this new policy. Thus, over the course of the iterations, the set of generated sessions will be of better quality, allowing the model to infer more easily the users' goals.

Precisely, during the iterative supervised learning, we seek at minimizing the Euclidean distance between the point $\pi(S)$ predicted by

the suggestion model and the user's final goal (represented as L_π on Figure 4.2), given various input pairs of (goal, session) as input. At each iteration i of the training algorithm, the following optimization problem is considered:

$$\arg\min_{\pi} \sum_{(g,S)\in\Gamma^{(i)}} \|\psi(g) - \pi(S)\|_2^2 \tag{4.3}$$

where $\psi(g)$ returns the representation of the query targeted by goal g. In our work, we focus here on the case where a goal corresponds to a single target query i. e. $\psi(g) = q_g$. $\Gamma^{(i)}$ is the training set at iteration i, obtained using the distribution of goals $\mathcal{G}$ and the policy $\pi^{(i-1)}$, obtained at iteration $i-1$ of the learning, π^0 being a random suggestion policy. At step i, after optimization of Equation 4.3, ψ is used as the new policy π_{i+1}.

4.6.3 Reinforcement Learning

While the iterative supervised learning proposed in the previous section enables to train the model accurately, this may suffer from some limitations:

- no convergence guarantee due to the iterative process which does not take into account the dependence of the training data on the optimized model
- strong relatedness with the user heuristics, which prevents from the ability to adapt to different kinds of users
- no direct consideration of the queries presented to the user.

Thus, we propose here to consider possible refinement of the supervised model via reinforcement learning techniques, notably DPPG (Barth-Maron et al., 2018), a policy gradient approach specifically designed for continuous actions as it is the case for our setting where the action corresponds to outputting point $\pi(S)$.

The basics of reinforcement learning have been described in Section 2.1.6.2. I detail below RL basic algorithms and DDPG.

Let us consider a MDP (S, A, R, P) with S the set of states, A the set of actions, $R : A \times S \to \mathbb{R}$ the reward function and P the transition probability $P(s'|a, s)$ of being in state s' after the agent took the action a in state s. The agent is defined by a policy $\pi(a|s)$ which is a distribution over possible actions $a \in A$ given a state $s \in S$. This policy is trained to improve the expected reward $\mathrm{E}_{\tau\sim\pi}[R(\tau)]$.

Q-learning In order to get closer to the optimal policy π^*, an action-value function Q^π which gives the expected return with the policy π, a state s and an action a is defined:

$$Q^\pi(s,a) = \mathop{\mathrm{E}}_{\tau\sim\pi}[R(\tau) \mid s_0 = s, a_0 = a] \tag{4.4}$$

Bellman equations The majority of RL algorithms are based on Bellman equations. The idea of the equation is that the value of the next state is the expected reward from being at this state plus the expected reward of the next state.

$$Q^\pi(s,a) = \mathop{\mathrm{E}}_{s'\sim P}\Big[r(s,a) + \gamma \mathop{\mathrm{E}}_{a'\sim\pi}\left[Q^\pi\left(s',a'\right)\right]\Big] \tag{4.5}$$

Deep Q-Network Mnih et al. (2015) propose a learning algorithm, Deep Q-Network (DQN). Let $Q^*(s,a)$ be the optimal action-value function, then the optimal action is $a^*(s) = \arg\max_a Q^*(s,a)$. Based on the Bellman equation, with ϕ the parameters of Q_ϕ, we want to minimize:

$$L(\phi, \mathcal{D}) = \mathop{\mathrm{E}}_{(s,a,r,s',d)\sim\mathcal{D}}\left[\left(Q_\phi(s,a) - \left(r + \gamma(1-d)\max_{a'} Q_\phi\left(s',a'\right)\right)\right)^2\right] \tag{4.6}$$

where $\mathcal{D}$ is a set of transitions (s,a,r,s',d), with $d = 1$ indicating that the state s' is terminal, else $d = 0$ and γ the discount factor.

DDPG The RL algorithm that we use to train our IRnator framework is Deep Deterministic Policy Gradient (Barth-Maron et al., 2018) which extends the DQN algorithm to continuous space. DDPG is built according to an actor-critic framework, i. e. two models are learned: the critic Q_ϕ of parameters ϕ, and the policy π_θ of parameters θ. The Bellman equation (Equation 4.5) is used to learn Q_ϕ (Equation 4.4), and DDPG leverages the latter to learn the policy π_θ. The algorithm relies on the fact that the action space is continuous to be able to differentiate Q_ϕ according to the actions, thus instead of computing $\max_a Q(s,a)$, it is approximated with $Q_\phi(s, \pi_\theta(s))$.

The policy π_θ parameters θ are learned by maximizing:

$$\mathop{\mathrm{E}}_{s\sim\mathcal{D}}\left[Q_\phi\left(s, \pi_\theta(s)\right)\right] \tag{4.7}$$

While critic parameters ϕ are learned by minimizing the following loss:

$$L(\phi, \mathcal{D}) = \mathop{\mathrm{E}}_{(s,a,r,s',d)\sim\mathcal{D}}\left[\left(Q_\phi(s,a) - \left(r + \gamma(1-d)Q_{\phi_{\text{targ}}}\left(s', \pi_{\theta_{\text{targ}}}\left(s'\right)\right)\right)\right)^2\right] \tag{4.8}$$

where $\mathcal{D}$ is a set of transitions (s,a,r,s',d) and γ the discount factor. $Q_{\phi_{\text{targ}}}$ and $\pi_{\theta_{\text{targ}}}$ are target networks that are used to stabilize the training

by regularly copying the weights from the networks Q_ϕ and π_θ to $Q_{\phi_{\text{targ}}}$ and $\pi_{\theta_{\text{targ}}}$.

Intent reward As previously mentioned, to deal with sparse rewards, and to gain flexibility regarding the considered user, we propose to consider a probabilistic intent model $\phi(g|S_{\leq t})$ as the intrinsic reward at each step t, implemented as a transformer that outputs the mean vector μ of a Gaussian $\mathcal{N}(\mu, I)$ with unit variance.

This allows rewarding suggestion sets that most improve knowledge about the hidden user's goal, according to the user's answer. The reward is hence defined as:

$$R_t = \log \phi(g|S_{\leq t}) - \log \phi(g|S_{<t}) \tag{4.9}$$

where the second term acts as a baseline. If $R_t > 0$ the intent model is getting closer to the goal, thus the last action has been useful. The bigger the reward, the bigger the information gain. This reward represents the information gained obtained after the user has been asked the question and responded to. The closer the intent model is to the goal, in relation to its previous knowledge, the more the policy is rewarded.

The intent model is refined regularly to update it regarding sessions generated with the last learned policy, via goal likelihood maximization.

Finally, rather than dealing with long-term reinforcement, which appeared unstable in our experiments, we propose to use a one-step ahead critic network $Q(S_{<t}, \pi(S_{<t}))$, that simply learns to predict R_t from past interactions and the output of the suggester π.

4.7 Experiments

4.7.1 Experimental Details

Data For our experiments, we use the Wikipedia dump from TREC CAR 2020 (Ramsdell and Dietz, 2020), which is interesting because it covers a large spectrum of domains and can provide pairs of queries. Indeed, Wikipedia page titles are used as initial queries, and the names of the subsections are concatenated to the title to obtain final goals. For example, for the page `` `anarchism' `` which contains a section `` `history' `` with a subsection `` `prehistoric and ancient world' ``, we get an initial query-goal pair: (`` `anarchism' ``, `` `anarchism prehistoric and ancient world' ``). This method provides general initial queries and specific goals. As the latter are specific, they would probably not have been the initial query of a user. An important advantage of this method is that an initial query can lead to different goals, so the model *must* actually learn to suggest discriminating queries and to use the user's answers (rather than relying only on the first query). Furthermore, to focus on rather complex search goals, we only kept pages with at least 3 sections,

each containing at least 2 subsections. Sections with too long titles (more than 3 words) or too generic – e.g. `see also, references, citations, sources, further reading, external links, notes, other, notes and references` – are also filtered out. Following this process, we get 633,647 pairs that we split into train and test with a 80-20 ratio. The data is split so that there is no common goal between the train set and the test set: test goals were never seen during the training phase. The scores reported are computed on the test set.

Note that the dataset can be easily expanded with other queries from different sources. We are aware that using synthetic data has its shortcomings, but using (filtered) query logs would have introduced too much noise, preventing analyzing the model behavior in such a controlled setting, where learning a working system is already challenging.

	SC	*SS*	SSC_{rand}	*SSC*	*DDPG*
% success	0.175	0.178	0.308	0.457	**0.475**
# steps	5.212	5.215	5.080	4.571	**4.568**
min. dist	0.371	0.409	0.415	0.259	**0.253**

Table 4.1: IRnator. Models scores

Compared Models In our experiments, unless specified otherwise, we use a simple model to simulate the user's choices, both at train and test time (except for the human evaluation experiment): at each step, our bot user chooses the closest query (in term of Euclidean distance) to its target goal in the representation space ψ. We compare our Self-Supervised Suggestion model with Clustering (*SSC*) with three of its ablations:

- To assess the usefulness of the suggestion model π, we remove it in the first ablation (*SC*). Instead of π, we use the previous user's choice to obtain the next suggestions, i. e. $\pi_{SC}(S_{<t}) = \psi(q_{t-1}^{u_{t-1}})$ where $\psi(q_{t-1}^{u_{t-1}})$ is the representation of the $q_{t-1}^{u_{t-1}}$ query.
- The second ablation (*SS*) removes the clustering step, and replaces it by proposing the K queries closest to $\pi(S_{<t})$. This ablation allows us to measure the value of clustering.
- Finally, the last ablation (SSC_{rand}) considers a random user for supervision rather than our heuristic bot user described above. This enables us to determine the extent to which users' responses are actually taken into account by the suggestion policy.

Finally, we also consider a policy fine-tuned via RL (*DDPG*), as described in Section 4.6.3.

All models use $K = 3$ and a maximal session length of $T = 6$. The policy and the intent models all have the same architecture: a transformer with 6 heads and 6 layers and a dropout $p = 0.1$. We use a feedforward network with two layers with hidden size of 768, which corresponds to the size of the embeddings from the chosen pre-trained sentence-transformer (Reimers and Gurevych, 2019), to compute $\pi(S_{<t})$ from the contextualized `CLS` token. The model is optimized with Adam (Kingma and Ba, 2015) – we observed that the Self-Supervised model converged quickly after a few steps (around 5-10).

4.7.2 Results

Suggestion policies are evaluated in terms of average success (i. e., the rate of sessions where the target query was finally suggested by the system within the T=6 steps of interaction), average number of steps to complete the task (using 6 if the goal was not reached) and minimum distance (i. e., the average distance between the closest suggestions and the target in each session).

Performances of Compared Strategies Table 4.1 reports the results of the compared policies π. First, the *SC* ablation obtains the worst results, which indicates that simply focusing on the neighborhood of expressed or selected user queries is not enough to help navigation, validating the usefulness of the learning task. Second, the *SS* ablation does not demonstrate significantly better results, which points out the relevance of the use of clustering to ensure the diversity of the suggestions. Third, and very importantly, the SSC_{rand} approach obtains significantly worse results than our *SSC*, which shows that the latter succeeds in leveraging useful feedback of users, only suggesting using the initial query and the structure of available ones is not enough. Finally, the reinforcement learning approach *DDPG* allows us to obtain the best results, with no big improvements over the self-supervised approach *SSC*, but showing the potential of using such a way more flexible learning paradigm that RL enables.

Human Evaluation To analyze if models trained with our heuristic user can be helpful for real users interacting with the system, we asked three annotators to use IRnator. At the beginning of each session, they are given an initial query and a goal to reach (see Section 4.7.1). At each step, they are asked to select the proposition that corresponds best to their final goal. Their aim is to navigate towards the specified target, only via selecting at each step a query among the three proposed. Suggestions are randomly proposed by one of the compared models, hidden from the annotator. The results are presented in Table 4.2 with 150 samples per model.

While the variance is high, we see that the general magnitude of the measures corresponds to the model scores with the simulated user, confirming the validity of our approach. The only difference is for *SS* (no clustering) – which can be explained because it is much harder for a human to know which query is closer to the target when they are not enough diverse, and for DDPG, which shows that we need more realistic user models to generalize better. We further discuss these points in the next section.

	SS	*SC*	*SSC*	*DDPG*
% success	0.06	$0.21^{\star}$	$0.5^{\star\dagger}$	$0.38^{\star\dagger}$
# steps	5.71	$5.17^{\star}$	$4.25^{\star\dagger}$	$4.83^{\star}$
min. dist	0.48	$0.34^{\star}$	$0.23^{\star\dagger}$	$0.32^{\star}$

Table 4.2: Human evaluation. $\star$ indicates significant gains ($p < 0.05$) compared to *SS*. † indicates significant gains ($p < 0.05$) compared to *SC*.

4.8 Discussion

IRnator is a generic framework for interactive search, which allows to study how an agent can guide a user in a knowledge space so that they reach their goal with minimal effort. We believe that, for a search engine, the challenge of learning to interact with a user is ambitious and requires simplifications that we restate and justify below.

User model Our user behavior is stationary (it does not depend on the previous interactions) and relies on heuristics. These simulated users are always able to choose the query closest to their goals (in the representation space). In reality, it might happen that none of the proposed queries matches what the user wants or that the user does not know which query is the best. We should study the possibility for the user to submit a new query, or to express negative feedback on the suggestions, rather than being forced to choose a proposition. Future works should explore more realistic user models, with more possible actions. However, even with such a simplified setting, we show in our human evaluation experiment that there exists a correlation between real and simulated users in terms of reduction of the effort to reach the goal.

Discrete query space We use a space with a finite number of queries to focus on the agent role as a guide towards the goal rather than dealing with text generation problems. However, in our experiments,

the database contains a large number of query/goals (633,647) from a Wikipedia dump, a website that covers many domains. The scores presented are from the test set, thus based on goals never seen in the training phase. This shows the generalization capacity of our model: the agent has learned to navigate in this knowledge space. The large size of the chosen space and the ability to generalize to new goals, allow us to think that simplifying the space to a finite number of queries is acceptable.

4.8.1 Conclusion

We introduced the IRnator framework, inspired by Akinator systems (Xie et al., 2018), for the context of complex search sessions in information retrieval. The aim of the system is to guess the hidden user's intent by suggesting sets of query suggestions and leveraging its feedback. Rather than hard-coding non-scalable suggestion heuristics, based for instance on conditional entropy minimization, the associated learning task aims at discovering efficient strategies according to the user's behavior. An efficient clustering-based solution on top of a Transformer architecture, learned via self-supervised and reinforcement learning, was proposed as a first solution for this innovative task. We expect many promising directions for this very challenging, but crucial, problem of intent discovery in IR. Those are discussed in the next chapter.

Chapter 5: Conclusion

5.1 Contributions

In this thesis, I studied the interactions between users and search engines with the long-term goal of assisting users with complex search tasks. I first focused on modeling users during a search task, before studying user-system interactions in a simple yet promising framework named IRnator.

User Model My first contribution was the analysis of query prediction models, which also serve as a query suggestion tool. This task is interesting since it requires anticipating user behavior from search engine logs. I compared task-specific models as well as generic language generators (e.g. Transformer). These analyses demonstrate that although models with a task-specific architecture perform well, large pre-trained Transformers are more robust. Further analyses of the generation process show that they are able to recover the structure of a search session and are also good language models. These models could be used to work on interaction systems as discussed next.

Interactive system The user models we propose mimic users' behavior during their search by learning to predict their next actions. However, this is usually insufficient since for complex search sessions where we want to design systems that can reduce the number of session steps to reach users goal. While there is a lot of work on interactive recommender systems, for which data are more structured and limited, there is no common framework for interactive search systems. Furthermore, evaluating their performances is hard. For this reason, we propose a paradigm in which the system and users interact in a limited setting. Inspired by the Akinator system, IRnator seeks to discover user's goal by asking them to make their preferences explicit. Although the framework is restricted, it allows a formalization of the problem and can be easily extended.

5.2 Experimental work and perspectives

We experimented with several ways to improve both the user model and the IRnator framework, which is only a first step towards truly interactive IR, by looking at several aspects: (1) architecture of the IRnator system, (2) realistic user simulation, (3) design of an intent model as a reward. We discuss each of these points in the following sections.

5.2.1 Improved suggestion system

IRnator applies a clustering algorithm to the k queries closest to the predicted intent. However, k should depend on the actual user-system state and not be fixed. Indeed, the space of our query collection does not have the same density at all points: depending on the topic, the structure of the neighbor queries change. Therefore, we tested a version of our model with an additional output to learn the number of queries to cluster. The results were however similar to the fixed number version.

This negative result can be explained by the fact that clustering itself might be a limiting factor. Clustering indeed supposes that relevant facets of any expressed intent are organized hierarchically in the representation space. A promising approach to enhance performance would therefore be hyperbolic representation space (Dhingra et al., 2018).

An alternative is to remove the clustering step. We designed a model that takes the history as input and directly predicts three suggestions. Initial experiments have shown that training this model is more complex, resulting in a model that tends to predict the same suggestion several times, contrary to the clustering approach. This suggests that a better exploration process is required. Rewards incentivizing diversity are also envisaged.

This approach, although not yet successful, is promising as it allows for more flexibility and does not rely on the imperfect clustering from our initial approach which might have been too naive.

5.2.2 Towards better user models

To simulate users in IRnator, we used a simple heuristic, namely that the selected query is closer to the final goals in the user's query space than other suggestions. Obviously, real users do not make choices in this way, and even if they did, we do not have access to this space. Therefore, we need a better user model. For training simulation, one promising work is to learn a parametric user model. We want to learn a model that is able to generate realistic search sessions from a small number of user logs, given a goal (a sampled information need).

As a preliminary setup to learn complex user models from a limited number of example sessions, we used a sequence model (a randomly initialized Transformer). From this model, we can generate a number of sessions, i.e. sequences of tokens where each token can be thought of as a user action (e.g., an issued query or a clicked document). The last action is deemed to be the *goal* of the user. Given those "user logs", we can now study how to learn a robust and realistic user model from them, that can express preferences among suggestions.

The simplest method to learn a user model is to use a behavioral cloning algorithm based on supervised learning on the available trajectories, similar to the models seen in Chapter 3. When a large number of user logs are available, the method succeeds in learning a user who is almost certain to reach their goal and has a low perplexity with the language model, i. e. the trajectory is successful and realistic.

On the other hand, when the number of available trajectories is limited, supervised learning is no longer conclusive. The simulated user fails to reach the final goal. To cope with this more realistic setting, we propose to use a RL technique, namely goalGAIL (Ding et al., 2019). In goalGAIL, an agent generates trajectories given an intent as a goal, and a discriminator $\mathcal{D}$ learns to differentiate the real user trajectories from those generated by the agent. $\mathcal{D}$ is then used to calculate the reward. The user model is then used to select the most relevant suggestions given the need, using likelihood ratios.

This preliminary work allowed us to study user models with the constraint of a small number of logs. This limitation is all the more important when proposing new search systems, such as IRnator, for which no large dataset exists. The next step would be to use real user logs and more complex user models that could be learned jointly with the suggestion system.

5.2.3 Enhanced intent model

RL algorithms depend on the defined reward. For the IRnator system, we proposed in Chapter 4 to use an intent model π that predicts the user goal given past interactions. The better the intent model predicts the goal, the better the reward and hence the learned IRnator model. In the experiments presented below, we were interested in how the IRnator system could represent the user's needs in a space where it could predict the user's goals.

The intent model π, presented Section 4.6.3, takes as input the history $S = S_0, \dots, S_{|S|}$ where each interaction step t corresponds to $S_t = (Q_t, a_t)$, with $Q_t = \{q_t^1, \dots, q_t^K\}$ the set of K suggestions, and a_t the user's preferred suggestion amongst the K, and predicts a Gaussian probability distribution over the possible goals. We implemented it with a Transformer-

based model. However, this architecture is not specifically designed to handle sequences of preferences. In addition, using a Gaussian distribution in a space learned by our system to generate suggestions is restrictive. First, the choice of using a normal distribution is limiting. Second, the suggestions are based on a representation of a need in the system space, not in the user space. This is a problem since we cannot realistically assume that the user and the system make their choices in the same space with same geometry.

We suppose that there exists a transformation ϕ^* that maps an information need x in the system Information Need (IN) space (e. g. a query) to a Euclidean space – called the user Information Need (IN) space – where the target intent is always closer to the selected suggestions than to the suggestions discarded by the user. In the following, we aim at learning this mapping and explore ways to leverage it to better predict $p(g|S)$. The learned representation of an information need x is denoted as $\phi(x)$.

Analytical solution To simplify, first assume that the user's Information Need space is (1) Euclidean and (2) known. We could directly partition this space with hyperplanes based on the expressed preferences. This partitioning could be used to discard the non-relevant parts of the space at each interaction step. Let $\mathcal{X}$ be the set of all possible queries and $\mathcal{Y}_t$ the current suggestion candidates at step t. At initialization, $\mathcal{Y}_0 = \mathcal{X}$, all queries of the space are candidates. At each step t, we remove all the candidates $\forall x \in \mathcal{Y}_t$ that are further away from the choice a_t of the user than any of the suggestions $s_{t,k}$. Formally, the analytical solution is:

$$\mathcal{Y}_{t+1} = \mathcal{Y}_t \backslash \{x \in \mathcal{Y}_t | \forall k = [1, \dots, K], d_\phi(x, q_t^k) < d_\phi(x, a_t) \text{ with } q_t^k \neq a_t\} \tag{5.1}$$

where $d_\phi(a, b) = d(\phi(a), \phi(b))$ and d the Euclidean distance. We can infer a distribution $p(g|S)$ for this partitioning by considering the distribution is uniform amongst candidates in the admissible area. With $\phi = \text{Id}$ where Id is the identity function, this solution implies that the user IN space and the system one are the same. To overcome this limitation, one possibility is to learn ϕ so as to minimize the pairwise loss:

$$\mathcal{L}(\mathcal{S}, g) = \sum_{t=1}^{T} \sum_{\substack{s_{t,k} \in S_t \\ s_{t,k} \neq a_t}} \max(d(\phi(g), \phi(a_t) - d(\phi(g), \phi(s_{t,k}))) + \alpha, 0) \tag{5.2}$$

with α an hyperparameter, namely the margin. However, using ϕ directly with the proposed analytical solution to compute $p(g|S)$ is difficult. First, it requires many computations to extract the remaining candidates. Second, it is not robust to noisy preferences. In the following, we propose to relax this approach.

n transformations	$n = 0$	$n = 2$	$n = 3$
Analytic	**0.935**	0.012	0.011
Bayes	0.487	0.159	0.159
LSTM	0.393	0.173	0.058
Transf	0.552	0.316	0.172
MaxEnt	0.875	**0.788**	**0.800**

Table 5.1: $P(g|S)$ with artificial data. Choices are made by the user in an euclidean representation space ($n = 0$). To simulate non-euclidean users, we report results obtained on entangled representation spaces, where a given number n of non-linear transformations are applied to items from this initial space. The higher n, the more different the user and system spaces are.

Intent model learning Given a cost $c_\phi(x, S)$ that reflects the rank of x given the user's preferences from S, we propose to consider a maximum entropy distribution that assigns probability mass to the preferred regions of the space. The latter is defined as $p(g \mid S) \propto e^{-c_\phi(g,S)}$. The model ϕ can be learned by maximizing the likelihood of $p(g|S)$ for a set of sampled user sessions, and optimized using a framework similar to (Finn, Levine, and Abbeel, 2016). ϕ is learned such that the cost of a suggestion ignored by the user is higher than the cost of a suggestion accepted by the user. Thus, the cost for a query $x \in \mathcal{X}$ is defined as:

$$c_\phi(x, S) = \sum_{t=1}^{T} \sum_{\substack{s_{t,k} \in S_t \\ s_{t,k} \neq a_t}} \max(0, 1 - d(\phi(x), \phi(s_{t,k})) + d(\phi(x), \phi(a_t))) \quad (5.3)$$

with d the Euclidean distance.

Experimental setup We compare the above intent model proposition with various methods: (1) using directly the analytic solution (2) a method that uses Bayes' theorem to decompose the probability of goals according to the user's previous actions (Yu et al., 2019), (3) two supervised models (a LSTM and a Transformer) that predict a normal distribution of the goal as in Chapter 4.

Experimental results The models were compared using three different setups. First, by unrealistically assuming that the user and the system IN space are the same space, i. e. $\phi^* = \text{Id}$, then by using transformations inducing changes between the system and the user space, implemented by n linear transformations, each followed by a tanh activation.

As expected, experiments (reported in Table 5.1) show that (1) the analytical solution only works if the user space is perfectly known. (2) Models following Bayes' theorem and supervised models performance decrease when n increases. (3) Our method has a stable performance significantly higher than the transformer-based ones. These results are quite promising for future work, it suggests that the idea of a user Information Need space is achievable and should be tested with real data.

However, these results are preliminary. The experiments were performed using low-dimensional representations of queries, while the representations generally used in IR are high-dimensional. Moreover, the results should be confirmed by testing the methods on real users.

To improve our IRnator system, we are furthermore considering several further improvements. First, we would like to give users more freedom by allowing them to not select any of the suggestions, and eventually to be able to resubmit queries during a session. Also, the retrieval system needs to be robust to user errors and adapt to the user's evolution, so we plan to train IRnator under more realistic conditions, with an imperfect and non-stationary simulated user. Finally, we want to extend the query suggestions to document suggestions to get closer to the search engine framework.

5.3 Discussions and Broader vision

The experiments of the previous section concern medium-term perspectives. We will now discuss the prospects for IR in the more distant future, and I will explain why I believe that the emergence of new interactive IR systems is fundamental.

5.3.1 Should search engines be conversational systems?

In Chapter 3, I looked at pre-trained transformer models. The latest successful transformer was ChatGPT[1] which has received a lot of media attention. This chatbot impresses with its ability to respond consistently and accurately. This clearly shows the trend of IR engines toward conversational IR systems to help users solve complex information needs.

It has been shown that the OpenAI model is able to answer rather advanced questions, on topics such as crowdfunding or alternative finance. However, when issuing more complex questions, the model is not always able to answer, although these answers are easily found with a search engine (Wenzlaff and Spaeth, 2022).

1 https://openai.com/blog/chatgpt/

The interactions enabled by chatbots may not be enough. Although users can express themselves freely, dialoguing takes time and is not always so intuitive. There are more interesting interactions that could be used, such as rating the results, highlighting some words or images, or voice interactions.

In addition to this, chatbots make the existence of sources, which may be contradictory, often invisible. Users need to know who they are reading. Scrolling around the results is also an essential part of the search process that the chatbot lacks. Thus, although ChatGPT is credible and often indistinguishable from human speech, the tool seems too limited to really allow serious and complex research.

The best of both worlds could be a fusion tool between a retrieval system and a conversational system. A first step in this direction has been made with WebGPT(Nakano et al., 2021). This model, based on GPT-3, answers open-ended questions using a text-based web browser. Thus, it can query a web browser before answering. An interesting feature of WebGPT is that it includes sources in these answers, which informs users more about them. However, I think that the addition of extra features (multiplication of sources, interactions, etc.) is essential for a truly interactive search system.

There is still no evidence that chat systems can handle complex information needs, as discussed in this thesis, so we need to integrate more sophisticated ways to model and to interact with users.

5.3.2 Glimpse of the future of IR

In this thesis, I worked on search systems that play a critical role in information access (as discussed in Chapter 1). It is worth considering the implications of our work. I now present a discussion of the existing systems and how interactive search (studied in Chapter 4) could counteract the almost unchallengeable influence of search engines.

The power of search engines on the opinions and behaviors of their users has been shown several times. Epstein and Robertson (2015) showed that by manipulating search engine results it was possible to influence the voting choices of undecided voters in elections. Pogacar et al. (2017) exposed the influences of search engines results on their opinion on the effectiveness of medical treatments.

This influence can have problematic consequences. Firstly, the results presented perpetuate, or even reinforce, existing biases, and contribute to the dissemination of false information. In addition, with personalization tools and the use of users' long-term histories, search engines have been accused of locking users into filter bubbles (Zuiderveen Borgesius et al., 2016). These filter bubbles would prevent users from being exposed to information that contradicts their opinions.

The influence of information search systems is all the more of a danger as it is possible to play on its mechanisms to highlight certain results or

modify search suggestions. For example, users of the Reddit platform manipulated Google search results by largely upvoting a post from the platform. As a result, the query "idiot" led to photos of Donald Trump, and the query "Michelle Obama" led to a racist photomontage[2].

While the results can be manipulated, the user has little control over what is returned. Interactive search can be seen as one of the levers to fight against these biases. Users could express their dissatisfaction, and even explain the reasons for it, to better guide the search system.

2 https://www.theguardian.com/us-news/2018/jul/17/trump-idiot-google-images-search

Appendix

A Résumé

Bien que les moteurs de recherche actuels fonctionnent convenablement pour des besoins d'information élémentaires avec des requêtes simples, il existe des situations pour lesquelles les résultats de recherche qu'ils retournent ne sont pas satisfaisants. Pour remédier à cela, au cours de l'histoire des systèmes de recherche, les interactions entre utilisateurs et machines ont évolué de manière significative. En effet, les données échangées entre les utilisateurs et les systèmes de recherche peuvent contenir des informations cruciales au succès de la recherche d'information.

D'autre part, l'architecture Transformer basée sur le mécanisme d'attention a permis des améliorations considérables pour plusieurs tâches de langage naturel, comme le résumé ou la traduction. Elle a rapidement été utilisée dans d'autres domaines, dont la recherche d'information (RI). Plusieurs modèles de RI ont bénéficié de la capacité de cette dernière à analyser les relations entre les termes du document et ceux des requêtes. Cependant, la majorité de ces travaux se sont concentrés sur la recherche ad hoc. L'objectif de cette thèse est d'étudier la modélisation de l'utilisateur et les interactions utilisateur-machine avec des modèles Transformers.

Les contributions de cette thèse peuvent être divisées en deux parties, celles liées à la modélisation de l'utilisateur et celles liées aux systèmes interactifs.

A.1 Échanges utilisateur-machine

Les échanges entre utilisateurs et systèmes de recherche sont essentiels. Dans cette partie, je détaille les interactions explicites et implicites, et expose les différents problèmes pour lesquels il n'existe pas de solutions satisfaisantes, et comment la modélisation des utilisateurs pourrait permettre d'améliorer les systèmes de recherche.

Tant qu'ils ne sont pas satisfaits, les utilisateurs cherchent à modifier les résultats de recherche (Huang and Efthimiadis, 2009). Pour cela, ils modifient les termes de leur requête initiale pour affiner les résultats en fonction de leurs besoins. Ils influencent aussi les résultats de recherche de manière inconsciente, via les mécanismes de personnalisation des moteurs qui prennent en compte le comportement et les données personnelles des utilisateurs.

Certains moteurs de recherche, comme Google, mettent en évidence les résultats liés à l'actualité. Campos et al. (2014). Ils peuvent également prendre en compte les coordonnées GPS des utilisateurs pour donner la priorité aux documents en lien avec leur zone géographique (Tabarcea, Gali, and Fränti, 2017). Le temps et le lieu jouent un rôle

dans les résultats présentés, de même que le type d'appareil utilisé. Par exemple, la recherche d'information mobile (Tsai et al., 2010) se consacre à la recherche d'informations à partir de smartphones. Tout ce qui précède définit l'environnement de recherche, qui joue un rôle clé dans le comportement et les attentes des utilisateurs, ainsi que dans les résultats des moteurs de recherche.

Alors que l'environnement, les moteurs de recherche et les utilisateurs déterminent les résultats de recherche, les moteurs de recherche influencent à leur tour les utilisateurs dans leur comportement de recherche. Tout d'abord, de manière évidente, les utilisateurs se conforment à l'interface affichée. Par exemple, ils expriment leur besoin sous forme de mots-clés avec les moteurs de recherche, alors qu'ils dialoguent lorsqu'il s'agit d'interagir avec un chatbot.

Plus généralement, le comportement des utilisateurs peut être analysé à travers le concept de *nudge*, élaboré par le prix Nobel d'économie Richard Thaler et le philosophe et juriste Cass Sunstein (Karlsen and Andersen, 2019).

Leurs travaux montrent que l'incitation est plus efficace que la coercition pour obtenir un comportement souhaité. Influencer le comportement des utilisateurs par un *nudge* est plus efficace que de les contraindre. Par exemple, la plupart des moteurs de recherche sont équipés d'un outil de complétion automatique de requêtes : après que les utilisateurs aient saisi les premiers caractères de leurs requêtes, les moteurs de recherche suggèrent une liste de requêtes. Cette liste de suggestions influence le parcours de recherche ultérieur des utilisateurs.

En conclusion, on ne peut pas penser aux systèmes de recherche sans considérer leurs utilisateurs, et on ne peut pas étudier ces utilisateurs sans considérer les algorithmes et les techniques qui composent les moteurs de recherche. L'objet de cette thèse est précisément d'étudier à la fois les utilisateurs et la machine lorsque les premiers cherchent à acquérir des informations. Dans cette thèse, j'étudie les modèles d'utilisateurs et le processus de recherche interactif afin d'améliorer la qualité des sessions de recherche.

A.2 Limites des systèmes actuels

Aujourd'hui, l'accès à la connaissance se fait en partie par le biais d'Internet et des moteurs de recherche. Si pour les requêtes simples, les moteurs de recherche remplissent bien leur rôle, il existe encore des cas pour lesquels les résultats de recherche ne sont pas satisfaisants. Examinons les principes de base des algorithmes des moteurs de recherche, avant de discuter de leurs limites.

À l'origine, les systèmes de recherche reposaient sur les statistiques d'occurrence des mots recherchés dans les documents. D'après cette méthode plus le nombre d'occurrences est élevé, plus le rang du docu-

ment est élevé. La limite de ces modèles est que les utilisateurs doivent utiliser exactement les mêmes mots que ceux présents dans les documents pertinents, ce problème est appelé “vocabulary mismatch”.

Bien que les moteurs de recherche aient considérablement évolué aujourd’hui, la formulation des requêtes repose toujours sur la correspondance des concepts présents dans la requête et dans le document, sans toujours se soucier du sens global de la requête. Par conséquent, cela ne satisfait pas toujours le besoin d’information. Par exemple, un utilisateur qui cherche “Quel animal ne mange pas de laitue ?” n’obtiendra que des pages sur les animaux dont le régime alimentaire contient de la laitue. En effet, les documents sur les animaux mentionnent ce qu’ils mangent mais pas ce qu’ils ne mangent pas.

Comme deuxième exemple des limites des moteurs de recherche, considérons les cas où le besoin d’information doit être décomposé en plusieurs requêtes. Dans ces cas, les moteurs de recherche échouent généralement. Par exemple, Google trouve des résultats pertinents pour les requêtes “héros film amoureux d’une AI” et “couleur film *Her*”,i. e. des pages qui traitent du film *Her* dans le premier cas, et les pages qui décrivent la prépondérance du rouge dans le film *Her* dans le second. En revanche, lors de la recherche de “couleur film héros amoureux d’une AI”, le moteur de recherche n’envoie que les documents qui évoquent le film et non la couleur rouge. Dans cet exemple, pour atteindre leur objectif, les utilisateurs doivent effectuer deux requêtes. Pour de tels besoins d’information, les utilisateurs essaieront probablement de nouvelles requêtes pour obtenir les informations qu’ils recherchent.

Plus généralement, Carmel and Yom-Tov (2010) ont établi une taxonomie des situations qui peuvent faire échouer un système de recherche, et donc conduire les utilisateurs à réitérer le processus. Ces situations sont regroupées en deux catégories, celles où les systèmes ne parviennent pas à identifier et à couvrir tous les aspects du sujet et celles où il analyse de manière incorrecte le sens de la requête. Dans la première catégorie, les systèmes font ressortir un aspect non pertinent ou alors omettent un aspect pertinent. C’est le cas de la requête “couleur film héros amoureux d’une AI” pour laquelle l’aspect “couleur” a échappé à Google. La deuxième catégorie comprend les échecs d’identification des relations entre les termes, des relations de proximité ou de la généralisation d’un terme (par exemple, l’extension du mot “Europe” à un pays spécifique comme la “France”). La requête “quel animal ne mange pas de laitue ?” entre dans cette catégorie : le moteur de recherche considéré, Google, ne prend pas en compte la négation. Les interactions avec la machine pourraient permettre aux utilisateurs de montrer, de manière indirecte et non explicite, qu’ils ne sont pas satisfaits des résultats.

A.3 Influences mutuelles entre utilisateurs et machines

Les interactions entre utilisateurs et machines ont beaucoup évolué depuis les débuts des moteurs de recherche. Les utilisateurs étaient assez passifs et interagissaient très peu avec le moteur de recherche. Les seules interactions possibles étaient celles avec les requêtes et les résultats de recherche renvoyés. Avec les années, et l'intérêt croissant des chercheurs et des informaticiens pour la recherche d'information, des outils permettant d'augmenter la qualité et la quantité de ces interactions ont été développés. C'est par exemple le cas de l'outil de suggestion de requêtes, ou de l'outil de complétion automatique de requêtes.

Intrinsèquement, le système transmet des informations - les documents retournés - aux utilisateurs, mais ces derniers transmettent également un certain nombre de feedbacks et d'informations de manière indirecte au système. Par exemple, la reformulation d'une requête constitue une interaction par laquelle les utilisateurs modifient leur requête pour obtenir des résultats plus pertinents. D'ailleurs, relativement tôt dans l'histoire de la RI, on a pris en considération l'ajout et la suppression de termes entre deux requêtes consécutives (Bruza and Dennis, 1997).

D'autres actions peuvent être prises en compte pour améliorer le système. Par exemple, les liens cliqués pendant une session de recherche aident à comprendre la trajectoire de la session de recherche (Mei, Zhou, and Church, 2008), les documents cliqués et ceux ignorés sur la page des résultats sont une forme de feedback (Ahmad, Chang, and Wang, 2018; Ahmad, Chang, and Wang, 2019), et même le mouvement de la souris de l'ordinateur (Diaz et al., 2013), ou le mouvement des yeux sur l'écran (eye-tracking process) (Cutrell and Guan, 2007) fournissent des informations.

A.4 Modèles utilisateurs

Comme nous l'avons vu dans la section précédente, les machines et les utilisateurs échangent donc des informations entre eux, et ces informations peuvent être décisives pour améliorer le processus de recherche. Nous montrons maintenant comment les modèles d'utilisateur peuvent aider à tirer parti de ces interactions.

Lors des sessions de recherche d'information, les utilisateurs commencent par une requête initiale, puis effectuent un ensemble d'actions : ils étudient les documents retournés, cliquent sur certains d'entre eux, peuvent poursuivre leur navigation sur d'autres pages web, formulent de nouvelles requêtes, et réitèrent jusqu'à ce qu'ils soient satisfaits ou abandonnent leur recherche. La modélisation des utilisateurs consiste à construire un modèle qui prédit tout ou une partie partie de ces actions. Brusilovsky and Tasso (2004) justifient la nécessité de la modélisation des utilisateurs de cette manière : "le système de recherche

d'information doit suivre dans le temps la manière dont l'utilisateur comprend et formule ses besoins en information". La modélisation de l'utilisateur pourrait permettre au système de s'adapter aux besoins spécifiques des utilisateurs. Cette modélisation peut être enrichie en intégrant d'autres données spécifiques aux utilisateurs telles que leur position géographique (Tabarcea, Gali, and Fränti, 2017), le dispositif utilisé (Tsai et al., 2010), leurs recherches précédentes (Sordoni et al., 2015), l'ensemble des langues admissibles, etc. L'amélioration de la modélisation des utilisateurs a de nombreuses applications potentielles que nous détaillons ci-dessous.

Améliorer les interactions Les moteurs de recherche et les utilisateurs interagissent au cours du processus de recherche. D'une part, les utilisateurs envoient des requêtes et d'autre part, les moteurs de recherche répondent avec une liste de documents. Pourtant, ce sont les utilisateurs qui initient les interactions, par la reformulation de la requête et la navigation dans la page de résultats, qui peuvent être utilisées comme un feedback implicite. Bien qu'aujourd'hui les moteurs de recherche ne se contentent pas de classer les documents. Ils disposent de plusieurs stratégies pour améliorer l'expérience de recherche, telles que la mise en évidence des réponses, les suggestions de recherche, ou l'agrégation des résultats. Des interactions plus riches entre les deux parties pourraient améliorer les expériences de recherche. La prédiction des prochaines requêtes des utilisateurs, qui est une forme de modélisation, est utilisée par les outils de suggestions de requêtes et d'autocomplétion (Sordoni et al., 2015; Dehghani et al., 2017; Mustar, Lamprier, and Piwowarski, 2021). Ces outils sont particulièrement importants dans le cas de recherches complexes pour guider les utilisateurs et leur faire gagner du temps. Une façon de rendre les interactions plus pertinentes est de modéliser correctement les utilisateurs pendant leur session de recherche. Cela a aussi été fait, par exemple, en prédisant l'intention des utilisateurs puis en leur posant des questions de clarification sur leur intention (Dhole, 2020).

Améliorer les interfaces L'interface classique des moteurs de recherche avec des résultats présentés sous forme de liste de documents est remise en question. Ainsi, certains moteurs de recherche proposent désormais des résultats de recherche verticaux qui consistent à agréger des résultats de différents types (texte, image, vidéo, news...) dans une interface ergonomique qui permet aux utilisateurs de trouver plus rapidement l'information qu'ils recherchent (Zhou et al., 2013).

Les interfaces présentées aux utilisateurs peuvent également être personnalisées grâce à des modèles d'utilisateurs. On parle alors d'applications Web adaptatives. Un travail représentatif dans ce domaine est celui de Lohmann, Kaltz, and Ziegler (2006) qui propose une approche pour prendre en compte les informations des utilisateurs dans

la façon dont leur interface graphique est affichée. Cependant, ils montrent qu'une adaptation ratée perturbe l'utilisateur, ce qui limite les propositions d'adaptation et les tâches pour lesquelles l'interface peut être personnalisée.

Entraînement avec des utilisateurs simulés Les systèmes de RI interactifs sont aujourd'hui des systèmes paramétriques, ils peuvent être entraînés en simulant les décisions des utilisateurs telles que leurs requêtes, leurs clics et leur satisfaction globale. L'entraînement de tels algorithmes nécessite des milliers d'interactions, ce qui est impossible à obtenir. De même, leur évaluation sur un nombre suffisant d'utilisateurs réels est très coûteuse.

Améliorer les métriques Les méthodes de RI qui visent à améliorer certaines métriques, telles que la précision ou le rappel, sont dites system-centered. À l'inverse, les recherches qui utilisent des utilisateurs réels pour évaluer leurs performances sont dites user-centered. Un écart a été constaté entre ces deux types de mesures, car il n'y a pas toujours de corrélation entre les mesures et la satisfaction des utilisateurs (Liu et al., 2019a). Cet écart montre que les mesures automatiques ne sont pas toujours satisfaisantes. Par conséquent, l'entraînement ou l'évaluation d'un modèle sur ces métriques n'est pas forcément très fructueux. En revanche, un modèle d'utilisateur parfait pourrait prédire si un utilisateur est satisfait ou non. Un utilisateur simulé pourrait être utilisé pendant l'entraînement pour définir une *reward*, ou pendant l'évaluation du modèle, à un coût bien inférieur à celui d'une évaluation humaine coûteuse (Dupret and Piwowarski, 2013).

Dans cette thèse, je travaille sur la modélisation utilisateur afin d'améliorer les interactions utilisateur-machine, c'est l'angle d'attaque qui me semble le plus prometteur pour améliorer le processus de recherche classique. La modélisation des utilisateurs en RI va au-delà des moteurs de recherche. Elle peut être appliquée à toutes les situations dans lesquelles les humains et les machines interagissent. Par exemple, dans le contexte de la modélisation de la résolution de tâches, comme la traduction automatique ou la comptabilité, les logiciels pourraient bénéficier de ces méthodes en anticipant le comportement de l'utilisateur ou en demandant explicitement des clarifications. Étant donné l'utilisation importante des machines pour résoudre des tâches aujourd'hui, il est intéressant de modéliser nos interactions avec elles.

A.5 Contributions

Dans cette thèse, j'ai étudié les interactions entre les utilisateurs et les moteurs de recherche lors de tâches de recherche complexes. Je me suis d'abord attaché à modéliser les utilisateurs pendant une tâche de

recherche, avant d'étudier de nouvelles façons d'interagir pour mieux satisfaire les utilisateurs.

Modèle de l'utilisateur Ma première contribution a été l'analyse des modèles de prédiction de requêtes, qui servent également d'outil de suggestion de requêtes. Cette tâche est intéressante car elle nécessite d'anticiper le comportement des utilisateurs, mais elle est réalisable grâce aux logs des moteurs de recherche disponibles. J'ai comparé des modèles spécifiques à cette tâche ainsi que des générateurs de langage génériques (par exemple, un transformateur). Ces analyses démontrent que bien que les modèles avec une architecture spécifique à la tâche soient performants, les transformateurs pré-entraînés de grande taille sont plus robustes. Une analyse plus poussée du processus de génération montre qu'ils sont capables de récupérer la structure d'une session de recherche web et sont également de bons modèles de langue. Ces modèles peuvent donc être utilisés pour des systèmes interactifs.

Système interactif Après ce travail centré sur l'utilisateur, j'ai étudié les systèmes d'interaction existants. À ce jour, il n'existe pas de cadre standard en RI. Alors qu'il existe de nombreux travaux sur les systèmes de recommandation interactifs, où les données sont plus structurées et limitées, les systèmes de recherche interactifs sont difficiles à modéliser. De plus, l'évaluation de leurs performances est difficile. Pour cette raison, nous proposons un paradigme dans lequel le système et les utilisateurs interagissent dans un cadre limité. Inspiré par le système Akinator, IRnator cherche à déterminer l'objectif de l'utilisateur en lui demandant de rendre ses préférences explicites. Bien que le cadre soit restreint, il permet une formalisation stricte du problème et peut être étendu facilement.

Bibliography

Abbate, Janet (1999). *Inventing the Internet*. Cambridge, MA, USA: MIT Press.

Abolghasemi, Amin, Suzan Verberne, and Leif Azzopardi (2022). "Improving BERT-based Query-by-Document Retrieval with Multi-task Optimization." In: *Advances in Information Retrieval - 44th European Conference on IR Research, ECIR 2022, Stavanger, Norway, April 10-14, 2022, Proceedings, Part II*. Ed. by Matthias Hagen, Suzan Verberne, Craig Macdonald, Christin Seifert, Krisztian Balog, Kjetil Nørvåg, and Vinay Setty. Vol. 13186. Lecture Notes in Computer Science. Springer, pp. 3–12.

Abri, Sara, Rayan Abri, and Salih Cetin (2020). "A Classification on Different Aspects of User Modelling in Personalized Web Search." In: *Proceedings of the 4th International Conference on Natural Language Processing and Information Retrieval*. NLPIR 2020. Seoul, Republic of Korea: Association for Computing Machinery, pp. 194–199.

Adamopoulou, Eleni and Lefteris Moussiades (2020). "An Overview of Chatbot Technology." In: *Artificial Intelligence Applications and Innovations*. Ed. by Ilias Maglogiannis, Lazaros Iliadis, and Elias Pimenidis. Cham: Springer International Publishing, pp. 373–383.

Ahmad, Subutai and Jeff Hawkins (2015). *Properties of Sparse Distributed Representations and their Application to Hierarchical Temporal Memory*.

Ahmad, Wasi Uddin, Kai-Wei Chang, and Hongning Wang (2018). "Multi-Task Learning for Document Ranking and Query Suggestion." In: *International Conference on Learning Representations*.

Ahmad, Wasi Uddin, Kai-Wei Chang, and Hongning Wang (2019). "Context Attentive Document Ranking and Query Suggestion." In: *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR'19. New York, NY, USA: ACM, pp. 385–394.

Alfaro-Paredes, Edwin, Leonardo Alfaro-Carrasco, and Willy Ugarte (2021). "Query by Humming for Song Identification Using Voice Isolation." In: *Advances and Trends in Artificial Intelligence. From Theory to Practice: 34th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2021, Kuala Lumpur, Malaysia, July 26–29, 2021, Proceedings, Part II*. Kuala Lumpur, Malaysia: Springer-Verlag, pp. 323–334.

Aliannejadi, Mohammad, Hamed Zamani, Fabio Crestani, and W. Bruce Croft (2019). "Asking Clarifying Questions in Open-Domain Information-

Seeking Conversations." In: *The 42nd International ACM SIGIR Conference on Research & Development in Information Retrieval*. SIGIR'19. New York, NY, USA: ACM, pp. 475–484.

Allen, Garrett, Benjamin L Peterson, Dhanush kumar Ratakonda, Mostofa Najmus Sakib, Jerry Alan Fails, Casey Kennington, Katherine Landau Wright, and Maria Soledad Pera (2021). "Engage!: Co-Designing Search Engine Result Pages to Foster Interactions." In: *Interaction Design and Children*. IDC '21. Athens, Greece: Association for Computing Machinery, pp. 583–587.

An, Xiangdong, Jimmy Xiangji Huang, and Yuqi Wang (2020). "Chapter Twelve - Diversity and novelty in biomedical information retrieval." In: *Biomedical Information Technology (Second Edition)*. Ed. by David Dagan Feng. Second Edition. Biomedical Engineering. Academic Press, pp. 369–396.

Aqle, Aboubakr, Kamran Khowaja, and Dena Al-Thani (2020). "Preliminary Evaluation of Interactive Search Engine Interface for Visually Impaired Users." In: *IEEE Access* 8, pp. 45061–45070.

Arjovsky, Martin, Soumith Chintala, and Léon Bottou (2017). "Wasserstein Generative Adversarial Networks." In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML'17. Sydney, NSW, Australia: JMLR.org, pp. 214–223.

Asri, Layla El, Jing He, and Kaheer Suleman (2016). "A Sequence-to-Sequence Model for User Simulation in Spoken Dialogue Systems." In: *Proc. Interspeech 2016*, pp. 1151–1155.

Aswani, Reema, SP Ghrera, Satish Chandra, and Arpan Kumar Kar (2021). "A hybrid evolutionary approach for identifying spam websites for search engine marketing." In: *Evolutionary Intelligence* 14.4, pp. 1803–1815.

Barth-Maron, Gabriel, Matthew W. Hoffman, David Budden, Will Dabney, Dan Horgan, Dhruva TB, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap (2018). "Distributional Policy Gradients." In: *International Conference on Learning Representations*.

Baskaya, Feza, Heikki Keskustalo, and Kalervo Järvelin (2013). "Modeling Behavioral Factors Ininteractive Information Retrieval." In: *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management*. CIKM '13. San Francisco, California, USA: Association for Computing Machinery, pp. 2297–2302.

Beck, Joseph E., Peng Jia, June Sison, and Jack Mostow (2003). "Predicting Student Help-Request Behavior in an Intelligent Tutor for Reading." In: *User Modeling 2003*. Ed. by Peter Brusilovsky, Albert Corbett, and Fiorella de Rosis. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 303–312.

Belkin, N.J. (1984). "Cognitive models and information transfer." In: *Social Science Information Studies* 4.2. Special Issue Seminar on the Psychological Aspects of Information Searching, pp. 111–129.

Beltagy, Iz, Matthew E. Peters, and Arman Cohan (2020). "Longformer: The Long-Document Transformer." In: arXiv: 2004.05150.

Bhatia, Sumit, Debapriyo Majumdar, and Prasenjit Mitra (2011). "Query Suggestions in the Absence of Query Logs." In: *The 34th International ACM SIGIR Conference on Research & Development in Information Retrieval*. SIGIR '11. New York, NY, USA: ACM, pp. 795–804.

Bhattacharya, B., I. Burhanuddin, A. Sancheti, and K. Satya (2017). "Intent-Aware Contextual Recommendation System." In: *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*. Los Alamitos, CA, USA: IEEE Computer Society, pp. 1–8.

Bidel, Sylvain, Laurent Lemoine, Frédéric Piat, Thierry Artières, and Patrick Gallinari (June 2003). "Statistical machine learning for tracking hypermedia user behaviour." In: *MLIRUM'03 - 2nd Workshop on Machine Learning, Information Retrieval and User Modeling*. Pittsburgh, PA, United States.

Boldi, Paolo, Francesco Bonchi, Carlos Castillo, Debora Donato, and Sebastiano Vigna (2009). "Query Suggestions Using Query-Flow Graphs." In: *Proceedings of the 2009 Workshop on Web Search Click Data*. WSCD '09. New York, NY, USA: ACM, pp. 56–63.

Bonchi, Francesco, Raffaele Perego, Fabrizio Silvestri, Hossein Vahabi, and Rossano Venturini (2012). "Efficient Query Recommendations in the Long Tail via Center-Piece Subgraphs." In: *The 35th International ACM SIGIR Conference on Research & Development in Information Retrieval*. SIGIR '12. New York, NY, USA: ACM, pp. 345–354.

Borisov, Alexey, Ilya Markov, Maarten de Rijke, and Pavel Serdyukov (2016). "A Neural Click Model for Web Search." In: *Proceedings of the 25th International Conference on World Wide Web*. WWW '16. Montréal, Québec, Canada: International World Wide Web Conferences Steering Committee, pp. 531–541.

Borisov, Alexey, Martijn Wardenaar, Ilya Markov, and Maarten de Rijke (2018). "A Click Sequence Model for Web Search." In: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. SIGIR '18. Ann Arbor, MI, USA: Association for Computing Machinery, pp. 45–54.

Borlund, Pia (Oct. 2009). "User-Centred Evaluation of Information Retrieval Systems." In: pp. 21 –37.

Brajnik, G., G. Guida, and C. Tasso (1990). "User modeling in expert man-machine interfaces: a case study in intelligent information retrieval." In: *IEEE Transactions on Systems, Man, and Cybernetics* 20.1, pp. 166–185.

Brajnik, Giorgio, Giovanni Guida, and Carlo Tasso (1987). "User modeling in intelligent information retrieval." In: *Information Processing & Management* 23.4. Special Issue: Artificial Intelligence and Information Retrieval, pp. 305–320.

Brin, Sergey and Lawrence Page (1998). "The anatomy of a large-scale hypertextual Web search engine." In: *Computer Networks and ISDN*

Systems 30.1. Proceedings of the Seventh International World Wide Web Conference, pp. 107–117.

Broccolo, Daniele, Lorenzo Marcon, Franco Maria Nardini, Raffaele Perego, and Fabrizio Silvestri (Mar. 2012). "Generating Suggestions for Queries in the Long Tail with an Inverted Index." In: *Information Processing & Management* 48.2, pp. 326–339.

Broder, Andrei (2002). "A Taxonomy of Web Search." In: *SIGIR Forum* 36.2, pp. 3–10.

Brown, Tom et al. (2020). "Language Models are Few-Shot Learners." In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., pp. 1877–1901.

Brunner, Gino, Yang Liu, Damian Pascual, Oliver Richter, Massimiliano Ciaramita, and Roger Wattenhofer (2020). "On Identifiability in Transformers." In: *International Conference on Learning Representations*.

Brusilovsky, Peter and Carlo Tasso (2004). "Preface to special issue on user modeling for web information retrieval." In: *User Modeling and User-Adapted Interaction* 14.2, pp. 147–157.

Bruza, P. D. and S. Dennis (1997). "Query ReFormulation on the Internet: Empirical Data and the Hyperindex Search Engine." In: *Computer-Assisted Information Searching on Internet*. RIAO '97. Montreal, Quebec, Canada: Le Centre De Hautes Etudes Internationales d'Informatique Documentaire, pp. 488–499.

Burgener, R (2006). "20q: The neural network mind reader." In: *Goddard Space Flight Center Engineering Colloquium*.

Câmara, Arthur, David Maxwell, and Claudia Hauff (2022). "Searching, Learning, and Subtopic Ordering: A Simulation-Based Analysis." In: *Advances in Information Retrieval*. Ed. by Matthias Hagen, Suzan Verberne, Craig Macdonald, Christin Seifert, Krisztian Balog, Kjetil Nørvåg, and Vinay Setty. Cham: Springer International Publishing, pp. 142–156.

Campos, Ricardo, Gaël Dias, Alípio M. Jorge, and Adam Jatowt (2014). "Survey of Temporal Information Retrieval and Related Applications." In: *ACM Comput. Surv.* 47.2.

Cao, Huanhuan, Daxin Jiang, Jian Pei, Qi He, Zhen Liao, Enhong Chen, and Hang Li (2008). "Context-Aware Query Suggestion by Mining Click-through and Session Data." In: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '08. New York, NY, USA: ACM, pp. 875–883.

Cao, Yang Trista, Sudha Rao, and Hal Daumé III (Aug. 2019). "Controlling the Specificity of Clarification Question Generation." In: *Proceedings of the 2019 Workshop on Widening NLP*. Florence, Italy: Association for Computational Linguistics, pp. 53–56.

Carmel, David and Elad Yom-Tov (2010). "Estimating the Query Difficulty for Information Retrieval." In: *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Re-*

trieval. SIGIR '10. Geneva, Switzerland: Association for Computing Machinery, p. 911.

Carterette, Ben, Paul Clough, Mark Hall, Evangelos Kanoulas, and Mark Sanderson (2016). "Evaluating Retrieval over Sessions: The TREC Session Track 2011-2014." In: *The 39th International ACM SIGIR Conference on Research & Development in Information Retrieval*. SIGIR '16. New York, NY, USA: ACM, pp. 685–688.

Ceruzzi, Paul E. (2003). *A History of Modern Computing*. 2nd ed. Language, Speech, and Communication. Cambridge, MA: MIT Press.

Chandramohan, Senthilkumar, Matthieu Geist, Fabrice Lefèvre, and Olivier Pietquin (Aug. 2011). "User Simulation in Dialogue Systems using Inverse Reinforcement Learning." In: *Interspeech 2011*. Florence, Italy, pp. 1025–1028.

Chen, Cen, Chilin Fu, Xu Hu, Xiaolu Zhang, Jun Zhou, Xiaolong Li, and Forrest Sheng Bao (2019). "Reinforcement Learning for User Intent Prediction in Customer Service Bots." In: *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR'19. Paris, France: Association for Computing Machinery, pp. 1265–1268.

Chen, Jia, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma (2020). "A Context-Aware Click Model for Web Search." In: *Proceedings of the 13th International Conference on Web Search and Data Mining*. WSDM '20. Houston, TX, USA: Association for Computing Machinery, pp. 88–96.

Chen, Wanyu, Fei Cai, Honghui Chen, and Maarten de Rijke (2018). "Attention-Based Hierarchical Neural Query Suggestion." In: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. SIGIR '18. New York, NY, USA: ACM, pp. 1093–1096.

Cho, Kyunghyun, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio (Oct. 2014). "On the Properties of Neural Machine Translation: Encoder–Decoder Approaches." In: *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Doha, Qatar: Association for Computational Linguistics, pp. 103–111.

Christakopoulou, Konstantina, Alex Beutel, Rui Li, Sagar Jain, and Ed H Chi (2018). "Q&R: A two-stage approach toward interactive recommendation." In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 139–148.

Clark, Kevin, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning (2019). "What Does BERT Look at? An Analysis of BERT's Attention." In: *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Florence, Italy: ACL, pp. 276–286.

Cleverdon, Cyril W (1960). "The aslib cranfield research project on the comparative efficiency of indexing systems." In: *Aslib Proceedings*. MCB UP Ltd.

Cleverdon, Cyril (1967). "The Cranfield tests on index language devices." In: *Aslib proceedings*. MCB UP Ltd.

Cleverdon, Cyril, Jack Mills, and Michael Keen (1966). "Factors determining the performance of indexing systems." In.

Conneau, Alexis and Guillaume Lample (2019). "Cross-lingual Language Model Pretraining." In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc.

Cooke, M.D. (2011). "CHAPTER SIX. Chemical structure handling by computer." In: *Information Sources in Chemistry*. Ed. by R. T. Bottle and J. F. B. Rowland. Berlin, Boston: K. G. Saur, pp. 105–116.

Crook, Paul A and Alex Marin (2017). "Sequence to Sequence Modeling for User Simulation in Dialog Systems." In: *INTERSPEECH*, pp. 1706–1710.

Cutrell, Edward and Zhiwei Guan (2007). "What Are You Looking for? An Eye-Tracking Study of Information Usage in Web Search." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '07. San Jose, California, USA: Association for Computing Machinery, pp. 407–416.

Dai, Xinyi, Jianghao Lin, Weinan Zhang, Shuai Li, Weiwen Liu, Ruiming Tang, Xiuqiang He, Jianye Hao, Jun Wang, and Yong Yu (2021). "An Adversarial Imitation Click Model for Information Retrieval." In: *Proceedings of the Web Conference 2021*. WWW '21. Ljubljana, Slovenia: Association for Computing Machinery, pp. 1809–1820.

Dai, Zhuyun and Jamie Callan (2020). "Context-Aware Document Term Weighting for Ad-Hoc Search." In: *Proceedings of The Web Conference 2020*. WWW '20. Taipei, Taiwan: Association for Computing Machinery, pp. 1897–1907.

Dai, Zihang, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov (July 2019). "Transformer-XL: Attentive Language Models beyond a Fixed-Length Context"." In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: ACL, pp. 2978–2988.

Datcu, M. and K. Seidel (1999). "Query by image content and information mining." In: *IEEE 1999 International Geoscience and Remote Sensing Symposium. IGARSS'99 (Cat. No.99CH36293)*. Vol. 2, 1335–1337 vol.2.

Datta, Ritendra, Dhiraj Joshi, Jia Li, and James Z. Wang (2008). "Image Retrieval: Ideas, Influences, and Trends of the New Age." In: *ACM Comput. Surv.* 40.2.

Dehghani, Mostafa, Sascha Rothe, Enrique Alfonseca, and Pascal Fleury (2017). "Learning to Attend, Copy, and Generate for Session-Based Query Suggestion." In: *Proceedings of the 26th ACM International on Conference on Information and Knowledge Management*. CIKM '17. New York, NY, USA: ACM, pp. 1747–1756.

Delgado, Alejandro, SkoT McDonald, Ning Xu, Charalampos Saitis, and Mark Sandler (2021). "Learning Models for Query by Vocal Percussion: A Comparative Study." In: *arXiv preprint arXiv:2110.09223*.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (June 2019). "BERT: Pre-training of Deep Bidirectional Transformers

for Language Understanding." In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Vol. 1. Minneapolis, Minnesota: ACL, pp. 4171–4186.

Dhingra, Bhuwan, Christopher Shallue, Mohammad Norouzi, Andrew Dai, and George Dahl (June 2018). "Embedding Text in Hyperbolic Spaces." In: *Proceedings of the Twelfth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-12)*. New Orleans, Louisiana, USA: Association for Computational Linguistics, pp. 59–69.

Dhole, Kaustubh D. (2020). *Resolving Intent Ambiguities by Retrieving Discriminative Clarifying Questions*. arXiv: 2008.07559 [cs.AI].

Diaz, Fernando, Ryen White, Georg Buscher, and Dan Liebling (2013). "Robust Models of Mouse Movement on Dynamic Web Search Results Pages." In: *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management*. CIKM '13. San Francisco, California, USA: Association for Computing Machinery, pp. 1451–1460.

Ding, Yiming, Carlos Florensa, Pieter Abbeel, and Mariano Phielipp (2019). "Goal-conditioned imitation learning." In: *Advances in Neural Information Processing Systems*, pp. 15298–15309.

Dumais, Susan (1990). "Indexing by Latent Semantic Analysis." In: *Journal of the American Society for Information Science* 41.6, pp. 391–407.

Dupret, Georges and Benjamin Piwowarski (2013). "Model Based Comparison of Discounted Cumulative Gain and Average Precision." In: *J. of Discrete Algorithms* 18, pp. 49–62.

Epstein, Robert and Ronald E. Robertson (2015). "The search engine manipulation effect (SEME) and its possible impact on the outcomes of elections." In: *Proceedings of the National Academy of Sciences* 112.33, E4512–E4521. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.1419828112.

Finn, Chelsea, Sergey Levine, and Pieter Abbeel (2016). "Guided cost learning: Deep inverse optimal control via policy optimization." In: *International conference on machine learning*, pp. 49–58.

Foerster, Jakob, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson (May 2017). "Counterfactual Multi-Agent Policy Gradients." en. In: *arXiv:1705.08926 [cs]*. arXiv: 1705.08926.

Forgues, Gabriel, Joelle Pineau, Jean-Marie Larchevêque, and Réal Tremblay (2014). "Bootstrapping dialog systems with word embeddings." In: *Nips, modern machine learning and natural language processing workshop*. Vol. 2. NIPS'14. Red Hook, NY, USA: Curran Associates Inc.

Formal, Thibault, Benjamin Piwowarski, and Stéphane Clinchant (2021). "SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking." In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '21. Virtual Event, Canada: Association for Computing Machinery, pp. 2288–2292.

Frej, Jibril, Philippe Mulhem, Didier Schwab, and Jean-Pierre Chevallet (2020). "Learning Term Discrimination." In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '20. Virtual Event, China: Association for Computing Machinery, pp. 1993–1996.

Gabeur, Valentin, Chen Sun, Karteek Alahari, and Cordelia Schmid (Aug. 2020). "Multi-modal Transformer for Video Retrieval." In: *ECCV 2020 - European Conference on Computer Vision*. Vol. 12349. Lecture Notes in Computer Science. Glasgow, United Kingdom: Springer, pp. 214–229.

Gao, Jianfeng, Michel Galley, and Lihong Li (July 2018). "Neural Approaches to Conversational AI." In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*. Melbourne, Australia: Association for Computational Linguistics, pp. 2–7.

Gao, Luyu, Zhuyun Dai, and Jamie Callan (June 2021). "COIL: Revisit Exact Lexical Match in Information Retrieval with Contextualized Inverted List." In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, pp. 3030–3042.

Gao, Yongqing, Guangda Huzhang, Weijie Shen, Yawen Liu, Wen-Ji Zhou, Qing Da, and Yang Yu (2021). *Imitate TheWorld: A Search Engine Simulation Platform*.

Garg, Vikas K., Inderjit S. Dhillon, and Hsiang-Fu Yu (2019). "Multiresolution Transformer Networks: Recurrence is Not Essential for Modeling Hierarchical Structure." In: arXiv: 1908.10408.

Geng, Qian, Ziang Chuai, and Jian Jin (2022). "Webpage retrieval based on query by example for think tank construction." In: *Information Processing & Management* 59.1, p. 102767.

Gershoum, Anatole (1981). "Figuring out What the User Wants: Steps toward an Automatic Yellow Pages Assistant." In: *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 1*. IJCAI'81. Vancouver, BC, Canada: Morgan Kaufmann Publishers Inc., pp. 423–425.

Goyal, Anirudh, Alex Lamb, Ying Zhang, Saizheng Zhang, Aaron Courville, and Yoshua Bengio (2016). "Professor Forcing: A New Algorithm for Training Recurrent Networks." In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. NIPS'16. Barcelona, Spain: Curran Associates Inc., pp. 4608–4616.

Graves, Alex (2013). "Generating Sequences With Recurrent Neural Networks." In: *CoRR* abs/1308.0850. arXiv: 1308.0850.

Groza, Adrian and Loredana Coroama (2019). "A mentalist agent for identifying characters using dynamic query strategies." In: *2019 IEEE 15th International Conference on Intelligent Computer Communication and Processing (ICCP)*. IEEE, pp. 319–326.

Gulrajani, Ishaan, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville (2017). "Improved Training of Wasserstein GANs." In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., pp. 5769–5779.

Gür, Izzeddin, Dilek Hakkani-Tür, Gokhan Tür, and Pararth Shah (2018). "User modeling for task oriented dialogues." In: *2018 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, pp. 900–906.

Gyöngyi, Zoltán, Hector Garcia-Molina, and Jan Pedersen (2004). "Combating Web Spam with Trustrank." In: *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*. VLDB '04. Toronto, Canada: VLDB Endowment, pp. 576–587.

Hagen, Matthias, Jakob Gomoll, Anna Beyer, and Benno Stein (2013). "From Search Session Detection to Search Mission Detection." In: *Proceedings of the 10th Conference on Open Research Areas in Information Retrieval*. OAIR '13. Paris, FRA: Centre de hautes études internationales d'informatique documentaire, pp. 85–92.

Harman, Donna (1993). "Overview of the First TREC Conference." In: *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '93. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, pp. 36–47.

– (2010). "Is the Cranfield Paradigm Outdated?" In: *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '10. Geneva, Switzerland: Association for Computing Machinery, p. 1.

Harman, Donna et al. (2019). "Information retrieval: the early years." In: *Foundations and Trends® in Information Retrieval* 13.5, pp. 425–577.

Hassan Awadallah, Ahmed, Ryen W. White, Patrick Pantel, Susan T. Dumais, and Yi-Min Wang (2014). "Supporting Complex Search Tasks." In: *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. CIKM '14. New York, NY, USA: ACM, pp. 829–838.

He, He, Derek Chen, Anusha Balakrishnan, and Percy Liang (2018). "Decoupling Strategy and Generation in Negotiation Dialogues." In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, pp. 2333–2343.

He, Qi, Daxin Jiang, Zhen Liao, Steven C. H. Hoi, Kuiyu Chang, Ee-Peng Lim, and Hang Li (2009). "Web Query Recommendation via Sequential Query Prediction." In: *Proceedings of the 2009 IEEE International Conference on Data Engineering*. ICDE '09. Washington, DC, USA: IEEE Computer Society, pp. 1443–1454.

Hearst, Marti A. (2009). *Search User Interfaces*. Cambridge University Press.

Ho, Jonathan and Stefano Ermon (2016). "Generative Adversarial Imitation Learning." In: *Proceedings of the 30th International Conference*

on Neural Information Processing Systems. NIPS'16. Barcelona, Spain: Curran Associates Inc., pp. 4572–4580.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long Short-Term Memory." In: *Neural Comput.* 9.8, pp. 1735–1780.

Howard, Jeremy and Sebastian Ruder (July 2018). "Universal Language Model Fine-tuning for Text Classification." In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*. Vol. 1: Long Papers. Melbourne, Australia: ACL, pp. 328–339.

Hu, Huang, Xianchao Wu, Bingfeng Luo, Chongyang Tao, Can Xu, Wei Wu, and Zhan Chen (2018). "Playing 20 question game with policy-based reinforcement learning." In: *arXiv preprint arXiv:1808.07645*.

Huang, Chien-Kang, Lee-Feng Chien, and Yen-Jen Oyang (May 2003). "Relevant Term Suggestion in Interactive Web Search Based on Contextual Information in Query Session Logs." In: *Journal of the American Society for Information Science and Technology* 54.7, pp. 638–649.

Huang, Jeff and Efthimis N. Efthimiadis (2009). "Analyzing and Evaluating Query Reformulation Strategies in Web Search Logs." In: *Proceedings of the 18th ACM Conference on Information and Knowledge Management*. CIKM '09. Hong Kong, China: Association for Computing Machinery, pp. 77–86.

Information Science in Transition (2009). Facet.

Jaeger, Herbert (2002). "Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the echo state network approach." In.

Jain, Alpa, Umut Ozertem, and Emre Velipasaoglu (2011a). "Synthesizing High Utility Suggestions for Rare Web Search Queries." In: *The 34th International ACM SIGIR Conference on Research & Development in Information Retrieval*. SIGIR '11. New York, NY, USA: ACM, pp. 805–814.

– (2011b). "Synthesizing High Utility Suggestions for Rare Web Search Queries." In: *The 34th International ACM SIGIR Conference on Research & Development in Information Retrieval*. SIGIR '11. New York, NY, USA: ACM, pp. 805–814.

Jang, Kyoung-Rok, Junmo Kang, Giwon Hong, Sung-Hyon Myaeng, Joohee Park, Taewon Yoon, and Heecheol Seo (Nov. 2021). "Ultra-High Dimensional Sparse Representations with Binarization for Efficient Text Retrieval." In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, pp. 1016–1029.

Jawahar, Ganesh, Benoît Sagot, and Djamé Seddah (July 2019). "What Does BERT Learn about the Structure of Language?" In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: ACL, pp. 3651–3657.

Jiang, Jyun-Yu and Wei Wang (2018). "RIN: Reformulation Inference Network for Context-Aware Query Suggestion." In: *Proceedings of*

the 27th ACM International Conference on Information and Knowledge Management. CIKM '18. New York, NY, USA: ACM, pp. 197–206.

Joachims, Thorsten, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay (2005). "Accurately Interpreting Clickthrough Data as Implicit Feedback." In: *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '05. Salvador, Brazil: Association for Computing Machinery, pp. 154–161.

Johnson, Jeff, Matthijs Douze, and Hervé Jégou (2019). "Billion-scale similarity search with GPUs." In: *IEEE Transactions on Big Data* 7.3, pp. 535–547.

Kai, Atsuhiko, Yoshifumi Hirose, and Seiichi Nakagawa (1998). "Dealing with out-of-vocabulary words and speech disfluencies in an n-gram based speech understanding system." In: *The 5th International Conference on Spoken Language Processing*. ISCA.

Karlsen, Randi and Anders Andersen (2019). "Recommendations with a Nudge." In: *Technologies* 7.2.

Khattab, Omar and Matei Zaharia (2020). "ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT." In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '20. Virtual Event, China: Association for Computing Machinery, pp. 39–48.

Kiesel, Johannes, Arefeh Bahrami, Benno Stein, Avishek Anand, and Matthias Hagen (2018). "Toward Voice Query Clarification." In: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. SIGIR '18. Ann Arbor, MI, USA: Association for Computing Machinery, pp. 1257–1260.

Kingma, Diederik P. and Jimmy Ba (2015). "Adam: A Method for Stochastic Optimization." In: *International Conference on Learning Representations*.

Kitaev, Nikita, Lukasz Kaiser, and Anselm Levskaya (2020). "Reformer: The Efficient Transformer." In: *International Conference on Learning Representations*.

Kolomiyets, Oleksandr and Marie-Francine Moens (2011). "A survey on question answering technology from an information retrieval perspective." In: *Information Sciences* 181.24, pp. 5412–5434.

Kotov, Alexander, Paul N. Bennett, Ryen W. White, Susan T. Dumais, and Jaime Teevan (2011). "Modeling and Analysis of Cross-Session Search Tasks." In: *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '11. Beijing, China: Association for Computing Machinery, pp. 5–14.

Kreyssig, Florian, Iñigo Casanueva, Paweł Budzianowski, and Milica Gašić (July 2018). "Neural User Simulation for Corpus-based Policy Optimisation of Spoken Dialogue Systems." In: *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*. Melbourne, Australia: Association for Computational Linguistics, pp. 60–69.

Kuenzer, Alexander, Christopher Schlick, Frank Ohmann, Ludger Schmidt, Holger Luczak, et al. (2001). "An empirical study of dynamic bayesian networks for user modeling." In: *Proc. of the UM'2001 Workshop on Machine Learning for User Modeling*. Citeseer, pp. 1–10.

Kumar, Manoj, Mohammad Babaeizadeh, Dumitru Erhan, Chelsea Finn, Sergey Levine, Laurent Dinh, and Durk Kingma (2020). "VideoFlow: A Conditional Flow-Based Model for Stochastic Video Generation." In: *International Conference on Learning Representations*.

Landthaler, Jörg, Bernhard Waltl, Dominik Huth, Daniel Braun, Christoph Stocker, Thomas Geiger, and Florian Matthes (2017). "Extending Thesauri Using Word Embeddings and the Intersection Method." In: *ASAIL@ ICAIL* 8.1, pp. 112–119.

Le, Quoc and Tomas Mikolov (2014). "Distributed Representations of Sentences and Documents." In: *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*. ICML'14. Beijing, China: JMLR.org, pp. II–1188–II–1196.

Lei, Wenqiang, Xiangnan He, Yisong Miao, Qingyun Wu, Richang Hong, Min-Yen Kan, and Tat-Seng Chua (2020). "Estimation-action-reflection: Towards deep interaction between conversational and recommender systems." In: *Proceedings of the 13th International Conference on Web Search and Data Mining*, pp. 304–312.

Levine, Nir, Haggai Roitman, and Doron Cohen (2017). "An Extended Relevance Model for Session Search." In: *The 40th International ACM SIGIR Conference on Research & Development in Information Retrieval*. SIGIR '17. New York, NY, USA: ACM, pp. 865–868.

Lewis, Mike, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer (July 2020). "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension." In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. ACL, pp. 7871–7880.

Li, Ruirui, Liangda Li, Xian Wu, Yunhong Zhou, and Wei Wang (2019). "Click Feedback-Aware Query Recommendation Using Adversarial Examples." In: *The World Wide Web Conference*. WWW '19. San Francisco, CA, USA: Association for Computing Machinery, pp. 2978–2984.

Li, Xiujun, Zachary C. Lipton, Bhuwan Dhingra, Lihong Li, Jianfeng Gao, and Yun-Nung Chen (2016). "A User Simulator for Task-Completion Dialogues." In: p. 14.

Li, Ziming, Julia Kiseleva, Alekh Agarwal, Maarten de Rijke, and Ryen W White (2020). "Optimizing Interactive Systems via Data-Driven Objectives." In: *arXiv preprint arXiv:2006.12999*.

Liu, Bing and Ian Lane (2017). "Iterative policy learning in end-to-end trainable task-oriented neural dialog models." In: *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 482–489.

Liu, Binsheng, Hamed Zamani, Xiaolu Lu, and J. Shane Culpepper (2021). "Generalizing Discriminative Retrieval Models Using Genera-

tive Tasks." In: *Proceedings of the Web Conference 2021*. WWW '21. Ljubljana, Slovenia: Association for Computing Machinery, pp. 3745–3756.

Liu, Jingjing and Nicholas J. Belkin (2010). "Personalizing Information Retrieval for Multi-Session Tasks: The Roles of Task Stage and Task Type." In: *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '10. Geneva, Switzerland: Association for Computing Machinery, pp. 26–33.

Liu, Mengyang, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma (2019a). "Investigating Cognitive Effects in Session-Level Search User Satisfaction." In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD '19. Anchorage, AK, USA: Association for Computing Machinery, pp. 923–931.

Liu, Qian, Yihong Chen, Bei Chen, Jian-Guang Lou, Zixuan Chen, Bin Zhou, and Dongmei Zhang (July 2020a). "You Impress Me: Dialogue Generation via Mutual Persona Perception." In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, pp. 1417–1427.

Liu, Xiaodong, Yelong Shen, Kevin Duh, and Jianfeng Gao (July 2018). "Stochastic Answer Networks for Machine Reading Comprehension." In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*. Vol. 1: Long Papers. Melbourne, Australia: ACL, pp. 1694–1704.

Liu, Xiaodong et al. (July 2020b). "The Microsoft Toolkit of Multi-Task Deep Neural Networks for Natural Language Understanding." In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. ACL, pp. 118–126.

Liu, Yinhan, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov (2019b). "RoBERTa: A Robustly Optimized BERT Pretraining Approach." In: arXiv: 1907.11692.

Liu, Zheng and Yingxia Shao (2022). "Retromae: Pre-training retrieval-oriented transformers via masked auto-encoder." In: *arXiv preprint arXiv:2205.12035*.

Lohmann, Steffen, J. Wolfgang Kaltz, and Jürgen Ziegler (2006). "Model-Driven Dynamic Generation of Context-Adaptive Web User Interfaces." In: *Proceedings of the 2006 International Conference on Models in Software Engineering*. MoDELS'06. Genoa, Italy: Springer-Verlag, pp. 116–125.

Luan, Yi, Jacob Eisenstein, Kristina Toutanova, and Michael Collins (2021). "Sparse, Dense, and Attentional Representations for Text Retrieval." In: *Transactions of the Association for Computational Linguistics* 9, pp. 329–345.

Luo, Jiyun, Xuchu Dong, and Hui Yang (Sept. 2015). "Session Search by Direct Policy Learning." In: *Proceedings of the 2015 International Conference on The Theory of Information Retrieval*. ICTIR '15. New York, NY, USA: Association for Computing Machinery, pp. 261–270.

Luo, Jiyun, Sicong Zhang, Xuchu Dong, and Hui Yang (2015). "Designing States, Actions, and Rewards for Using POMDP in Session Search." en. In: *Advances in Information Retrieval*. Ed. by Allan Hanbury, Gabriella Kazai, Andreas Rauber, and Norbert Fuhr. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 526–537.

MacAvaney, Sean, Andrew Yates, Arman Cohan, and Nazli Goharian (2019). "CEDR: Contextualized Embeddings for Document Ranking." In: *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR'19. Paris, France: Association for Computing Machinery, pp. 1101–1104.

Makhzani, Alireza and Brendan J Frey (2015). "Winner-Take-All Autoencoders." In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett. Vol. 28. Curran Associates, Inc.

Mallia, Antonio, Omar Khattab, Torsten Suel, and Nicola Tonellotto (2021). "Learning Passage Impacts for Inverted Indexes." In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '21. Virtual Event, Canada: Association for Computing Machinery, pp. 1723–1727.

Maxwell, David and Leif Azzopardi (2016a). "Agents, Simulated Users and Humans: An Analysis of Performance and Behaviour." In: *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. CIKM '16. Indianapolis, Indiana, USA: Association for Computing Machinery, pp. 731–740.

– (2016b). "Simulating Interactive Information Retrieval: SimIIR: A Framework for the Simulation of Interaction." In: *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '16. Pisa, Italy: Association for Computing Machinery, pp. 1141–1144.

Mehrotra, Rishabh and Emine Yilmaz (2017). "Extracting Hierarchies of Search Tasks & Subtasks via a Bayesian Nonparametric Approach." In: *The 40th International ACM SIGIR Conference on Research & Development in Information Retrieval*. SIGIR '17. New York, NY, USA: ACM, pp. 285–294.

Mei, Qiaozhu, Dengyong Zhou, and Kenneth Church (2008). "Query Suggestion Using Hitting Time." In: *Proceedings of the 17th ACM Conference on Information and Knowledge Management*. CIKM '08. New York, NY, USA: ACM, pp. 469–478.

Mikolov, Tomas, Wen-tau Yih, and Geoffrey Zweig (2013). "Linguistic Regularities in Continuous Space Word Representations." In: *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Atlanta, Georgia: Association for Computational Linguistics, pp. 746–751.

Mitra, Bhaskar and Nick Craswell (2015). "Query Auto-Completion for Rare Prefixes." In: *Proceedings of the 24th ACM International on*

Conference on Information and Knowledge Management. CIKM '15. New York, NY, USA: ACM, pp. 1755–1758.

Mnih, Volodymyr et al. (2015). "Human-level control through deep reinforcement learning." In: *Nature* 518.7540, pp. 529–533.

Muhammad, Aliyu (Nov. 2017). "Efficiency of Boolean Search strings for Information Retrieval." In: *American Journal of Engineering Research* 6, pp. 216–222.

Mustar, Agnès, Sylvain Lamprier, and Benjamin Piwowarski (July 2020). "Using BERT and BART for Query Suggestion." In: *Joint Conference of the Information Retrieval Communities in Europe*. Vol. 2621. CEUR Workshop Proceedings. Samatan, France: CEUR-WS.org.

– (2021). "On the study of transformers for query suggestion." In: *ACM Transactions on Information Systems (TOIS)* 40.1, pp. 1–27.

Mustar, Agnès, Sylvain Lamprier, and Benjamin Piwowarski (2022). "IRnator: A Framework for Discovering Users Needs from Sets of Suggestions." In: *Proceedings of the 2022 ACM SIGIR International Conference on Theory of Information Retrieval*. ICTIR '22. Madrid, Spain: Association for Computing Machinery, pp. 138–143.

Mysore, Sheshera, Tim O'Gorman, Andrew McCallum, and Hamed Zamani (2021). "CSFCube - A Test Collection of Computer Science Research Articles for Faceted Query by Example." In: *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Nakano, Reiichiro et al. (2021). "WebGPT: Browser-assisted question-answering with human feedback." In: *CoRR* abs/2112.09332. arXiv: 2112.09332.

Negi, Suraj, Shakhaf Joseph, Cydnelle Alemao, and Vincy Joseph (2020). "Intuitive User Interface for Enhanced Search Experience." In: *2020 3rd International Conference on Communication System, Computing and IT Applications (CSCITA)*, pp. 115–119.

Ng, Andrew Y., Daishi Harada, and Stuart J. Russell (1999). "Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping." In: *Proceedings of the Sixteenth International Conference on Machine Learning*. ICML '99. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., pp. 278–287.

Nguyen, Tri, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng (2016). "MS MARCO: A Human Generated MAchine Reading COmprehension Dataset." In: *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems*. Vol. 1773. CEUR Workshop Proceedings (NIPS' 2016).

Nogueira, Rodrigo Frassetto and Kyunghyun Cho (2019). "Passage Re-ranking with BERT." In: *CoRR* abs/1901.04085. arXiv: 1901.04085.

Nogueira, Rodrigo, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin (Nov. 2020). "Document Ranking with a Pretrained Sequence-to-Sequence

Model." In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics, pp. 708–718.

Nogueira, Rodrigo, Jimmy Lin, and AI Epistemic (2019). "From doc2query to docTTTTTquery." In: *Online preprint* 6.

Nogueira, Rodrigo, Wei Yang, Jimmy Lin, and Kyunghyun Cho (2019). "Document Expansion by Query Prediction." In: arXiv: 1904.08375.

Ozertem, Umut, Olivier Chapelle, Pinar Donmez, and Emre Velipasaoglu (2012a). "Learning to Suggest: A Machine Learning Framework for Ranking Query Suggestions." In: *The 35st International ACM SIGIR Conference on Research & Development in Information Retrieval*. New York, NY, USA: ACM, pp. 25–34.

– (2012b). "Learning to Suggest: A Machine Learning Framework for Ranking Query Suggestions." In: *The 35th International ACM SIGIR Conference on Research & Development in Information Retrieval*. SIGIR '12. New York, NY, USA: ACM, pp. 25–34.

Pallagani, Vishal and Biplav Srivastava (2021). "A Generic Dialog Agent for Information Retrieval Based on Automated Planning Within a Reinforcement Learning Platform." In: *Bridging the Gap Between AI Planning and Reinforcement Learning (PRL)*.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu (July 2002). "Bleu: a Method for Automatic Evaluation of Machine Translation." In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA: ACL, pp. 311–318.

Pasquale, Frank (2015). *The Black Box Society: The Secret Algorithms That Control Money and Information*. USA: Harvard University Press.

Paulus, Romain, Caiming Xiong, and Richard Socher (2017). "A deep reinforced model for abstractive summarization." In: *arXiv preprint arXiv:1705.04304*.

Phillips, Heather (Sept. 2010). "The Great Library of Alexandria?" In: *Library Philosophy and Practice* 2010.

Pogacar, Frances A., Amira Ghenai, Mark D. Smucker, and Charles L.A. Clarke (2017). "The Positive and Negative Influence of Search Results on People's Decisions about the Efficacy of Medical Treatments." In: *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*. ICTIR '17. Amsterdam, The Netherlands: Association for Computing Machinery, pp. 209–216.

Pradeep, Ronak, Rodrigo Nogueira, and Jimmy Lin (2021). "The Expando-Mono-Duo Design Pattern for Text Ranking with Pretrained Sequence-to-Sequence Models." In: *CoRR* abs/2101.05667. arXiv: 2101.05667.

Punera, Kunal and Srujana Merugu (2010). "The anatomy of a click: modeling user behavior on web information systems." In: *Proceedings of the 19th ACM international conference on Information and knowledge management*, pp. 989–998.

Purves, Ross and Christopher Jones (2011). "Geographic Information Retrieval." In: *SIGSPATIAL Special* 3.2, pp. 2–4.

Qiao, Yifan, Chenyan Xiong, Zhenghao Liu, and Zhiyuan Liu (2019). "Understanding the Behaviors of BERT in Ranking." In: arXiv: 1904.07531.

Qu, Chen, Chenyan Xiong, Yizhe Zhang, Corby Rosset, W. Bruce Croft, and Paul Bennett (2020). "Contextual Re-Ranking with Behavior Aware Transformers." In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '20. Virtual Event, China: Association for Computing Machinery, pp. 1589–1592.

Radford, Alec, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever (2019). "Language models are unsupervised multitask learners." In: *OpenAI blog*.

Radlinski, Filip and Nick Craswell (2017). "A Theoretical Framework for Conversational Search." In: *Proceedings of the 2017 Conference on Conference Human Information Interaction and Retrieval*. CHIIR '17. Oslo, Norway: Association for Computing Machinery, pp. 117–126.

Raffel, Colin, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu (2020). "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer." In: *Journal of Machine Learning Research* 21.140, pp. 1–67.

Ramsdell, Jordan and Laura Dietz (2020). "A Large Test Collection for Entity Aspect Linking." In: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pp. 3109–3116.

Reddy, Pritika, Bibhya Sharma, and Kaylash Chaudhary (2020). "Digital Literacy: A Review of Literature." In: *Int. J. Technoethics* 11.2, pp. 65–94.

Reimers, Nils and Iryna Gurevych (Nov. 2019). "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks." In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, pp. 3982–3992.

Rich, Elaine (1979). "User modeling via stereotypes." In: *Cognitive Science* 3.4, pp. 329–354.

Robertson, Stephen and K. Sparck Jones (1976). "Relevance weighting of search terms." In: *Journal of the American Society for Information Science* 27, pp. 129–146.

Robertson, Stephen, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford (1994). "Okapi at TREC-2." In: *The Second Text REtrieval Conference (TREC-2)*. Gaithersburg, MD: NIST, pp. 21–34.

Rocchio, Joseph (1971). "Relevance feedback in information retrieval." In: *The Smart retrieval system-experiments in automatic document processing*, pp. 313–323.

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams (1986). "Learning representations by back-propagating errors." In: *Nature* 323, pp. 533–536.

Ruotsalo, Tuukka, Giulio Jacucci, Petri Myllymäki, and Samuel Kaski (2014). "Interactive Intent Modeling: Information Discovery beyond Search." In: *Commun. ACM* 58.1, pp. 86–92.

Sadikov, Eldar, Jayant Madhavan, Lu Wang, and Alon Halevy (2010). "Clustering Query Refinements by User Intent." In: *Proceedings of the 19th International Conference on World Wide Web*. WWW '10. New York, NY, USA: ACM, pp. 841–850.

Salton, G. (1971). *The SMART Retrieval System—Experiments in Automatic Document Processing*. USA: Prentice-Hall, Inc.

Salton, G., A. Wong, and C. S. Yang (1975). "A Vector Space Model for Automatic Indexing." In: *Commun. ACM* 18.11, pp. 613–620.

Santos, Rodrygo L.T., Craig Macdonald, and Iadh Ounis (2011). "Intent-Aware Search Result Diversification." In: *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '11. Beijing, China: Association for Computing Machinery, pp. 595–604.

Schatzmann, Jost, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young (Apr. 2007). "Agenda-Based User Simulation for Bootstrapping a POMDP Dialogue System." In: *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*. Rochester, New York: Association for Computational Linguistics, pp. 149–152.

Schatzmann, Jost and Steve Young (2009). "The Hidden Agenda User Simulation Model." In: *IEEE Transactions on Audio, Speech, and Language Processing* 17.4, pp. 733–747.

Schulman, John, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov (2017). "Proximal policy optimization algorithms." In: *arXiv preprint arXiv:1707.06347*.

Schuster, M. and K.K. Paliwal (1997). "Bidirectional recurrent neural networks." In: *IEEE Transactions on Signal Processing* 45.11, pp. 2673–2681.

Scialom, Thomas, Sylvain Lamprier, Benjamin Piwowarski, and Jacopo Staiano (Nov. 2019a). "Answers Unite! Unsupervised Metrics for Reinforced Summarization Models." In: *2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. ACL Anthology. Hong Kong, China: Association for Computational Linguistics, pp. 3237–3247.

– (Nov. 2019b). "Answers Unite! Unsupervised Metrics for Reinforced Summarization Models." In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: ACM, pp. 3246–3256.

Sekaran, Kaushik, P Chandana, J Rethna Virgil Jeny, Maytham N Meqdad, and Seifedine Kadry (2020). "Design of optimal search engine using text summarization through artificial intelligence techniques." In: *TELKOMNIKA (Telecommunication Computing Electronics and Control)* 18.3, pp. 1268–1274.

Sennrich, Rico, Barry Haddow, and Alexandra Birch (Aug. 2016). "Neural Machine Translation of Rare Words with Subword Units." In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Vol. 1: Long Papers. Berlin, Germany: ACM, pp. 1715–1725.

Shah, Pararth, Dilek Hakkani-Tür, Gökhan Tür, Abhinav Rastogi, Ankur Bapna, Neha Nayak, and Larry P. Heck (2018). "Building a Conversational Agent Overnight with Dialogue Self-Play." In: *CoRR* abs/1801.04871. arXiv: 1801.04871.

Sherstinsky, Alex (2020). "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network." In: *Physica D: Nonlinear Phenomena* 404, p. 132306.

Singhal, Amit and Fernando Pereira (1999). "Document Expansion for Speech Retrieval." In: *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '99. Berkeley, California, USA: Association for Computing Machinery, pp. 34–41.

Sloan, Marc, Hui Yang, and Jun Wang (Apr. 2015). "A Term-Based Methodology for Query Reformulation Understanding." In: *Information Retrieval Journal* 18.2, pp. 145–165.

Song, Yang, Dengyong Zhou, and Li-wei He (2011). "Post-Ranking Query Suggestion by Diversifying Search Results." In: *The 34th International ACM SIGIR Conference on Research & Development in Information Retrieval*. SIGIR '11. New York, NY, USA: ACM, pp. 815–824.

Sordoni, Alessandro, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie (2015). "A Hierarchical Recurrent Encoder-Decoder for Generative Context-Aware Query Suggestion." In: *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. CIKM '15. New York, NY, USA: ACM, pp. 553–562.

Strohman, Trevor, Donald Metzler, Howard Turtle, and W Bruce Croft (2005). "Indri: A language model-based search engine for complex queries." In: *Proceedings of the international conference on intelligent analysis*. Vol. 2. 6. Citeseer, pp. 2–6.

Sussillo, David and Omri Barak (Mar. 2013). "Opening the Black Box: Low-Dimensional Dynamics in High-Dimensional Recurrent Neural Networks." In: *Neural Computation* 25.3, pp. 626–649. eprint: https://direct.mit.edu/neco/article-pdf/25/3/626/881886/neco_a_00409.pdf.

Tabarcea, Andrei, Najlah Gali, and Pasi Fränti (2017). "Framework for location-aware search engine." In: *Journal of Location Based Services* 11.1, pp. 50–74. eprint: https://doi.org/10.1080/17489725.2017.1407001.

Tan, Zhixing, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi (2018). "Deep semantic role labeling with self-attention." In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1.

Tangelder, J.W.H. and R.C. Veltkamp (2004). "A survey of content based 3D shape retrieval methods." In: *Proceedings Shape Modeling Applications, 2004.* Pp. 145–156.

Tay, Yi, Luu Anh Tuan, and Siu Cheung Hui (2018). "Hyperbolic representation learning for fast and efficient neural question answering." In: *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pp. 583–591.

Tedesco, Roberto, Peter Dolog, Wolfgang Nejdl, and Heidrun Allert (2006). "Distributed Bayesian networks for user modeling." In: *E-Learn: World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education*. Association for the Advancement of Computing in Education (AACE), pp. 292–299.

Thomas, Paul, Alistair Moffat, Peter Bailey, and Falk Scholer (Aug. 2014). "Modeling decision points in user search behavior." In: *Proceedings of the 5th Information Interaction in Context Symposium*. IIiX '14. New York, NY, USA: Association for Computing Machinery, pp. 239–242.

Trippas, Johanne R., Damiano Spina, Paul Thomas, Mark Sanderson, Hideo Joho, and Lawrence Cavedon (2020). "Towards a model for spoken conversational search." In: *Information Processing & Management* 57.2, p. 102162.

Tsai, F.S., Minoru Etoh, Xing Xie, Wang-Chien Lee, and Qiang Yang (2010). "Introduction to Mobile Information Retrieval." In: *IEEE Intelligent Systems* 25.1, pp. 11–15.

Tsukuda, Kosetsu, Tetsuya Sakai, Zhicheng Dou, and Katsumi Tanaka (2013). "Estimating Intent Types for Search Result Diversification." In: *Information Retrieval Technology*. Ed. by Rafael E. Banchs, Fabrizio Silvestri, Tie-Yan Liu, Min Zhang, Sheng Gao, and Jun Lang. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 25–37.

Turney, Peter D. and Patrick Pantel (2010). "From Frequency to Meaning: Vector Space Models of Semantics." In: *J. Artif. Int. Res.* 37.1, pp. 141–188.

Uc-Cetina, Victor, Nicolás Navarro-Guerrero, Anabel Martin-Gonzalez, Cornelius Weber, and Stefan Wermter (2022). "Survey on reinforcement learning for language processing." In: *Artificial Intelligence Review*, pp. 1–33.

Van Opijnen, Marc and Cristiana Santos (2017). "On the Concept of Relevance in Legal Information Retrieval." In: *Artif. Intell. Law* 25.1, pp. 65–87.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, undefinedukasz Kaiser, and Illia Polosukhin (2017). "Attention is All You Need." In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., pp. 6000–6010.

Wang, Hongning, Cheng Xiang Zhai, Feng Liang, Anlei Dong, and Yi Chang (2014). "User modeling in search logs via a nonparametric Bayesian approach." English (US). In: *WSDM 2014 - Proceedings of the 7th ACM International Conference on Web Search and Data Mining*. WSDM 2014 - Proceedings of the 7th ACM International Conference on Web Search and Data Mining. 7th ACM International Conference on Web Search and Data Mining, WSDM 2014 ; Conference date: 24-02-2014 Through 28-02-2014. Association for Computing Machinery, pp. 203–212.

Wang, Ju-Chiang, Yu-Chin Shih, Meng-Sung Wu, Hsin-Min Wang, and Shyh-Kang Jeng (2011). "Colorizing tags in tag cloud: a novel query-by-tag music search system." In: *Proceedings of the 19th ACM international conference on Multimedia*, pp. 293–302.

Wang, Sinong, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma (2020). "Linformer: Self-Attention with Linear Complexity." In: arXiv: 2006.04768.

Wenzlaff, Karsten and Sebastian Spaeth (2022). "Smarter than Humans? Validating how OpenAI's ChatGPT model explains Crowdfunding, Alternative Finance and Community Finance." In: *Validating how OpenAI's ChatGPT model explains Crowdfunding, Alternative Finance and Community Finance.*(*December 22, 2022*).

Wu, Bin, Chenyan Xiong, Maosong Sun, and Zhiyuan Liu (2018a). "Query Suggestion with Feedback Memory Network." In: *Proceedings of the 2018 World Wide Web Conference*. WWW '18. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, pp. 1563–1571.

Wu, Xianchao, Huang Hu, Momo Klyen, Kyohei Tomita, and Zhan Chen (2018b). "Q20: Rinna riddles your mind by asking 20 questions." In: *Japan NLP*.

Xie, Qing, Feng Xiong, Tian Han, Yongjian Liu, Lin Li, and Zhifeng Bao (2018). "Interactive resource recommendation algorithm based on tag information." In: *World Wide Web* 21.6, pp. 1655–1673.

Yan, Ming, Chenliang Li, Bin Bi, Wei Wang, and Songfang Huang (2021). "A Unified Pretraining Framework for Passage Ranking and Expansion." In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.5, pp. 4555–4563.

Yang, Hui, Dongyi Guan, and Sicong Zhang (2015). "The Query Change Model: Modeling Session Search as a Markov Decision Process." In: *ACM Trans. Inf. Syst.* 33.4.

Yang, Liu, Minghui Qiu, Chen Qu, Cen Chen, Jiafeng Guo, Yongfeng Zhang, W. Bruce Croft, and Haiqing Chen (2020). "IART: Intent-

Aware Response Ranking with Transformers in Information-Seeking Conversation Systems." In: *Proceedings of The Web Conference 2020*. WWW '20. Taipei, Taiwan: Association for Computing Machinery, pp. 2592–2598.

Yang, Min, Weiyi Huang, Wenting Tu, Qiang Qu, Ying Shen, and Kai Lei (2021a). "Multitask Learning and Reinforcement Learning for Personalized Dialog Generation: An Empirical Study." In: *IEEE Transactions on Neural Networks and Learning Systems* 32.1, pp. 49–62.

Yang, Wei, Haotian Zhang, and Jimmy Lin (2019). "Simple Applications of BERT for Ad Hoc Document Retrieval." In: arXiv: 1903.10972.

Yang, Yatao, Biyu Ma, Jun Tan, Hongbo Deng, Haikuan Huang, and Zibin Zheng (2021b). "FINN: Feedback Interactive Neural Network for Intent Recommendation." In: *Proceedings of the Web Conference 2021*. WWW '21. Ljubljana, Slovenia: Association for Computing Machinery, pp. 1949–1958.

Yu, Lili, Howard Chen, Sida Wang, Yoav Artzi, and Tao Lei (2019). "Interactive Classification by Asking Informative Questions." In: *arXiv preprint arXiv:1911.03598*.

Zamani, Hamed, Susan Dumais, Nick Craswell, Paul Bennett, and Gord Lueck (2020). "Generating Clarifying Questions for Information Retrieval." In: *Proceedings of The Web Conference 2020*. WWW '20. New York, NY, USA: ACM, pp. 418–428.

Zhan, Jingtao, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma (2020). "RepBERT: Contextualized text embeddings for first-stage retrieval." In: *arXiv preprint arXiv:2006.15498*.

Zhang, Ruiyi, Tong Yu, Yilin Shen, Hongxia Jin, Changyou Chen, and Lawrence Carin (2020). "Reward Constrained Interactive Recommendation with Natural Language Feedback." In: *arXiv preprint arXiv:2005.01618*.

Zhao, Xiangyu, Liang Zhang, Zhuoye Ding, Long Xia, Jiliang Tang, and Dawei Yin (2018). "Recommendations with negative feedback via pairwise deep reinforcement learning." In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1040–1048.

Zheng, Guoqing and Jamie Callan (2015). "Learning to Reweight Terms with Distributed Representations." In: *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '15. Santiago, Chile: Association for Computing Machinery, pp. 575–584.

Zhou, Ke, Ronan Cummins, Mounia Lalmas, and Joemon M. Jose (2013). "Which Vertical Search Engines Are Relevant?" In: *Proceedings of the 22nd International Conference on World Wide Web*. WWW '13. Rio de Janeiro, Brazil: Association for Computing Machinery, pp. 1557–1568.

Zhou, Xinyi and Reza Zafarani (2020). "A Survey of Fake News: Fundamental Theories, Detection Methods, and Opportunities." In: *ACM Comput. Surv.* 53.5.

Zhu, Yukun, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler (2015). "Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books." In: *Proceedings of the IEEE international conference on computer vision*, pp. 19–27.

Zou, Jie and Evangelos Kanoulas (2019). "Learning to Ask: Question-based Sequential Bayesian Product Search." In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 369–378.

Zuiderveen Borgesius, Frederik, Damian Trilling, Judith Möller, Balázs Bodó, Claes H De Vreese, and Natali Helberger (2016). "Should we worry about filter bubbles?" In: *Internet Policy Review. Journal on Internet Regulation* 5.1.