THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par l'Université Toulouse 3 - Paul Sabatier

Présentée et soutenue par

Luis Eduardo LUGO MARTINEZ

Le 15 décembre 2021

Modélisation des comportements de recherche basé sur les interactions des utilisateurs

Ecole doctorale : **EDMITT - Ecole Doctorale Mathématiques, Informatique et Télécommunications de Toulouse**

Spécialité : **Informatique et Télécommunications**

Unité de recherche :
IRIT : Institut de Recherche en Informatique de Toulouse

Thèse dirigée par
Gilles HUBERT

Jury
Catherine BERRUT, Professeure, Rapporteure
Guillaume CLEUZIOU, Professeur, Rapporteur
Gilles HUBERT, MCF HDR, Directeur de thèse
José G. MORENO, MCF, Co-encadrant

Acknowledgements

I would like to thank the Institut de Recherche en Informatique de Toulouse for the opportunity to be part of an amazing research team. Likewise, much gratitude to the Université Toulouse 3 Paul Sabatier and the École Doctorale MITT for providing a multicultural environment that stimulates scientific research and education.

I am very grateful to directeurs Gilles HUBERT and José G. MORENO for their guidance throughout a fascinating field of research. Their advice and feedback were fundamental to the successful completion of the project. Thank you very much as well to rapporteurs Pr Catherine BERRUT and Pr Guillaume CLEUZIOU.

Furthermore, much gratitude to the Agence Nationale de la Recherche (ANR) for supporting this research work through project CoST 2019 – 2022, Modélisation de Tâches Complexes en Recherche d'Information, code ANR-18-CE23-0016. By the same token, many thanks to all my friends and colleagues from the Institut de Recherche en Informatique de Toulouse. I sincerely appreciate your friendship, cordiality, and generosity.

Last but not least, I owe much appreciation to my family for their unconditional support, love, and encouragement.

Le résumé

Les utilisateurs de systèmes d'information divisent normalement les tâches en une séquence de plusieurs étapes pour les résoudre. En particulier, les utilisateurs divisent les tâches de recherche en séquences de requêtes, en interagissant avec les systèmes de recherche pour mener à bien le processus de recherche d'informations. Les interactions des utilisateurs sont enregistrées dans des journaux de requêtes, ce qui permet de développer des modèles pour apprendre automatiquement les comportements de recherche à partir des interactions des utilisateurs avec les systèmes de recherche. Ces modèles sont à la base de multiples applications d'assistance aux utilisateurs qui aident les systèmes de recherche à être plus interactifs, faciles à utiliser, et cohérents.

Par conséquent, nous proposons les contributions suivantes : un modèle neuronale pour apprendre à détecter les limites des tâches de recherche dans les journaux de requête ; une architecture de regroupement profond récurrent qui apprend simultanément les représentations de requête et regroupe les requêtes en tâches de recherche ; un modèle non supervisé et indépendant d'utilisateur pour l'identification des tâches de recherche prenant en charge les requêtes dans seize langues ; et un modèle de tâche de recherche multilingue, une approche non supervisée qui modélise simultanément l'intention de recherche de l'utilisateur et les tâches de recherche.

Les modèles proposés améliorent les méthodes existantes de modélisation, en tenant compte de la confidentialité des utilisateurs, des réponses en temps réel et de l'accessibilité linguistique. Le respect de la vie privée de l'utilisateur est une préoccupation majeure, tandis que des réponses rapides sont essentielles pour les systèmes de recherche qui interagissent avec les utilisateurs en temps réel, en particulier dans la recherche par conversation. Dans le même temps, l'accessibilité linguistique est essentielle pour aider les utilisateurs du monde entier, qui interagissent avec les systèmes de recherche dans de nombreuses langues. Les contributions proposées peuvent bénéficier à de nombreuses applications d'assistance aux utilisateurs, en aidant ces derniers à mieux résoudre leurs tâches de recherche lorsqu'ils accèdent aux systèmes de recherche pour répondre à leurs besoins d'information.

Abstract

Users of information systems normally divide tasks in a sequence of multiple steps to solve them. In particular, users divide search tasks into sequences of queries, interacting with search systems to carry out the information seeking process. User interactions are registered on search query logs, enabling the development of models to automatically learn search patterns from the users' interactions with search systems. These models underpin multiple user assisting applications that help search systems to be more interactive, user-friendly, and coherent. User assisting applications include query suggestion, the ranking of search results based on tasks, query reformulation analysis, e-commerce applications, retrieval of advertisement, query-term prediction, mapping of queries to search tasks, and so on.

Consequently, we propose the following contributions: a neural model for learning to detect search task boundaries in query logs; a recurrent deep clustering architecture that simultaneously learns query representations through self-training, and cluster queries into groups of search tasks; Multilingual Graph-Based Clustering, an unsupervised, user-agnostic model for search task identification supporting queries in sixteen languages; and Language-agnostic Search Task Model, an unsupervised approach that simultaneously models user search intent and search tasks.

Proposed models improve on existing methods for modeling user interactions, taking into account user privacy, realtime response times, and language accessibility. User privacy is a major concern in Ethics for intelligent systems, while fast responses are critical for search systems interacting with users in realtime, particularly in conversational search. At the same time, language accessibility is essential to assist users worldwide, who interact with search systems in many languages. The proposed contributions can benefit many user assisting applications, helping users to better solve their search tasks when accessing search systems to fulfill their information needs.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

On ne voit bien qu'avec le cœur.
L'essentiel est invisible pour les yeux.

Antoine de Saint-Exupéry

Users divide mental tasks in a sequence of multiple easy steps to deal with mental overload (Kahneman, 2013). In particular, when users interact with search systems to satisfy their information needs, they divide search tasks into sequences of queries. Those interactions are registered in query logs, and mining users' interactions enable the development of models that learn the patterns users follow to carry out their search tasks (Hearst, 2009; Lucchese et al., 2013; Mehrotra and Yilmaz, 2017).

Initially, when users run a search query, search systems produced a list of hyperlinks for ranked documents. Commonly known as the "ten-blue-links" approach to search, this paradigm is no longer in use (Rosset et al., 2020). Nowadays, search systems assist users while they carry out their search tasks using multiple supporting applications, including query autocompletion (Li et al., 2017; Sun and Lou, 2014), search clarification (Zamani et al., 2020), query suggestion (Ahmad et al., 2019; Tamine et al., 2020), query term prediction, e-commerce applications, product recommendations, advertising retrieval for products (Hearst, 2009; Mehrotra and Yilmaz, 2017) query-task mapping (Völske et al., 2019), ranking of results based on search tasks (Ahmad et al., 2019), suggestion of questions (Rosset et al., 2020), and so on.

Furthermore, advances in speech recognition, natural language understanding, and text-to-speech synthesis enable users to leverage digital personal assistants to satisfy their information needs (Thomas et al., 2020; Zamani and Craswell, 2020). Digital assistants are now available in desktops, smartphones, tablets, smartwatches, and dedicated smart speakers. When users access digital assistants for information seeking, tracking the conversation topic is essential so that users can carry out their search tasks (Khatri et al., 2018; Venkatesh et al., 2018).

To underpin those user assisting applications and retrieval methods, it is crucial to develop models that automatically learn from user interactions with search systems. As search systems evolve and new platforms emerge, centering on users to help them in their information seeking journeys, learning from user interactions is becoming increasingly important in information retrieval research (Mehrotra et al., 2020).

Therefore, the research problem we aim to address in this dissertation is the development of models that learn search patterns from user interactions with search systems. For developing these models, along with the interactions of users with search systems, we also leverage advances in natural language processing for encoding search queries. As queries are steps users execute as part of a search task, and because of the relevance of search tasks in many user assisting applications and retrieval methods, we focus our research on search patterns related to user search tasks.

To address the research problem, the first approach is the detection of search task boundaries in user interactions with search systems. Some existing models use surrounding user queries or clicked URLs, which make them unfeasible to use in realtime interactions. Other models use heuristics with constant thresholds, but a manual statistical analysis is required to update the thresholds. We propose a neural model that learns to detect search task boundaries in query logs. The interactions of the users in the search query logs are ordered by time and grouped by user identifiers, providing a chronologically ordered set of interactions per user. Such interactions are the input to the proposed model, which is supervised (Lugo et al., 2020b).

The second approach deals with the modeling of search tasks from user interactions with search systems. We propose Recurrent Deep Clustering (RDC), a deep clustering architecture that extends the foregoing neural model to extract user search tasks from query logs in an unsupervised way. The deep clustering architecture uses a multiobjective approach, simultaneously optimizing the query representation and the clustering of queries into search tasks. The representation for user queries is learned through a self-supervised method, leveraging data augmentation for language inputs and dual-channel neural architectures (Lugo et al., 2021a,b).

Most existing models for search task identification are monolingual, supporting user queries normally in English only. However, users around the world access search systems in many languages. Also, a single user can issue queries in two or more languages to solve a particular information need. Another limitation of existing models for search tasks is the lack of realtime response times, which relegates them to offline modeling applications. For the third approach, we propose Multilingual Graph-based Clustering (MGBC), a multilingual unsupervised approach for search task identification, supporting user queries in several languages and providing the possibility of mapping queries to search tasks in realtime (Lugo et al., 2020a).

Finally, the fourth approach is Language-agnostic Search Task Modeling (LASTM), an unsupervised model that uses a multiobjective approach, simultaneously optimizing search task clusters and user search intents through the relationship between user queries and clicked documents. The modeling scheme for user search intent leverages user query – clicked document relationships, representing queries in a language-agnostic semantic space. This proposed model takes advantage of query-task mapping to provide results in realtime (Lugo et al., 2021c). Moreover, the code for all contributions is publicly available in a GitHub repository[1], facilitating the replication of results and the utilization of proposed models for further research.

1.1 Overview

The remaining chapters of this dissertation are organized as follows. Chapter 2 presents a critical review of existing work, ranging from publicly available datasets to machine learning methods, to natural language models for text representation, to existing information retrieval literature centered around user search patterns. Chapter 3 proposes a neural model for learning to detect search task boundaries in query logs. In chapter 4, a recurrent deep clustering architecture for search task extraction is proposed, along with a self-supervised method to learn query representations using data augmentation. Chapter 5 develops a multilingual, unsupervised approach for identifying user search tasks, with the ability to map queries to search tasks in realtime. Chapter 6 proposes a model for user search intent, which leverages the relationship between user queries and clicked documents. At the same time, it develops a language-agnostic, unsupervised approach for modeling search tasks using semantic relationships between queries. Finally, chapter 7 explores the significance of the contributions, along with perspectives for future research directions.

[1] https://github.com/lelugom/search

Chapter 2
Background

...sin saber que la búsqueda de las cosas perdidas está entorpecida por los hábitos rutinarios, y es por eso que cuesta tanto trabajo encontrarlas.

Gabriel García Márquez

More than half the world's population uses the internet, according to the Internet World Stats[1]. Essential components in such interactions are search systems, which help users access the ever-increasing amount of information available to satisfy their information needs. Search systems allow users to perform their search tasks, relying on large web indexes to respond to users' requests. For example, Google's search index has an estimated size of around 55 billion web pages, while Microsoft's Bing search index has an estimated size of around 12 billion web pages[2].

Likewise, the most popular mobile applications, like WhatsApp Messenger, Twitter, and Facebook[3], have search systems available for the millions of users who access these applications every day. Android, the mobile OS with the biggest user base, has a default search bar on its main screen. By the same token, Windows, the most popular desktop OS in the world[4], has a search box right beside the Start button. The search box is also a default feature in its Windows 10 version. On top of that, the advent of smart home devices

[1] https://www.internetworldstats.com/stats.htm

[2] https://www.worldwidewebsize.com

[3] https://www.cnet.com/news/10-most-downloaded-apps-of-the-decade-facebook-dominated-2010-2019

[4] https://https://arstechnica.com/gadgets/2021/02/the-worlds-second-most-popular-desktop-operating-system-isnt-macos-anymore

and digital assistants like Amazon Alexa, Apple Siri, Google Assistant, and Microsoft Cortana, constitutes another way to interact with search systems.

All those interactions enable the development of models that learn search patterns to better assist users in their information seeking. In this chapter, we perform a critical review of existing literature, comparing and contrasting existing research related to the modeling of users' search patterns.

To structure the literature review, we consider that a wide range of needs and desires from users are converted to queries and run in search systems (Hearst, 2009). Search query logs register the queries users run to fulfill their information needs. Some of those search query logs are publicly available for analysis and model extraction (Section 2.1). Mining such query logs allows the modeling of search behaviors through automatic or manual analysis of user interactions. For automatic analysis, machine learning (Section 2.2) provides methods for processing the high dimensional data present in query logs. Machine learning approaches require representing queries in a vector space. Thus, multiple language representation models (Section 2.3) have been proposed to encode the text data in user queries. Using query representations and machine learning, we can model the tasks that users perform on search systems (Section 2.4), as well as other search patterns, including the segmentation of query logs (Section 2.5), that enable the development of several user assisting applications and retrieval models to support users while they perform their information seeking.

2.1 Search query log datasets

Multiple research works (Thomas et al., 2020; Wang et al., 2013a,b; Zhang et al., 2019) use proprietary search query logs mainly for two reasons: user privacy and commercial concerns that could arise when other search systems exploit the information (Craswell et al., 2020a). However, there are some publicly available datasets that we can leverage to model search behaviors (Table 2.1).

SogouQ[5] released the SogouQ User Query Log (SUQL) (Liu et al., 2011). The Lite version from June 2008 has 1,724,264 queries corresponding to around one day of search activities. Queries were registered from 519,876 user identifiers and are mostly in Chinese (Table 2.2). A subset of the SUQL collection with search tasks labels (SUQLST) contains 18,600 queries from 532 users (Du et al., 2018); however, SUQLST is not publicly available for download.

AOL[6] released the AOL Query Log (AOLQL) in 2006 (Pass et al., 2006). It contains 36,389,566 queries from 657,426 users, collected over a period of

[5] `http://www.sogou.com`

[6] `https://www.aol.com`

around three months (Gayo-Avello, 2009). Multiple subsets from the AOLQL collection has been released, with ground-truth labels for topical sessions (Gayo-Avello, 2009; Gomes et al., 2019) and search tasks (Hagen et al., 2013; Lucchese et al., 2013; Sen et al., 2018; Völske et al., 2019).

The AOLQL subset with topical sessions (AOLTS) (Gayo-Avello, 2009; Gomes et al., 2019) contains 10235 queries issued by 215 unique users. Labels for the queries in the dataset correspond to topical sessions, which are groups of successive queries related to the same information need (Gayo-Avello, 2009). Topical sessions are independent of information needs. For instance, the user with identifier 1713103 issued the query "ebay.com", which has the topical session '733'. The same user issued the query "ebay" later on, after issuing queries with other information needs like "club pogo" or "metal detecting". Nevertheless, the topical session label for the query "ebay" changed to '737' even though it reflects the same user intent as the query "ebay.com".

Dataset	**Reference**	**No. of queries**	**Labels**
SUQL	(Liu et al., 2011)	1724264	None
SUQLST	(Du et al., 2018)	18600	Search tasks
AOLQL	(Pass et al., 2006)	36389566	None
AOLTS	(Gayo-Avello, 2009)	10235	Topical sessions
AOLQTM	(Völske et al., 2019)	41780	Search tasks
WSMC12	(Hagen et al., 2013)	8840	Search tasks
CSTE	(Sen et al., 2018)	1424	Search tasks
TGSST	(Lucchese et al., 2013)	1424	Search tasks
TRECQTM	(Völske et al., 2019)	47514	Search tasks
ORCAS	(Craswell et al., 2020a)	10400000	Document IDs
WHQTM	(Völske et al., 2019)	119292	Search tasks
CUSTA	(Dosso et al., 2020)	2390	Search tasks

Table 2.1: Search query collections for modelling several users' search patterns.

The Webis Search Mission Corpus 2012 (WSMC12) dataset (Hagen et al., 2013) has 8840 entries with 2881 search task labels of 127 users. It is a subset of the AOLQL collection. Labels in this dataset correspond to search tasks – known as search missions – for a particular user identifier, grouping queries by user. If two users run a query for the same search task, the search task label is not necessarily the same. For instance, the user with identifier 9887420 issued the query "maps" and it was labeled with search task '2'; the user with identifier 1713103 issued the query “maps” as well, but it was labeled with search task ’22’. Overall, both queries represent the same search task because user intents reflect maps websites, as the clicked URLs show: the clicked URL for the first case was maps.google.com while the clicked URL for the second case was maps.yahoo.com.

The Cross-Session Task Extraction (CSTE) dataset (Sen et al., 2018) is a subset of the AOLQL collection. The CSTE dataset has 1424 entries with 224 labels corresponding to cross-session search tasks, without grouping queries by user information or query timestamps. Because of this, ground-truth labels are independent from users and time sessions. By contrast, the Time Gap Session with Search Tasks (TGSST) dataset (Lucchese et al., 2013) has the same queries than the CSTE dataset, but the 1424 entries are grouped by user identifiers and time sessions. Time sessions are clusters of chronological queries with a time span of less than 26 minutes between subsequent query pairs. Every time session has its own search task labels. If queries with the same tasks happens in different time sessions, they could have distinct search task labels. For instance, the query ”precious momunts”, run by user with identifier 117514, has a search task label of ’3’ for time session 3, while the query ”precious momounts”, run by the same user, has a search task label of ’1’ for time session 5.

Combining search queries and user clicks, the Open Resource for Click Analysis in Search (ORCAS) (Craswell et al., 2020a) contains 18.8M clicked document query pairs for 10.4M unique queries. Entries in ORCAS comprise query ID, query text, document ID, and document URL. The document IDs and URLs come from the TREC deep learning track document collection (Craswell et al., 2020b); by doing so, it is possible to avoid revealing the ranking mechanism of the search system. Queries are anonymized to protect user information. Queries are also processed to remove sensitive information like adult content or offensive words.

The Webis Query-Task-Mapping Corpus 2019 has three datasets with search task labels (Völske et al., 2019), where commercial search engines[7][8] provide suggested queries to perform data augmentation. The AOL-based Query-Task-Mapping (AOLQTM) dataset has 41780 queries and labels for 1423 search tasks. The TREC-based Query-Task-Mapping (TRECQTM) dataset has 47514 queries with labels for 276 search tasks. And the WikiHow-based Query-Task-Mapping (WHQTM) dataset has 119292 queries with labels for 7202 search tasks.

The Complex User Search Task Analysis (CUSTA) dataset (Dosso et al., 2020) comes from a study of user interactions with search systems when solving complex search tasks. The study included search tasks in computer science, psychology, and medicine, with questionaries to assess if users had experience in the field, as experienced users in a particular field already have

[7] `https://www.google.com`
[8] `https://www.bing.com`

several abstractions in that field. They also can use analogies to solve problems (Mitchell, 2019; Oakley, 2014), making it easier to solve complex search tasks. Five types of learning tasks were evaluated: simple tasks, learning tasks, decision tasks, problem-solving tasks, and multicriteria tasks. Queries in the dataset are mostly in French (Table 2.2), totaling 2390 entries for 32 users.

Dataset	**Query Language**	**Query Timestamps**	**Clicked URLs**	**Ground-truth labels**
SUQL	Chinese	√	√	X
AOLTS	English	√	X	√
AOLQTM	English	X	X	√
WSMC12	English	√	√	√
CSTE	English	X	X	√
TGSST	English	X	X	√
TRECQTM	English	X	X	√
ORCAS	English	X	√	X
WHQTM	English	X	X	√
CUSTA	French	X	X	√

Table 2.2: Common features in search log datasets. Only publicly available datasets are considered.

AOLQTM, TRECQTM, and WHQTM are benchmark datasets for testing methods that map user queries to search tasks. Table 2.3 include reference results for the three datasets, considering Word Movers Distance (WMD), MinHash Locality-Sensitive Hashing (MinHash LSH), the Trie data structure[9], which is the fastest mapping method, and ElasticSearch BM25[10], which is the most accurate one (Völske et al., 2019).

Though AOLQTM, TRECQTM, and WHQTM have ground-truth labels for search tasks, they do not have query timestamps. Thus, for models of search behaviors requiring the computation of time spans between queries, WSMC12 offers both query timestamps and ground-truth labels for search tasks (Table 2.2). Query timestamps in WSMC12 are also essential when it is required to determine whether two queries are adjacent. Finally, as search task labels in WSMC12 are not user-independent, models for search tasks can

[9] https://github.com/google/pygtrie
[10] https://www.elastic.co

leverage the CSTE dataset, which has user-independent ground-truth labels. CSTE contains, however, user queries in English only, so the CUSTA dataset can complement CSTE when testing search task models in multilingual setups.

Dataset	**Method**	**Accuracy**	**Query time**
AOLQTM	Trie	0.69	0.46ms
	MinHash LSH	0.66	2.42ms
	WMD	0.67	7.16s
	ElasticSearch BM25	0.78	2.80ms
TRECQTM	Trie	0.66	0.51ms
	MinHash LSH	0.68	2.50ms
	WMD	0.73	9.24s
	ElasticSearch BM25	0.80	2.95ms
WHQTM	Trie	0.48	0.33ms
	MinHash LSH	0.41	2.28ms
	WMD	0.55	22.65s
	ElasticSearch BM25	0.63	4.21ms

Table 2.3: Reference results for mapping queries to search tasks.

2.2 Machine learning

Given the large size of search query logs and the high dimensionality of their information, automatic analysis methods are crucial to model user behaviors from query logs. Automatic methods can extract models from query logs in order to make descriptions or predictions of future user needs and interactions. As machine learning provides methods to process large and high dimensional data in an efficient manner (Murphy, 2012), machine learning enables the modeling of user behaviors from their interactions with search systems.

Existing query logs with ground-truth labels provide an input – output relationship that the machine learning method can discover through the learning process (Alpaydin, 2014). This is known as supervised learning, where a set of inputs is mapped to a certain output, and the parameters of the system are changed to improve the matching of the model. Nonetheless, the need for input-output samples represents a challenge for supervised learning models.

They usually need big datasets that are cleaned and labeled by humans. In most datasets, there is a long tail problem: common categories have high probabilities, providing large counts of samples. But there are a lot of categories with few or no samples. Also known as edge cases, they are unlikely but possible. These cases are challenging for machine learning models and can easily confuse them. To deal with such scenarios, a possibility is to train the models with common cases and let them learn the rest with unsupervised learning (Mitchell, 2019).

In unsupervised learning, there is a set of inputs but no expected output, thus, the model should extract patterns from the inputs to create valid parameters. As unsupervised learning approaches do not need labels, this is a viable alternative. Examples of unsupervised learning include models that group similar items using abstract similarities or learn new categories by analogy to existing ones (Murphy, 2012; Mitchell, 2019). Another type of machine learning is reinforcement learning, which is based on a system of punishments or rewards for the outputs generated, which tailor the system for appropriate modeling (Murphy, 2012).

Furthermore, machine learning models can be parametric or non-parametric. Parametric models have a fixed number of parameters in spite of the input dataset size. They are computationally faster but less flexible. Parametric models could be relatively simple like logistic regression or more complex like deep neural networks. Conversely, In non-parametric models, the number of parameters depends on the dataset. They are more flexible; however, large datasets can make them computationally intractable. Also, non parametric models could be difficult to scale because they grow as the dataset grows. An example of a non-parametric machine learning method is k-nearest neighbor (KNN). The label of one data point depends on the class with the higher number of neighbors. K defines the complexity of the model. A low k has a complex decision surface while a high k has a smoother decision surface, usually generating as output the class with the higher data density (Murphy, 2012).

2.2.1 Supervised models

In supervised learning, the classical error rate in pattern classification – the classification error rate – enables the computation of a loss function for the models. The error rate is the relationship between correct input-target outputs and the testing set size (Graves, 2012):

$$E = \frac{1}{|S'|} \sum_{(x,z)\in S'} \begin{cases} 0 \text{ if } h(x) = z \\ 1 \text{ otherwise} \end{cases} \tag{2.1}$$

where S' is the test set, x is the input, z the expected output, and h is the learning algorithm.

Traditional supervised machine learning algorithms depend heavily on the way input data is represented. Each attribute of the proposed representation is known as a feature. It usually requires considerable time and effort to find the correct way to represent input data in order to get a good performance in the machine learning method. The process also needs high domain specific knowledge to extract the most important attributes of the input dataset. But automatically learned representations usually have a better performance than handcrafted feature extraction. When methods learn to represent input data by themselves, we are talking about representation learning. Deep learning models are a kind of representation learning. They learn not only the mapping between features and results, but the representation of input data itself (Goodfellow et al., 2016).

Therefore, deep neural networks provide a way to infer models from the raw data by extracting hidden structures from the information. They also automatically extract the set of attributes that best represents the input dataset – improving the learning process by encoding representations as a nested hierarchy of simpler or less abstract representations (Goodfellow et al., 2016; Rampasek and Goldenberg, 2016). Very similar deep neural network architectures have worked for various case studies in not related domains, which proves the suitability of deep learning models to adapt to various problem domains. At the same time, it proves the capacity of deep neural models to automatically extract features from data in spite of dataset domain origin. It also shows the ability of deep neural networks to learn non–linear transformations in their hidden structures and to generate distributed representations of input datasets (Goodfellow et al., 2016; Angermueller et al., 2016).

Common deep learning architectures include Multilayer Perceptrons, Convolutional Neural Networks, and Recurrent Neural Networks (Jouppi et al., 2017).

Multilayer perceptrons (MLPs) represent models of artificial neural networks that are useful for classification and regression tasks. They are also known as feedforward networks. The perceptron is the basic processing unit of the MLP. It comprises a series of weights that transform the input nodes through a mathematical operation. Then, an output node computes the result by combining such transformations into a single value. If a system uses only one perceptron, it can only approximate linear functions of input data. Thus, it only performs linear regression. But real applications usually require nonlinear regressions. MLPs implement hidden layers of perceptrons between input and output nodes. Those hidden layers enable the approximation of nonlinear functions for the input data. An MLP with a single hidden layer and enough nonlinear nodes – processing units – can approximate any continuous function in a compact domain. Therefore, they are considered universal function approximators. (Alpaydin, 2014; Graves, 2012).

Both hidden layers and output nodes in MLPs have nonlinear activation functions. Once one or more input nodes are activated, the information propagates through the whole set of interconnected nodes in the subsequent layers, up to the output layer. Such propagation is known as a forward pass. The MLP response does not depend on the past or future inputs, it only depends on the current input. Because of this, they are a good alternative for pattern classification tasks (Graves, 2012).

Convolutional neural networks (CNNs) are models inspired by biological visual systems. Specifically, they are inspired by the visual cortex of the brain, where a combination of simple and complex neurons interact to build the powerful natural visual system. CNNs comprise convolution layers, nonlinear layers, and pooling layers. These models are quite successful nowadays for their ability to process spatial and multidimensional information (Min et al., 2016). Most of the record-breaking applications of these neural networks are part of the machine vision area, including semantic segmentation, object recognition, image classification, and image retrieval use CNNs (Angermueller et al., 2016; Dolz et al., 2016).

Current advances in parallel computing, optimization techniques, and network architectures have enabled recurrent neural network (RNN) applications in large-scale deep learning problems (Lipton et al., 2015). They are inspired by the cyclical connection of the neurons inside our brain. Neural networks with cyclical connections also include recursive and feedback networks (Graves, 2012). Novel applications for recurrent networks include unsupervised video encoding, video captioning, biological sequence analysis, and program execution. Also, they are widely used in the natural language processing landscape (Graves, 2012; Lipton et al., 2015; Mitchell, 2019).

2.2.1.1 Recurrent neural networks

An RNN is a type of neural network ideally suited to process sequential information. Sequences appear in multiple domains, including queries in a search log, frames in a video, or words in a sentence. In an RNN, the output for a step in the input sequence depends on the input of that step and information of the preceding steps (Rajaraman and Ullman, 2011). These networks are inspired by the cyclical connection of the neurons inside our brain. They store information from input sequences by using iterative function loops (Graves, 2012).

RNNs process the output in the forward direction, starting from the first step in the sequence, they compute forward hidden states until the last step is reached. However, steps ahead of the current step in the input sequence can provide valuable information to the overall results. Bidirectional RNNs (BiRNNs) were proposed to process the input sequence simultaneously in a forward and backward direction. To do so, they compute forward hidden states and backward hidden states. Concatenating forward and backward hidden states enables the generation of outputs that leverage information from preceding and following steps in the sequence (Bahdanau et al., 2015).

To store preceding information, RNNs have a hidden state vector, which works as a memory while the network processes the sequence (Rajaraman and Ullman, 2011). In practical setups, the memory in the standard RNN only works effectively with information very close to the step that the network is processing. Thus, Long Short-Term Memory (LSTM) networks were proposed to enhance the performance of standard RNNs with long term information. LSTMs have the ability to save important information, forget information that is not relevant, and focus on the parts of the sequence that better serve the overall network performance. To implement the ability to focus, a two-tier configuration comprises a hidden state vector, which works as the working memory, and a cell state vector, which corks as the long term memory. A popular variant of the LSTM network is the Gated Recurrent Unit (GRU), a much simpler architecture to compute and implement. It relies on a single vector for memory purposes, decreasing the number of parameters needed (Cho et al., 2014; Rajaraman and Ullman, 2011).

2.2.1.1.1 Equations for RNNs

Formally, given an input sequence $x_1, x_2, \dots x_T$, equations for a standard RNN are defined as follows (Martens and Sutskever, 2011):

$$\begin{aligned} t_i &= W_{hx}x_i + W_{hh}h_{i-1} + b_h \\ h_i &= e(t_i) \\ s_i &= W_{yh}h_i + b_y \\ y_i &= g(s_i) \end{aligned} \tag{2.2}$$

where W_{hx}, W_{hh}, W_{yh} are learnable weight matrices, b_h, b_y are biases, h_i are the hidden states, y_i are the outputs, e and g represent activation functions. Common activation functions for RNNs are the hyperbolic tangent (Equation 2.3) and the logistic sigmoid (Equation 2.4). Both functions are vector valued functions which are differentiable and non-linear. A differentiable activation function allows the use of gradient descent for neural network training. On the other hand, non-linearity makes neural networks more powerful than their linear equivalents (Graves, 2012; Martens and Sutskever, 2011).

$$tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{2.3}$$

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.4}$$

The following equations represent an LSTM unit (Graves, 2013):

$$\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i)\\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f)\\
c_t &= f_t c_{t-1} + i_t tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c)\\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o)\\
h_t &= o_t tanh(c_t)
\end{aligned} \tag{2.5}$$

where i is the input gate, f is the forget gate, o is the output gate, and c is the unit cell. b are biases and W are learnable weight matrices. The hidden state of the LSTM is the concatenation of h and c.

GRUs are similar to LSTM units. However, they are simpler (Jozefowicz et al., 2015; Cho et al., 2014):

$$\begin{aligned}
r_t &= \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r)\\
z_t &= \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z)\\
\tilde{h}_t &= tanh(W_{xh}x_t + W_{hh}(r_t h_{t-1}) + b_h)\\
h_t &= z_t h_{t-1} + (1 - z_t)\tilde{h}_t
\end{aligned} \tag{2.6}$$

where r_t is the reset gate, z_t is the update gate, b are biases and W are learnable weight matrices.

For the network outputs in classification systems, a softmax function (Equation 2.7) provides normalized output activations that represent the class probabilities (Graves, 2012).

$$y_j = \frac{e^{y_j}}{\sum_{k=1}^{K} e^{y_j}} \text{ where } j = 1, 2, \ldots, K \tag{2.7}$$

True class probabilities are obtained by representing the true labels with a 1-of-K coding scheme (Cho et al., 2014), a binary vector with one-hot encoding. The cross entropy loss function (Equation 2.8) provides the target function that we minimize in order to train the network. By doing so, we minimize the classification error rate in Equation 2.1.

$$\mathcal{L} = -\sum_{k=1}^{K} z_k ln(y_k) \tag{2.8}$$

where z represents the true class probabilities and y represents the network output probabilities.

2.2.1.1.2 The attention mechanism

To improve the performance of RNN architectures, the attention mechanism (Luong et al., 2015; Bahdanau et al., 2015) processes the output of the recurrent layers to focus in the most relevant parts of the sequences. The attention mechanism has been applied in search task identification, neural machine translation, language models, and several other language tasks (Du et al., 2018; Vaswani et al., 2017). It creates a context vector $\mathbf{c}_t$ from a weighted combination of intermediate output states (Equation 2.12) from the recurrent layer. In the global attention (Luong et al., 2015) with general content-based score (Equation 2.10), the weights are stored in an alignment vector $\mathbf{a}_t$:

$$\mathbf{a}_t = \frac{\exp(score(\mathbf{h}_t, \tilde{\mathbf{h}}_s))}{\sum_{s'} \exp(score(\mathbf{h}_t, \tilde{\mathbf{h}}_{s'}))} \tag{2.9}$$

$$score(\mathbf{h}_t, \tilde{\mathbf{h}}_s) = \mathbf{h}_t^T \mathbf{W}_a \tilde{\mathbf{h}}_s \tag{2.10}$$

where $\tilde{\mathbf{h}}_s$ are the intermediate output states, $\mathbf{h}_t$ is the last output state and $\mathbf{W}_a$ is a learnable weight matrix. The concatenation of the context vector and the final output state into a dense layer generates the output of the attention mechanism:

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_c [\mathbf{c}_t, \mathbf{h}_t]) \tag{2.11}$$

$$\mathbf{c}_t = \sum_{s'} \mathbf{a}_t \tilde{\mathbf{h}}_{s'} \tag{2.12}$$

The attention mechanism takes as input the recurrent layer outputs. However, it is possible to replace the recurrent layer altogether, using a stack of self-attention layers that process input sequences directly. This stack of self-attention layers is commonly known as a transformer (Vaswani et al., 2017). As the transformer processes the input directly, without the iterations required by recurrent layers, it is faster to train using massive collections of text. Nonetheless, the longer the input sequence, the bigger the transformer becomes, requiring large computational resources for training and inference. Indeed, transformers have a complexity of $\mathcal{O}(n^2)$ in time and space (Murphy, 2021). If resources are constrained, it is necessary to trim the input or use small batch sizes for training. Another alternative is to use long-range transformer architectures, which are specifically designed to process long sequences (Tay et al., 2020). Nowadays, transformers are widely used in natural language models and computer vision (Devlin et al., 2019; Feng et al., 2020; Khan et al., 2021; Radford et al., 2018; Yang et al., 2020).

2.2.2 Unsupervised models

The need for large datasets that are cleaned and labelled by humans is challenging (Mitchell, 2019; Du et al., 2018; Wang et al., 2013a). As mentioned before, a possibility is to train the models with common cases and let them learn the rest with unsupervised learning. As unsupervised learning approaches do not need labels, this is a viable alternative. In an unsupervised approach, there is a density estimation for the input data. No output is provided, thus, the method should discover useful patterns from the available input (Murphy, 2012; Mitchell, 2019).

Examples of unsupervised learning include dimensionality reduction and clustering models that group similar items using abstract similarities or learn new categories by analogy to existing ones (Mitchell, 2019). Dimensionality reduction aims to extract the most important latent information in the data to better describe the original data with a lower number of dimensions. Examples include Principal Component Analysis (PCA) and Latent Semantic Analysis (LSA), which uses global matrix factorization for document retrieval (Dumais, 2004; Murphy, 2012; Pennington et al., 2014).

Clustering is a canonical example of unsupervised learning. Here, the cluster is not part of the input data, thus, it is a latent or hidden variable. Two goals comprise a clustering task: first, we need to find the number of clusters which better fits the existing data. Secondly, we want to know which cluster each data point pertains to (Murphy, 2012). Clustering methods are essential in multiple data-driven applications. They are primarily based on partitioning, density, and hierarchies (Aljalbout et al., 2018; Min et al., 2018). An example is Latent Dirichlet Allocation (LDA) (Blei et al., 2003), which is a probabilistic generative model which has been successful in clustering documents into topics by using the co-occurrence of common words in them (Li et al., 2017).

Another type of clustering uses graphs, representing each sample in the input dataset as a node and using similarities between samples as weights in the edges of the graphs (Chen and Ji, 2010; Nascimento and De Carvalho, 2011; Nie et al., 2016; Wang et al., 2019b). This approach is commonly used for processing query log datasets (Du et al., 2018; Lucchese et al., 2011, 2013; Sen et al., 2018).

Along with dimensionality reduction and clustering, there is another unsupervised method specifically designed to train neural network architectures. Though it is possible to pretrain neural architectures using labeled datasets, there is an unsupervised alternative that trains neural architectures without the need for ground-truth labels. Commonly known as self-training, this unsupervised alternative allows neural architectures to optimize for a certain objective that can be automatically calculated without manual labels (Karamanolakis et al., 2021). For example, the masked language modeling (MLM) (Devlin et al., 2019) objective allows the training of large language architectures using massive text datasets, without the need for any ground-truth label, which would be challenging to generate for language datasets with millions of samples.

A recent self-training configuration uses a dual-channel architecture (Chen et al., 2020) and data augmentation techniques to train models in an unsupervised way (Figure 2.1), using only 1% of existing labels. The configuration performs contrastive learning (Yang et al., 2019), maximizing the agreement

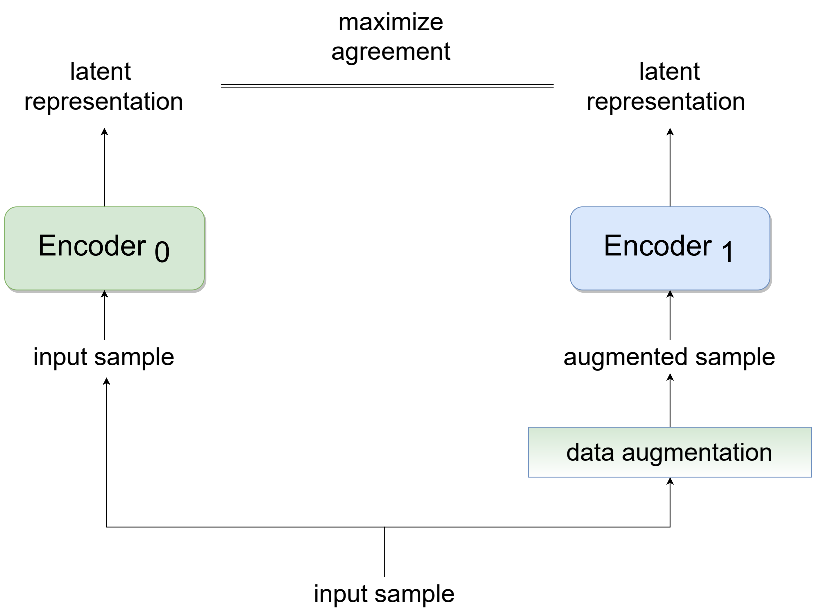


Fig. 2.1: Dual encoder configuration (Chen et al., 2020; Yang et al., 2019) for self-training neural network architectures using data augmentation techniques on input samples.

between the output of the encoders situated at each one of the channels. Known as the Simple framework for contrastive learning (SimCLR) of visual representations, it uses a predefined neural architecture for image classification - ResNet (He et al., 2016), a set of stochastic data augmentations for image samples, a projection head, and a contrastive learning loss to better learn latent representations of image samples.

The whole SimCLR framework uses two identical branches, modifying the input sample to create two altered versions of the same image. The contrastive crossentropy loss is designed to minimize the distance between the pair of altered versions, comparing the altered version to the remaining altered versions of the minibatch. There is only one positive pair of augmented images in the minibatch. For helping the architecture to avoid learning trivial features and generalize better, the series of data augmentations is applied to a single channel of the dual-channel model. The projection head on top of the ResNet output is a crucial component of the architecture, helping the model to preserve more information in the latent space, which improves the overall accuracy results.

Data augmentations in SimCLR are specifically designed for image datasets. They include stochastic croppings and resizing, color transformations, rotations, and other transformations like Gaussian noise, Gaussian blur, Sobel filter, and so on. Results show it is crucial to combine several image transformations to increase the performance of the architecture. Applying single image transformations yields lower metrics than their application in tandem. Also, bigger and deeper neural architectures have a higher impact in SimCLR than when training the architecture with a supervised learning approach. The rate of improvement is higher as the dual-channel architecture grows, while the rate of improvement for the ResNet architecture alone is slower. Moreover, larger batch sizes and longer training times represent a benefit for SimCLR. This phenomenon happens because the larger batch size and longer training times expose the model to a bigger count of negative samples.

2.2.3 Model evaluation

There are several metrics to assess the performance of machine learning models, depending on the objective we are evaluating. Accuracy defines the relationship between the correct outputs and the total number of outputs (Equation 2.13). However, accuracy by itself could be a poor estimator of model performance in some cases. A more robust approach includes the calculation of precision and recall. Precision defines the fraction of detections reported by the model that is correct. On the other hand, recall defines the fraction of true events in the testing set that were successfully detected by the model (Goodfellow et al., 2016):

$$Accuracy = \frac{T_P + T_N}{T_P + F_P + T_N + F_N} \tag{2.13}$$

$$Precision = \frac{T_P}{T_P + F_P} \tag{2.14}$$

$$Recall = \frac{T_P}{T_P + F_N} \tag{2.15}$$

where T_P are true positives, F_P are false positives, T_N are true negatives, and F_N are false negatives. Table 2.4 has a tabular representation of T_P, F_P, T_N, and F_N for a supervised binary case (Tan et al., 2018).

	Predicted Class 0	Predicted Class 1
Actual Class 0	T_P	F_P
Actual Class 1	F_N	T_N

Table 2.4: Confusion matrix for a binary classification model

A common summarization of model performance combines precision p and recall r into the F_β score (Du et al., 2018):

$$\mathrm{F}_\beta = \frac{(1+\beta)^2 pr}{\beta^2 p + r} \tag{2.16}$$

where β is the factor that controls the weight of precision and recall. When $\beta = 1.0$, we have a balanced F-score (Goodfellow et al., 2016). But in certain cases, we would like to give more weight to precision or recall in the F-score calculation, thus, when $\beta < 1.0$, the precision has more weight in the F_β score, while a $\beta > 1.0$ gives more weight to recall in the F_β score.

In the case of unsupervised learning models, when ground-truth labels are available, we compute the accuracy by maximizing the match between the labels predicted by the model and the ground-truth labels (Min et al., 2018):

$$ACC = \max_m \frac{\sum_{i=1}^{n} \mathbf{1}\left\{y_i = m(c_i)\right\}}{n} \tag{2.17}$$

where y_i is the ground-truth label, c_i is the label predicted by the model, and m is the matching function that ranges over all possible mappings between ground-truth and generated labels. For maximizing the matching function, the Hungarian algorithm enables the computation of the optimum mapping between predicted labels and ground-truth labels (Min et al., 2018; Xie et al., 2016).

As far as the F_β score is concerned, we also calculate it with equation 2.16, but we use a pairwise approach to compute T_P, F_P, T_N, and F_N. What matters in the pairwise approach is the relationship between the predicted labels and the ground-truth labels of a given pair of samples. For a given pair of input samples, if the predicted labels pertain to the same group and the ground-truth labels also pertain to the same group, then we have a T_P. If the predicted labels do not pertain to the same group and the ground-truth labels do not pertain to the same group either, then we have a T_N. If the predicted labels pertain to the same group and the ground-truth labels do not pertain to the same group, then we have a F_P. Finally, if the predicted labels do not pertain to the same group and the ground-truth labels do pertain to the same group, then we have a F_N (Mehrotra and Yilmaz, 2017).

Other metrics for unsupervised learning models include the Normalized Mutual Information (NMI) and the Adjusted Rand Index (ARI) (Min et al., 2018). Considering U as the ground-truth labels and V as the predicted labels, NMI is calculated as follows (Vinh et al., 2010):

$$NMI(U,V) = \frac{2I(U,V)}{H(U) + H(V)} \tag{2.18}$$

where $I(U,V)$ represents the mutual information and H is the entropy.

While NMI is an information theoretic based metric, ARI is a paired counting based measure. ARI is calculated as follows (Vinh et al., 2010):

$$ARI(U,V) = \frac{2(N_{00}N_{11} - N_{01}N_{10})}{(N_{00}+N_{01})(N_{01}+N_{11}) + (N_{00}+N_{10})(N_{10}+N_{11})} \quad (2.19)$$

where N_{00} are the number of pairs that are in different clusters in both U and V, N_{01} are the number of pairs that are in the same cluster in U but in different clusters in V, N_{10} the number of pairs that are in different clusters in U but in the same cluster in V, and N_{11} the number of pairs that are in the same cluster in both U and V.

2.3 Language representation

When processing search query logs with machine learning models, the language data in user queries needs to be represented in a vector space. But language is hard for computers to understand because it heavily depends on context, it requires a large amount of background knowledge common to the parts communicating for proper understanding, and it has an inherent ambiguity (Mitchell, 2019). First attempts to encode language data used one-hot encodings for a large vocabulary. However, one-hot representations can not encode semantic relationships between words (Mitchell, 2019). Similarly, Bag-of-words (Zhang et al., 2010) text representations could pose problems when encoding texts that are lexically similar but have different semantic content. They can also miss texts with different words that have the same semantic meaning (Zhang et al., 2019).

Therefore, distributed representations were proposed to improve text encodings from a semantic standpoint. Distributed representations can learn to map semantically related words very close in the word embeddings space by leveraging the fact that semantically similar words appear in similar surroundings (Sen et al., 2018; Zhang et al., 2019). According to the distributional hypothesis, the meaning of a word is related to the context in which it appears. In the field of linguistics, you can know a word by the context accompanying it. More precisely, the linguist John Firth said "You shall know a word by the company it keeps". This is formally known as distributed semantics: the semantic similarity of two linguistic expressions is related to the similarity of the contexts in which they appear (Mitchell, 2019). Distributed semantics also affect the quality of word vectors. The quality of word vectors depends on the quantity and quality of text datasets used to learn the word vector models (Grave et al., 2018; Bojanowski et al., 2017).

Furthermore, a word can have different meanings depending on its context; thus, if we use a vector to represent it, in certain dimensions of that vector the word is close to words with a similar meaning, while in other dimensions, it should be closer to other words that share another meaning (Mitchell, 2019). For that reason, word vectors capture the multiclustering concept from distributed representations (Bengio, 2009; Pennington et al., 2014).

2.3.1 Word vectors

Based on the distributed semantics concept from linguistics and a multidimensional representation of words using vectors, word2vec (Mikolov et al., 2013) uses a one-hot encoding for the input and output vocabulary, along with an intermediate feedforward layer with 300 units to learn a word vector for each input in the dictionary. The training method for the model scans millions of texts to produce pairs of words that appear in documents. The scanning ignores words like prepositions to get more meaningful pairs. It also considers the pair of words both in the original order in which they appear in the text and the inverse order. The feedforward layer learns to predict the next word of the pair. Once it is trained, the neural layer produces a 300 dimensional word vector for each input word in the vocabulary. The word2vec multidimensional representation captures semantic relatedness, as points in space representing words with similar meanings appear nearby (Mitchell, 2019).

Word2vec (Mikolov et al., 2013) has two types of training objectives: Skip Gram and Continuous Bag-of-Words (CBOW). Both objectives rely on surrounding context to model the word embeddings space. Skip Gram predicts a word's context from the word itself, while CBOW predicts the word from its context. To define the context, the training objectives consider a local window of words. From that context, a matrix is learned to minimize the hinge loss between related words and non-related words (Sen et al., 2018; Zhang et al., 2019). Both Skip Gram and CBOW produce quantitatively and qualitatively similar word vectors (Mikolov et al., 2013; Nalisnick et al., 2016)

NLP applications using Word2vec representations normally discard the output layer, using only the input vectors to compute semantic representations of text (Mitchell, 2019; Nalisnick et al., 2016). However, output vectors have important semantic relations with words that appear the texts to be transformed because the input-output dataset for training word embeddings depends on the context of the input words (Nalisnick et al., 2016). Output vectors can be useful in certain scenarios. For instance, Dual Embedding Space Model (DESM) (Nalisnick et al., 2016) uses input word embeddings to represent the query words and output word embeddings to represent the documents in the index, using the cosine similarity to compute a relevance metric for the indexed documents. Documents are represented by the centroid vector of the averaged vectors that encode the sentences, which can be precomputed to decrease latency. The relevance depends on the cosine similarity of the word embeddings in the query and the centroids encoding the documents. The best performing embeddings are trained using a large collection of queries from Bing, using the CBOW model. The DESM model outperforms the BM25 index baseline as well as Latent Semantic Analysis (LSA) results.

The use of local context windows in word vectors like Word2Vec has one major limitation: it overlooks the global co-occurrence statistics from the training corpus. This is a major concern because of the amount of repetition that text datasets tend to contain. Because of this limitation, GloVe (Pennington et al., 2014) word vectors combine the local window context of Word2Vec with the global statistics present in matrix factorization methods like LSA. The combination proves effective, allowing GloVe vectors to improve on word vectors based on local context windows. Improvements are observed in word analogy tasks, similarity tasks, and named entity recognition.

As machine learning architectures usually require inputs in the form of vectors, word vectors like Word2Vec and GloVe – also known as word embeddings – are ideal for approaches that rely on machine learning models. Word vectors encode syntactic and semantic information in a continuous vector space, facilitating multiple language tasks (Grave et al., 2018; Mikolov et al., 2013). Several variations of word vector representations have been proposed and they are now widely used in various successful NLP applications (Mitchell, 2019). Furthermore, pre-trained word vectors are now available in multiple languages (Grave et al., 2018). However, an initial phase for language selection (Kiela et al., 2018) or the use of automatic integration of models (Conneau et al., 2017; Lample et al., 2017) are necessary.

Moreover, when the input for the machine learning model is a sentence, we might want to compute a single vector representation per sentence. In this case, individual word vectors could be averaged, a representation approach used in several recent studies (Gomes et al., 2019; Mehrotra and Yilmaz, 2017; Sen et al., 2018; Saini et al., 2019; Yang et al., 2020; Zamani and Croft, 2016a,b). Another alternative is the use of convolutional neural networks to produce one vector per sentence (Kim, 2014).

2.3.2 Custom word vectors

In spite of the way word vectors map semantically and syntactically related words, they still pose problems when capturing fine grained details from sentences. In addition, they can suffer from topic shifting (Rekabsaz et al., 2017; Zhang et al., 2019).

To address such limitations in distributed word representations, it is possible to train the word vectors using search-related signals like relevance feedback (Zamani and Croft, 2017; Diaz et al., 2016; Zhang et al., 2019). Using top retrieved results from user queries, a recent work (Zamani and Croft, 2017) proposed a query embedding that relies on the relationship between terms that appear both in the query and the resulting documents.

Another possibility is to fine tune word vectors for specific search related tasks (Diaz et al., 2016; Sen et al., 2018). The tempo-lexical word embedding approach (Sen et al., 2018) relies on an embedding mechanism that incorporates both temporal and lexical information. This word embedding uses the task context to find query representations in a semantic space oriented to search tasks. One vector represents each query in the semantic space. The tempo-lexical word embedding is used in a global approach to identify users tasks in search logs. It takes into account the whole query log to train the word embedding, improving the subsequent results in the clustering of cross-session queries related to the same task.

Word embeddings could work for long documents but they might not reflect the context in queries, where short sentences or keywords are utilized by users to look for information. Thus, the number of words can be low and the window for context can not include the amount of information it has in long documents. In a temporal context, a 26-minute window (Lucchese et al., 2013) provides the context to train the word vectors. In the tempo lexical case, additional clustering of queries according to their relatedness helps to improve the space for learning the representations (Sen et al., 2018).

2.3.3 Universal language models

Often, there is not enough data to custom train end-to-end word vector models. Also, some NLP problems are not supervised. Thus, models focused on creating universal representations were proposed (Zhang et al., 2019). Such models include Bidirectional Encoder Representations for Transformers (BERT) (Devlin et al., 2019), Embeddings from Language Models (ELMo) (Peters et al., 2018), and Generative Pretraining (GPT) (Radford et al., 2018), now in its third version (Brown et al., 2020).

Some universal language models like GPT process input language sequences in only one direction (Radford et al., 2018). Instead, ELMo uses two separate recurrent models for both directions and then join the results (Peters et al., 2018). It uses hidden states from right-to-left (RTL) and left-to-right (LTR) sections of the recurrent neural network. The hidden states are then concatenated for subsequent steps, providing a bidirectional approach for processing the input. BERT also takes a bidirectional approach, taking advantage of context on both backward and forward directions, but in contrast to ELMo, it uses a transformer-based architecture. The BERT architecture is based on a bidirectional transformer and its results prove it outperforms existing alternatives in various natural language processing tasks. Results also highlight the importance of creating an architecture, pre-training it, and then performing transfer learning by fine-tuning it to perform different natural language tasks (Devlin et al., 2019).

Indeed, model pre-training for tasks based on NLP has proven instrumental in a number of applications, including natural language inference, paraphrasing, named entity recognition, and question answering. There are two trends. In the first trend, pretraining is performed in order to extract features from language. Feature extraction architectures focus on generating representations for other supervised tasks in the processing pipeline. Word vectors (Bojanowski et al., 2017; Grave et al., 2018; Sen et al., 2018) take advantage of this approach. There are word embeddings approaches with broader scopes, extracting features from sentences or paragraphs. Another scope is context. Models like ELMo (Peters et al., 2018) and tempo-lexical word vectors (Sen et al., 2018) leverage context to generate word vectors from text data (Devlin et al., 2019)

The second trend performs transfer learning, fine-tuning models for specific tasks. Transfer learning approaches leverage fine-tuning to perform NLP Tasks. This approach starts by training supervised models with a certain objective, usually with abundant labeled data available, and then perform transfer learning and fine-tune the model for the target goal. Although unsupervised pretraining for models leverages huge amounts of unlabeled data, supervised pretraining has also proven useful in various fields, including natural language tasks and computer vision applications (Devlin et al., 2019).

Together with pretraining, BERT uses several components as input to the model. The input can include three components (Devlin et al., 2019):

1. Embeddings from tokens, where a CLS token marks the beginning of a sentence, and a SEP token marks the separation for cases when the pretraining requires two sentences

2. Embeddings for sentences, A for one sentence, A - B for two sentences

3. Positional embeddings for learning the position of the token with respect to the whole sentence

Few additions are stacked on top of the BERT architecture for fine tuning specific tasks. Tasks include both one or two sentences for inference. For sentence labeling, the architecture takes as input the output tokens and learns to classify them. For sentence classification, the fine tuning is performed over the output of the CLS token. The same happens for pairs of sentences, where the CLS token output lets the fine tuning system learn sentiments, relatedness, etc. Another alternative is to use BERT output as features for further processing. In this case, although there is a loss in metrics with respect to the fine-tuned model, the best working configurations concatenate the last four layers outputs (Devlin et al., 2019).

Results show that BERT outperforms similar architectures in all tasks considered for evaluation, including the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2019a). Unsupervised pretraining has a big impact on BERT final results. The bidirectional approach to process the input texts has a significant impact in the final results as well. Furthermore, bigger language models create incremental improvements when there are larger datasets. BERT helps problems with small datasets to also get improvements thanks to model pretraining. Similarly, as language models are less sensitive to hyperparameters when there are large training datasets, BERT parameters create a larger impact on final language model results when there is a small dataset in a fine-tuning configuration (Devlin et al., 2019).

2.3.4 Language models for sentences

Universal Sentence Encoder (USE) (Cer et al., 2018) learns universal representations using multitask learning on language semantic tasks (Zhang et al., 2019). To produce sentence embeddings, USE can have two configurations. The first configuration is more accurate but more complex computationally. It is based on the encoding part of the transformer architecture. Context-aware word embeddings are generated by the encoder; then, they are averaged together to produce one vector per sentence. As mentioned before (Section 2.2.1), the computational complexity is $\mathcal{O}(n^2)$ in sentence length, making it more computationally expensive as the length of the sentence grows. The second configuration averages together word embeddings and bi-gram embeddings. The resulting average vector feeds a Deep Average Network (DAN), a feedforward network which computes a fixed vector representation for the input sentence. It is less computationally expensive than the transformer-based model. Its computational complexity is linear with respect to sentence length. However, the transformer-based model is more accurate.

The training of the universal sentence encoder uses a multitask approach to improve model performance. A supervised training phase with the Stanford Natural Language Interface (SNLI) augments the unsupervised training, which relies on sources like Wikipedia, web news, discussion forums, etc. To test the model, several natural language processing tasks are utilized in a transfer learning setup, proving the ability of the architecture even when small labeled datasets are available. For transfer learning experiments, a task-specific neural network takes as input the sentence embeddings. Furthermore, the semantic textual similarity between pairs of sentences uses the arccos to compute the angle between the embeddings. The angular distance between two sentence embeddings has been found to work better for pairwise semantic similarity tasks than the cosine similarity (Cer et al., 2018).

Most universal language models (Cer et al., 2018; Devlin et al., 2019; Peters et al., 2018; Radford et al., 2018) use contextual information to learn universal representations. But it is not clear how those generic representations relate to user intent (Zhang et al., 2019). Instead, Generic Intent Representation (GEN) creates a distributed space to represent queries modeling the user intent – what the user wants to get from the query. It learns from user clicks in retrieved documents, a weak supervision signal strongly related to the user intent. Training the model with query paraphrasing and question paraphrasing datasets help to improve the generalization abilities of the model (Zhang et al., 2019). Applications of the GEN (Zhang et al., 2019) encoder include the generation of more relevant results, better query suggestions, and more precise answers. It also helps to deal with the long-tail effect in search engines and model query reformulation behaviors.

The GEN (Zhang et al., 2019) architecture comprises three branches: word embeddings, character embeddings, and a mix encoder. The word embedding maps words to a continuous vector space. The character embedding deals with rare words. The mix encoder composes word embeddings into query encodings. It uses a BiGRU to learn cases in which the ordering of words has an impact on the query intent. Both the word and character embedding parts of the architecture include a highway neural layer to improve model capacity. The character embedding uses a CNN architecture and max-pooling before the highway layer.

The learning is performed end-to-end. It has two phases. In the first phase, it utilizes user co-clicks to learn user intent. The loss for query co-clicks is designed to minimize the distance between related query pairs while maximizing it between negative query pairs. Co-click queries are sampled from a large log. To avoid trivial negatives, the model uses noise-contrastive estimation (NCE) to get good adversarial samples. During the second training phase, the model uses a multitask approach, which combines the query co-clicks weak supervision loss along with a query paraphrasing loss and a question paraphrasing loss to improve the generalization ability of the model (Zhang et al., 2019).

For evaluation purposes, an intrinsic user intent strategy is based on three datasets with increasing levels of difficulty: general, easy, and hard. The normalized discounted cumulative gain (NDCG) is the metric for query intent evaluation. The area under the curve (AUC) provides performance metrics for paraphrasing tasks. GEN consistently outperforms USE (Cer et al., 2018) sentence representation, the best performing encoding of all the baselines considered for experiments. In turn, USE gets better results than the best performing information retrieval based representation: the Relevance Language Model (RLM) (Zamani and Croft, 2017).

Moreover, an ablation study of the GEN encoder is conducted to evaluate the multiple components of the proposed model. The most important conclusion of the ablation study is the importance of user clicks in training the model. Simple models trained with user click get results very close to state of the art text encodings, while complex methods failed to outperform Term Frequency – Inverse Document Frequency (TF-IDF) (Robertson, 2004). Thus, the supervision signal is more important than the architecture used to learn the model (Zhang et al., 2019).

2.3.5 Multilingual language models

Multilingual Universal Sentence Encoder (MUSE) (Yang et al., 2020) extends previous work (Cer et al., 2018) to provide sentence embeddings suited to information retrieval tasks in a multilingual semantic space. It supports 16 languages and comprises three configurations: a CNN based model, a transformer based model, and a question answering model. All configurations use SentencePiece (Kudo and Richardson, 2018), a language independent subword tokenizer to process the input sequence. SentencePiece covers above 99% of possible tokens in all languages. The sentence encoder uses a dual encoder architecture. The first encoder generates sentence embeddings. It is shared in all the retrieval tasks, where additional layers process the sentence embeddings to produce the results.

The transformer based model relies on the encoder part of the transformer architecture. It takes into account context-aware embeddings and averages together the results from the encoder to produce one vector per sentence. The CNN based module processes the vectors for tokens using convolutional and average pooling layers, similar to the approach in previous work (Kim, 2014). On the output of the convolutional network, additional dense layers process the average pooling layer results to generate the sentence embedding. The transformer based model is more computationally expensive, with a complexity of $\mathcal{O}(n^2)$ on sentence length (Section 2.2). CNN has a complexity linear with respect to sentence length, but it is less accurate than the transformer-based model (Yang et al., 2020).

Training is based on question answer pairs, translation pairs, and Stanford Natural Language Inference (SNLI). Google translate is used for the SNLI dataset – which only has english data – and for translating question answer pairs to balance the data in all languages considered. Downstream tasks include sentence retrieval, bitext retrieval, and retrieval question answering. Sentence retrieval returns all the sentences semantically related to the query sentence. Bitext retrieval deals with bilingual sentence pairs. And retrieval question answering process question answer tuples, considering the question as the query and retrieving the answer corresponding to that question in the tuple (Yang et al., 2020).

Language-agnostic BERT Sentence Embedding (LABSE) (Feng et al., 2020) provides the sentence embeddings to represent texts in a language-agnostic latent space. Using a 12-layer transformer architecture (Devlin et al., 2019; Vaswani et al., 2017) in a dual configuration, LABSE takes the transformer's hidden state for the last token in the sentence to generate the query representation.

The training process of LABSE comprises two phases: a pre-training phase and a fine-tuning phase, with a total of 109 languages. The pretraining uses masked language modeling (MLM) (Devlin et al., 2019) and translation language modeling (TLM) (Conneau and Lample, 2019) tasks to train the 12-layer transformer architecture. MLM masks words in the monolingual input phrases and trains the model to predict the masked word, using 17 billion monolingual phrases. TLM concatenates a phrase in one language with its translation, masking words and training the model to predict the masked words with 6 billion bilingual phrases. After pre-training with MLM and TLM, the LABSE architecture is fine-tuned using the translation ranking task. In translation ranking, the model learns to select the phrase in the target language, representing the best translation for the phrase in the source language (Feng et al., 2020).

In cross-lingual text retrieval, using the TATOEBA corpus (Artetxe and Schwenk, 2019), LABSE outperforms MUSE in the 14 languages that the latter supports. It also outperforms the baseline Language-agnostic Sentence Representations (LASER) (Artetxe and Schwenk, 2019) in the 82 languages that LASER was trained with. Additionally, it can support languages not included in the training dataset, achieving an average of 83.7% in the 112 languages of the TATOEBA dataset, even though there are more than 30 languages that are not part of the training dataset for LABSE. By contrast, LASER obtains 65.5% in the 112 languages.

Overall, LABSE and MUSE provide representations for user queries in a multilingual vector space. Each query is represented with a single vector and the language coverage varies according to the language model. for representing queries in English only, GloVe leverages both local context and global statistic in language corpus. GloVe vectors can be averaged together to generate a single vector per user query, or used individually for machine learning models requiring one vector per word.

2.4 Search task identification

Leveraging language models (Section 2.3) and machine learning approaches (Section 2.2), we can identifiy information seeking patterns in search query logs (Section 2.1). The standard model for the information seeking process establishes certain steps users take when they need to look for information. Such steps include 1) recognizing the need for information, 2) formulating the query according to the information they need, the context, and the time, 3) running the query in a search system to retrieve results, and 4) analyzing the retrieved information and reformulating the query if the results are not satisfactory (Hearst, 2009).

To support users during the different steps of the information seeking process, it is crucial to correctly model users' queries according to their search tasks. Information needs could include complex search tasks, where users may need to multitask, shift goals, or iterate to refine queries. Moreover, search history improves personalization of search engine results and provides data to extract predictive models for supporting users while they execute their search tasks (Hearst, 2009). In particular, search system logs provide fine-grained

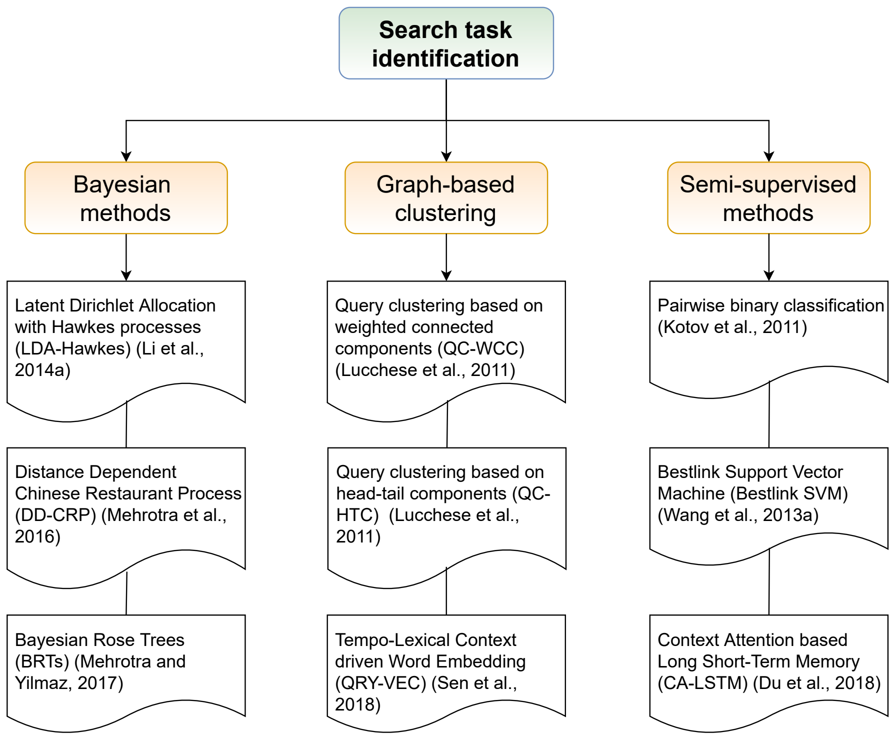

Search task identification
Bayesian methods
Graph-based clustering
Semi-supervised methods
Latent Dirichlet Allocation with Hawkes processes (LDA-Hawkes) (Li et al., 2014a)
Distance Dependent Chinese Restaurant Process (DD-CRP) (Mehrotra et al., 2016)
Bayesian Rose Trees (BRTs) (Mehrotra and Yilmaz, 2017)
Query clustering based on weighted connected components (QC-WCC) (Lucchese et al., 2011)
Query clustering based on head-tail components (QC-HTC) (Lucchese et al., 2011)
Tempo-Lexical Context driven Word Embedding (QRY-VEC) (Sen et al., 2018)
Pairwise binary classification (Kotov et al., 2011)
Bestlink Support Vector Machine (Bestlink SVM) (Wang et al., 2013a)
Context Attention based Long Short-Term Memory (CA-LSTM) (Du et al., 2018)

Fig. 2.2: Recent machine learning models for search task identification, including Bayesian methods, semi-supervised methods, and graph-based clustering approaches.

details at the query level, enabling the characterization and classification of individual queries according to their tasks. A precise task identification improves user supporting applications, helping users to complete their information seeking tasks. Supporting the user while performing complex tasks is essential to avoid the cognitive load on the user. For that support, it is crucial that search systems identify the tasks related to the issued query. For instance, query term prediction requires the search system to correctly identify the task the user is realizing to properly predict the next likely steps (Mehrotra et al., 2016; Mehrotra and Yilmaz, 2017).

Similarly, the correct clustering of queries by search tasks enables the design of proactive search systems that can suggest next queries after the analysis of user intents. Suggestions help to support the user in the information seeking process, as the user does not have to issue additional related queries (Sen et al., 2018). Other applications like query recommendation (Boldi et al., 2008; Zhao et al., 2018), conversational search (Zamani and Craswell, 2020), user modeling based on tasks (Hearst, 2009), query - task mapping (Völske et al., 2019), personalization in e-commerce web interfaces (Mehrotra, 2015), advertisement retrieval, ranking of results according to the task, and prediction of user satisfaction based on tasks (Mehrotra and Yilmaz, 2017), and so on, rely on accurate identification of tasks from user interactions.

Search task identification can be performed with methods using heuristics-based models (Hagen et al., 2013). A model based on a cascade of heuristics (Hagen et al., 2013) first segments the search query log in logical sessions; then, it performs a post-processing step to detect search tasks – known as search missions – based on the queries pertaining to the logical sessions. However, several manually set thresholds in the heuristics make it challenging to adapt heuristics-based models to other datasets without manually adjusting them. Therefore, machine learning models has been proposed for identifying search tasks (Figure 2.2), including semi-supervised clustering (Du et al., 2018; Kotov et al., 2011; Wang et al., 2013a), Bayesian approaches (Li et al., 2014a; Mehrotra et al., 2016; Mehrotra and Yilmaz, 2017), and graph-based clustering (Lucchese et al., 2011, 2013; Sen et al., 2018).

2.4.1 Graph-based clustering

To identify queries representing the same search task, the Query Clustering based on Weighted Connected Components (QC-WCC) (Lucchese et al., 2013) is a query clustering based on weighted connected components. QC-WCC does not need the number of clusters beforehand. It relies on the construction of a graph where the nodes correspond to the queries and the edges are weighted according to the similarities between the queries. Every query is a node in the graph. The similarities are based on two features: one content-based computing a Jaccard index on tri-grams, and the other semantic-based exploiting Wikipedia and Wiktionary to infer the semantics. The graph is then pruned by removing weak edge distances and the query cluster are obtained from the connected components remaining in the graph.

Query Clustering based on Head-Tail Components (QC-HTC) (Lucchese et al., 2013) is a computationally simpler algorithm based on QC-WCC, although less accurate. It exploits the sequential nature of queries to decrease the computational complexity of the graph based method. The QC-HTC algorithm first builds sequences of queries according to the distance between them, creating the first set of clusters, then takes the first and last queries of a cluster to represent the set and group it with other sets depending on query distances (Lucchese et al., 2013). Using only head and tail queries in each cluster avoids the computation of the full similarity graph required for QC-WCC, making QC-HTC less computationally expensive.

Tempo-Lexical Context driven Word Embedding (QRY-VEC) (Sen et al., 2018) improves the QC-WCC (Lucchese et al., 2013) algorithm for search task identification by using word embedding similarities instead of lexically based similarities. Queries for the same tasks in search logs tend to have a low lexical similarity (Sen et al., 2018), which affects methods that rely on lexical similarities or character-level information (Lucchese et al., 2013; Du et al., 2018). Some queries might have no lexical similarity but could be highly related from a semantic standpoint. Indeed, semantic similarity represents a better way to detect queries related to the same task. Queries for the same task clusters tend to be semantically similar rather than lexically similar, as queries in the same tasks contain more synonym words than exact words (Völske et al., 2019; Lucchese et al., 2013).

Because of this limitation, instead of relying on lexically based similarities and retrieved documents from the Wikipedia collection (Lucchese et al., 2013), QRY-VEC uses the cosine similarity on a tempo lexical word representation and documents retrieved from the ClueWeb12B (Carterette et al., 2016; Callan, 2012) collection. The ClueWeb12B dataset comprises a more extensive set of web documents. It retrieves information even for terms that are not expository, which are more challenging to match with Wikipedia documents because of vocabulary mismatch. Furthermore, instead of segmenting queries inside time sessions, QRY-VEC identifies global search tasks for clustering. As a baseline, QRY-VEC considers cosine similarities with pretrained word embeddings (Mikolov et al., 2013). The tempo lexical word representations surpass the baseline, providing a more accurate method for search task identification.

2.4.2 Semi-supervised clustering

A pairwise approach (Kotov et al., 2011) uses a binary classification model to determine if query pairs pertain to the same search task. However, a post-processing step is needed to determine the search tasks in the query log (Mehrotra et al., 2016).

Bestlink SVM (Wang et al., 2013a) first trains a support vector machine to detect if a pair of adjacent queries from a user pertains to the same search tasks or not. Then, it clusters the related queries in the search log using the SVM output to establish links between queries. Bestlink SVM uses a backward context from users' queries to improve the task clustering results. Links between queries are established only with past queries in order to leverage the chronological structure of the log. For the links with past queries, a pairwise parameter allows establishing if two queries should be related or not. For instance, if the retrieved URLs are the same, if the clicked URLs are the same, or if a query is a sub-query of another query, there must be a link between them. No link between queries happens when queries are different, and the intersection of the retrieved URL sets is null. Given that very different queries can be semantically related, they can provide similar sets of related queries. Similar sets allow an automatic labeling scheme for search logs, a significant contribution because manual labeling tends to be expensive and sometimes there are only small annotated logs available. Likewise, the two most important features from the query representation in Bestlink SVM includes the cosine similarity between query embeddings and the similarity between clicked URLs.

Context Attention based Long Short-Term Memory (CA-LSTM) (Du et al., 2018) uses neural networks and graph-based clustering to build a pipeline for task identification. First, it learns to segment search logs in a supervised way. CA-LSTM relies on RNNs instead of SVMs, using both backward and forward queries to provide context while training the RNN to detect if a pair of adjacent queries pertain to the same task or not. Then, it uses the unsupervised QC-HTC clustering method to identify task clusters in the segmented search log.

2.4.3 Bayesian approaches

Latent Dirichlet Allocation with Hawkes processes (LDA-Hawkes) (Li et al., 2014a) combines LDA with Hawkes processes to identify and label search tasks from query logs. LDA performs topic modeling, identifying semantically related queries from different users, while Hawkes processes take into account time lapses between query timestamps in individual query sequences, assigning temporally close queries to the same search task.

Some search tasks are atomic, where a sequence of related queries solves the information needs. However, often information needs are complex. Users perform complex tasks by executing related subtasks, some of which are also complex information needs. Given the relationship between complex tasks and subtasks, a hierarchical model could also serve to cluster search queries (Mehrotra et al., 2016; Mehrotra and Yilmaz, 2017).

The Distance Dependent Chinese Restaurant Process (DD-CRP) (Blei and Frazier, 2011) provides an approach to compute task relatedness from the hierarchies of tasks. The CRP method is a non-parametric Bayesian approach. It was proposed to model random groups of non-exchangeable data. It assumes a restaurant with an infinite number of tables. Customers enter the restaurant in tandem; they are assigned to a nonempty table based on the number of already existing customers in the table, or to an empty table depending on a scaling parameter. In this case, tables represent search subtasks, while customers are entries in the query log (Blei and Frazier, 2011; Mehrotra et al., 2016).

The hierarchies of search tasks are extracted from the query logs using a representation that models queries as a linear combination of word vectors for its terms. Weights in the linear combinations represent the maximum likelihood for each term according to the query task relationships. (Mehrotra et al., 2016; Sen et al., 2018). Search logs are considered as convoluted structures of complex tasks and subtasks. Thus, given an on-task query collection, a DD-CRP model enhanced with word embedding distances determines subtasks in the search log (Mehrotra et al., 2016). The proposed approach does not need the number of subtasks per complex task. It also uses word embeddings to deal with lexically similar queries that have different semantics, a kind of queries where lexical based methods like bag of words or TF-IDF can not discriminate.

The DD-CRP method models the query links to other queries and not to sub-tasks. Distances between queries are based on cosine distance of word embeddings. DD-CRP uses skip-gram based embeddings to represent queries and computes a weighted average vector per query. Weights in the average come from the frequency of the terms because low-frequency words could be key for helping to identify subtotals. To compare results, (Mehrotra et al., 2016) considers three baseline methods: QT-HTC (Lucchese et al., 2013), LDA (Blei et al., 2003) learned over documents containing aggregation of queries, and a vanilla CRP model to process queries in the AOLQL (Pass et al., 2006) dataset (Section 2.1). The AOLQL is augmented with related queries from available APIs of search engines.

A qualitative approach is used to validate the results. The subject chosen is weddings. For that subject, the proposed method is the only one to detect the subtasks. A quantitative evaluation is challenging because there are no publicly available datasets with tasks - subtasks labels. Thus, the evaluation is carried out based on randomly selecting pairs that the method detected for the same subtasks, and asking users to assess if the queries pertain to the same subtasks or not. One hundred repetitions were performed, and the results of the DD-CRP method surpassed all the three baseline methods (Mehrotra et al., 2016).

Bayesian Rose Trees (BRTs) (Blundell and Teh, 2013) extend the hierarchy of search tasks to multiple levels, modeling search tasks using an agglomerative clustering approach (Mehrotra and Yilmaz, 2017). Existing agglomerative clustering methods consider nodes in trees as binary, but complex tasks could have one or more related subtasks. Thus, it is essential to use an agglomerative method that supports multiple children from nodes. BRTs provide a clustering method with multiple children in each node. They are adapted and extended to model query hierarchical structures in an unsupervised process (Mehrotra and Yilmaz, 2017).

BRTs represent data samples using binary features. But to better model relationships between queries, binary features can not properly represent the semantic relationships between queries. Thus, queries are represented as a set of affinities. The marginal distribution of the data is modeled as a Bernoulli distribution when using binary features to encode data. When using affinities, the Bernoulli distribution is replaced by a Gamma-Poisson distribution. Query affinities include (Mehrotra and Yilmaz, 2017):

1. Query term affinity uses edit distance and Jaccard coefficients between query terms, as well as the proportion of common terms and the cosine similarity between term sets

2. Retrieved URL affinity uses edit distance and Jaccard coefficients between URLs, given that similar information needs tend to produce similar sets of URLs

3. Time / user session affinity is a binary feature indicating if the queries are part of the same time session and if they were issued by the same user

4. Query embedding affinity uses word embeddings, which are averaged together to form one vector per query. Cosine similarity between query vectors model the semantic relationships of queries. Word embeddings for query terms are custom-trained using the skip gram model and the AOL dataset

The unsupervised process using BRTs starts with all queries in the log as a subtree. In each step, trees are clustered using one of three operations: join, collapse, or absorb. Join creates a top node and assigns each subtree as a child to it. Collapse creates a top node whose children are the children of all the merged subtrees; the top node remains while the nodes of the merged subtrees disappear. Absorb assigns a subtree to the children of the top node; by doing so, the merged subtree node becomes a child of the top node. The clustering operation to perform and the subtrees to merge are selected to maximise the quality of the hierarchical representation. The merging process stops when possible merging operations on remaining subtrees do not improve the quality of the hierarchical representation (Mehrotra and Yilmaz, 2017).

To avoid subtrees in nodes representing atomic tasks, a task coherence computation allows the collapse of all subtreees if such coherence surparsses a given threshold. This step is necessary because queries in atomic search subtasks are strongly related semantically - a phenomenon not observed in complex search tasks with multiple related subtasks. For computing task coherence, the Pointwise Mutual Information (PMI) metric is utilized. It is calculated using the AOL dataset query terms. Evaluation of the task identification effectiveness of the model is performed using (Lucchese et al., 2011) dataset, a subset of the AOL dataset with manual labels for search tasks in time sessions. Pairwise F-score is the metric used for assessing model performance. Also, tests are performed using query term prediction to test the hierarchical model effectiveness in task identification applications (Mehrotra and Yilmaz, 2017).

Dataset	**Method**	**Result**
CSTE	QC-WCC (Lucchese et al., 2013)	0.471
	QRY-VEC word2vec (Sen et al., 2018)	0.473
	QRY-VEC tempo-lexical (Sen et al., 2018)	0.538
TGSST	QC-WCC (Lucchese et al., 2013)	0.812
	QC-HTC (Lucchese et al., 2013)	0.800
	DD-CRP (Mehrotra et al., 2016)	0.845
	BRT (Mehrotra and Yilmaz, 2017)	0.845
	QRY-VEC word2vec (Sen et al., 2018)	0.837
	QRY-VEC tempo-lexical (Sen et al., 2018)	0.840
WSMC12	QC-HTC (Lucchese et al., 2013)	0.851
	LDA-Hawkes (Li et al., 2014a)	0.871
	BRT (Mehrotra and Yilmaz, 2017)	0.878
	CA-LSTM (Du et al., 2018)	0.883
SUQLST	QC-HTC (Lucchese et al., 2013)	0.821
	LDA-Hawkes (Li et al., 2014a)	0.837
	BRT (Mehrotra and Yilmaz, 2017)	0.843
	CA-LSTM (Du et al., 2018)	0.851

Table 2.5: Search task identification performance for several query log datasets, including F_1 results for semisupervised, unsupervised, and Bayesian approaches.

2.4.4 Additional search patterns based on tasks

Personalization in search systems can be implicit or explicit. In explicit personalization, users directly specify their interests using tools such as forms, surveys, or buttons for ranking. However, users tend to ignore the steps needed to recollect information as they find the process for explicit personalization difficult or undesirable. Thus, implicit personalization is better suited to create models for users. In implicit personalization, search systems automatically infer users' features utilizing their search history. The automatic processing of search history for implicit modeling of users relies on the analysis of query logs, where search systems store data they get from tracking user queries (Hearst, 2009).

Search history from users include query sentences, timestamps and click through data. In click through data, we can find the clicks users perform after retrieving search results, while they are browsing resulting documents or reviewing web pages from results. Clicks help search systems assess the quality of their ranking and improve relevance in personalized results (Borisov et al., 2016; Hearst, 2009). They also help to model user intent and create distributed query representations (Zhang et al., 2019).

For query auto-completion (QAC) or query suggestions, several works consider click-through data or QAC logs (Bar-Yossef and Kraus, 2011; Boldi et al., 2009; Cao et al., 2009; Li et al., 2017; Sun and Lou, 2014). However, the two types of logs are considered separately. Given the user's interaction with the search system, these two logs are strongly correlated. A user performs a query search, generating entries in the QAC log; then, the user analyzes the retrieved documents, generating entries in the click log. If results are not satisfactory, a new query search starts, and a subsequent click log entry is added. Therefore, users' behaviors extracted from both logs have a sequential correlation. Also, users' behaviors usually are stable during periods of time (Li et al., 2017).

(Li et al., 2017) leverage both types of logs to extract and characterize user search behaviors from them. The click log provides context to the QAC log and vice-versa. Search logs are clustered according to the behavior of the users, looking for correlations at both logs. Experiments to prove the correlation between both logs, experiments include an LDA method that models logs separately, a Hidden Markov Model (HMM) that takes advantage of the correlation between both logs, and a contextual LDA, which also jointly models both logs. Using the log predictive likelihood as a metric, results prove the correlation between both types of logs: HMM and contextual LDA generate a better metric than LDA. Also, contextual LDA outperforms HMM in the log predictive likelihood, proving that it better models the behavior correlation in both types of logs.

The QAC log is not a low resolution log, which registers the query for the last keystroke and the run query. Instead, it is a high resolution log. It saves ten suggested queries for every keystroke. Thus, every keystroke generates its own session of queries in the log. Features from the high resolution log include user behavior characteristics, like the variance of typing speed, the length of the keystrokes, or the average speed. For example, if the user has a low variance in the typing speed, it can suggest that the user ignores the suggestions. Alternatively, if the user types the whole query fast, it can suggest that the user is proficient in the subject she is looking for. Additional tests were performed on the correlated model. A QAC task used contextual LDA along with a two-dimensional click model (TDCM) (Li et al., 2014b), outperforming previous methods for QAC. For the click prediction and relevance ranking, Contextual LDA was used with the Bayesian Sequential State (BSS) model (Wang et al., 2013b), also outperforming previous works.

Personalization can also include individual features or aggregate information of search users, where other users' interactions provide clues about potential suggestions. In particular, implicit personalization can leverage acquired knowledge from other users' interactions to better predict user needs. Similarly, personalization could be content-based and users-based. In content-based personalization, suggestions can be based on context related to users' query terms. In users-based personalization, information from users with similar characteristics helps to create tailored results (Hearst, 2009).

Search engine users change their behavior when analyzing search results already seen. For instance, in file browsers, users prefer to browse the file tree instead of using the search bar for finding files. Furthermore, users tend to repeat a percentage of the previous searches; thus, the inclusion of previous queries in suggestions helps to improve user productivity. In general, methods for assisting search engine users can help to predict the information users are likely to read or need. However, automatic suggestions can interfere with the users' intentions, hindering the usability of the search engine (Hearst, 2009). Another change in user behavior can be seen when analyzing mobile and personal computer logs, suggesting that users change their behavior when searching on different platforms (Li et al., 2017).

Another aspect in user modelling is the analysis of user behaviors for subsequent queries. The GEN encoder (Zhang et al., 2019), the query representation based on user intent (Section 2.3), can be used for modeling those past user behaviors for the same query. Distances of subsequent pairs of queries follow a bimodal distribution when using the GEN encoding. This bimodal distribution allows the identification of query reformulation behavior from four scenarios (Zhang et al., 2019):

- Topic change happens when the user focuses her attention on other tasks, generating an unrelated query
- Exploration includes a related query inside the same topic
- Specification refers to queries aimed at narrowing the results of the previous query execution
- Paraphrasing includes changes to the query that represents the exact same intent

Results using the GEN encoder bimodal distribution for query reformulation were compared to a number of query encodings, including TF-IDF (Robertson, 2004), USE (Cer et al., 2018), and BERT (Devlin et al., 2019). A 30 minutes gap was used for segmenting sessions (Mehrotra and Yilmaz, 2017; Wang et al., 2013a). Sessions without clicks or with less than three queries were ignored. The GEN encoder surpassed the performance of all the other query encodings used for comparison; however, it still trails the average performance of human labelers (Zhang et al., 2019).

The long-tail effect in machine learning (Section 2.2) can also be observed in queries run by search engine users. The long-tail effect in search query logs refers to the appearance of queries with few or no previous coincidence.

However, they can be strongly related to other queries in terms of user intent. This could happen, for example, when there are orthographic errors. Using the GEN encoder and Approximate Nearest Neighbors (ANN), the long tail is reduced in half, with a reasonable response time - 10ms for 700M query index (Zhang et al., 2019). The ANN implementation uses hierarchical navigable small world graphs (HNSWG) (Malkov and Yashunin, 2018).

2.4.5 Limitations of existing search task models

Most models for search task identification are non-parametric (Du et al., 2018; Li et al., 2014a; Lucchese et al., 2013; Mehrotra et al., 2016; Mehrotra and Yilmaz, 2017; Sen et al., 2018). Those models will grow as the size of the input query log grows, becoming computationally expensive, especially when processing large input datasets.

Also, some models requiring user identifiers can not be used in user-independent modeling scenarios (Du et al., 2018; Hagen et al., 2013; Li et al., 2014a; Mehrotra and Yilmaz, 2017), raising issues regarding user privacy. When users carry out their information seeking, they can release personal information during their interactions with search systems (Craswell et al., 2020a), which brings privacy front and center, as privacy is crucial for the broader field of ethics in intelligent systems (Manikonda et al., 2018; Stahl and Wright, 2018).

Moreover, some models (Du et al., 2018; Wang et al., 2013a) require labeled datasets to train the supervised component of the model. Whether it is a neural network or a support vector machine, labels are indispensable to train them. But labeled datasets are challenging to create because of the resources required to manually label them. When considering large datasets, this limitation is even more critical (Wang et al., 2013a). Also, large collections are difficult to find publicly available, mainly because of user privacy issues and commercial concerns that could arise when other search engines exploit the information (Craswell et al., 2020a).

Furthermore, few models leverage clicked URLs, even though they are strongly related to user search intent (Zhang et al., 2019). For instance, two lexically different queries with the same clicked document are likely to reflect the same information need (Mehrotra and Yilmaz, 2017; Zhang et al., 2019). Also, a similarity metric based on clicked URLs is the second most important similarity in Bestlink SVM (Wang et al., 2013a), just after the most important one, which is the semantic similarity of the queries.

Another limitation of most models is the lack of realtime response times, which relegates them to offline modeling only. Updating graphs or trees when a user runs a query can cause delays in search systems, affecting the interaction with users in realtime. Most search systems need to generate results interactively. For example, systems for query autocompletion, query suggestion, or ranking based on tasks need to have model results as the user runs the query (Ahmad et al., 2019; Boldi et al., 2009; Borisov et al., 2016; Li et al., 2017). Similarly, in conversational search, systems can have strict time thresholds to generate answers, and systems aggregating results from several indexes can discard an index result if it exceeds a certain limit of time (Arguello and Capra, 2016; Zamani and Craswell, 2020). Such time thresholds are normally in the order of milliseconds so that it is possible to interact with users (Zamani and Craswell, 2020).

At the same time, most models are monolingual, supporting mostly users queries in English only. But users worldwide can access search systems in many languages. The use of custom-trained word vectors, which use AOLQL for training, or the use of collections like ClueWeb12B or Wikipedia (Du et al., 2018; Lucchese et al., 2013; Mehrotra and Yilmaz, 2017; Sen et al., 2018; Wang et al., 2013a), limit the number of languages the model can support, normally to English. Though it is possible to custom-train for other languages, few languages have large collections available for training text representations (Joshi et al., 2020). The number of languages is even smaller when considering publicly available query logs (Section 2.1). On top of that, it would be necessary to manually select the custom-trained vectors or the specific collection, or create a pipeline, inserting another method that allows selecting the word vectors depending on the query language.

2.5 Search log segmentation

As a first step in multiple search pattern models, a search log segmentation is performed to help in the subsequent modeling of user interactions (Hearst, 2009). Once the segments in the search log are extracted, it is possible to identify search tasks (Mehrotra and Yilmaz, 2017; Sen et al., 2018; Wang et al., 2013a), analyze query reformulation behaviors (Tamine et al., 2020; Zhang et al., 2019), improve query suggestions (Ahmad et al., 2019; Tamine et al., 2020), or optimize document ranking (Ahmad et al., 2019).

To perform the search log segmentation, a chronologically ordered log of search queries is partitioned into smaller sequences of queries. Those query log partitions are commonly known as sessions (Gayo-Avello, 2009; Hagen et al., 2013). The boundaries for the query log partitions lie in pairs of adjacent queries. To determine if a query pair is a boundary, one may use time spans between the queries (Lucchese et al., 2013). If the time span is larger than a certain threshold, the query pair is considered a segment boundary, which means that each query belongs to a different segment. The time span for the segmentation can be 26 minutes (Lucchese et al., 2013), 30 minutes (Mehrotra and Yilmaz, 2017), or 90 minutes (Hagen et al., 2013), depending on statistical analyses of the query logs. By the same token, in some heuristic-based approaches (Gomes et al., 2019; Hagen et al., 2013), segmentation based on time is the first heuristic to be considered.

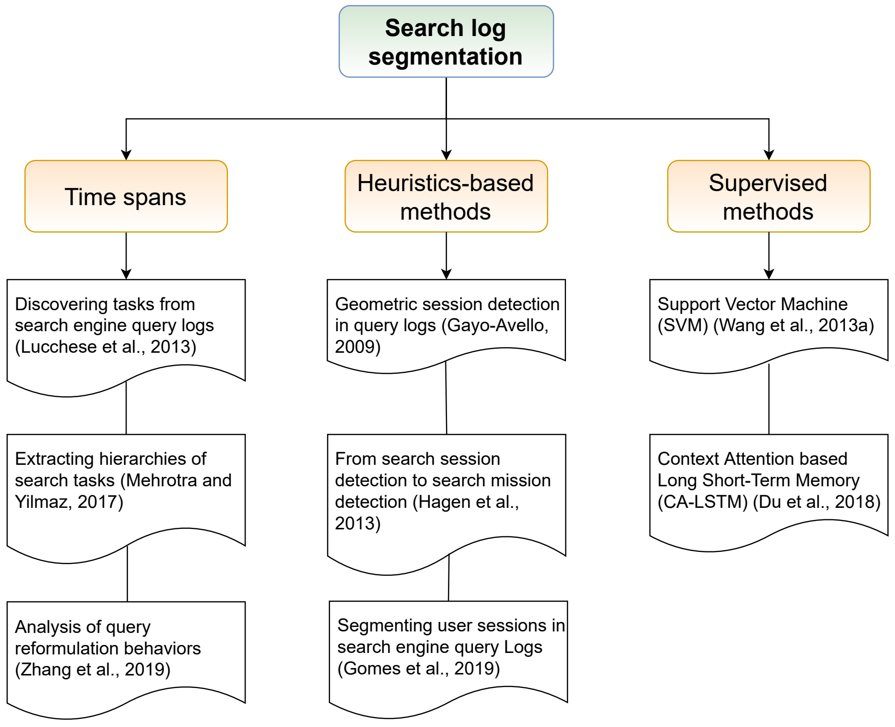

Search log segmentation
Time spans
Heuristics-based methods
Supervised methods
Discovering tasks from search engine query logs (Lucchese et al., 2013)
Extracting hierarchies of search tasks (Mehrotra and Yilmaz, 2017)
Analysis of query reformulation behaviors (Zhang et al., 2019)
Geometric session detection in query logs (Gayo-Avello, 2009)
From search session detection to search mission detection (Hagen et al., 2013)
Segmenting user sessions in search engine query Logs (Gomes et al., 2019)
Support Vector Machine (SVM) (Wang et al., 2013a)
Context Attention based Long Short-Term Memory (CA-LSTM) (Du et al., 2018)

Fig. 2.3: Recent methods for search log segmentation, including segmentation based on time spans, heuristics-based methods, and supervised models.

Nonetheless, according to the analysis of query logs, users tend to interleave multiple search tasks in a single time session. Also, some tasks are performed during multiple time sessions. Thus, the search process for complex tasks could be iterative, lengthy, and multistage, with shifting goals (Mehrotra and Yilmaz, 2017). More sophisticated approaches learn models (Du et al., 2018; Wang et al., 2013a) or establish a cascade of heuristics (Gayo-Avello, 2009; Gomes et al., 2019; Hagen et al., 2013) to determine the boundaries in the sequential query log (Table 2.3).

The cascade of heuristics (Hagen et al., 2013) uses a time span between queries as the starting rule in the procedure. The cascade method then uses additional heuristics to detect segments – logical sessions – in the query log before performing the identification of search tasks. The heuristics-based session segmentation method (HBSSM) (Gomes et al., 2019), an improved heuristics-based method, leverages pre-trained word embeddings to provide semantic similarity measures to segment the query logs, complementing se-

Dataset	Method	Metric	Result
WSMC12	Geometric (Gayo-Avello, 2009)	F_1	0.886
	Heuristics (Hagen et al., 2013)	$F_{1.5}$	0.946
	HBSSM (Gomes et al., 2019)	F_1	0.915
WSMC12	QC-HTC (Lucchese et al., 2013)	Accuracy	0.851
	BRT (Mehrotra and Yilmaz, 2017)	Accuracy	0.878
	CA-LSTM (Du et al., 2018)	Accuracy	0.883
SUQLST	QC-HTC (Lucchese et al., 2013)	Accuracy	0.821
	BRT (Mehrotra and Yilmaz, 2017)	Accuracy	0.843
	CA-LSTM (Du et al., 2018)	Accuracy	0.851

Table 2.6: Search log segmentation results for several existing query log datasets.

mantic relatedness with temporal, lexical, and clicked URLs heuristics. Thus, a cascade of heuristics is applied to each query pair, and manually set parameters provide thresholds in each heuristic to generate the output of the model. However, as mentioned before (Section 2.4), several manually set thresholds in the rules make it challenging to adapt heuristics-based methods (Gayo-Avello, 2009; Gomes et al., 2019; Hagen et al., 2013) to other datasets without manually adjusting them.

Instead, models can learn to segment logs using task labels, as tasks are recognized as a good atomic unit to divide search logs (Hearst, 2009; Lucchese et al., 2013). Formally, the task segmentation is defined as follows. Let us consider a sequence of queries $S = \{q_1, ..., q_n\}$, a collection of pairs of successive queries $C = \{(q_1, q_2), (q_2, q_3), ..., (q_{n-1}, q_n)\}$ and its respective known labels $L = \{l_1, ..., l_{n-1}\}$ where $l_i \in \{0, 1\}$, $\forall i \in [\![1; n-1]\!]$. The task segmentation goal is to correctly predict the labels of unseen query pairs, where $l_i = 1$ or $l_i = 0$ indicate if q_{i+1} and q_i belong to the same search task or not, respectively. Support Vector Machines (SVMs) or neural networks can be used to create the segmentation model (Du et al., 2018; Wang et al., 2013a). In particular, CA-LSTM (Du et al., 2018) uses RNNs for segmenting search query logs. Three sequences encode the data, in contrast with other models (Liu, 2017), where a single input encodes data, and a bidirectional RNN processes the single input to produce the model results. Moreover, exploiting the context from query logs in CA-LSTM is necessary to generate an improvement over the existing methods and surpass its baseline approach (Mehrotra and Yilmaz, 2017). The performance of the CA-LSTM architecture depends on at least six adjacent queries for obtaining its best metrics, requiring context around the pair of adjacent queries to properly segment the search query log.

2.5.1 Limitations of existing segmentation approaches

Even though clicked URLs are related to user search intent (Section 2.4), when segmenting a log for interactive user assisting applications, it is not possible to wait until the user clicks the URL. A similar scenario arises if the model requires forward and backward queries to generate a context for the segmentation model (Du et al., 2018), which limits the use of the segmentation methods to offline modeling.

Also, heuristics-based models pose some limitations. The thresholds for the rules are set depending on a manual statistical analysis of the search query logs (Gayo-Avello, 2009; Hagen et al., 2013; Gomes et al., 2019). No method is included to adjust the thresholds when the search query log registers new entries, which requires a manual update of the heuristics. Also, lexically different queries can pertain to the same information need (Mehrotra and Yilmaz, 2017; Zhang et al., 2019); thus, semantic comparisons are a better choice when analyzing user search queries.

Chapter 3
Segmenting search query logs by learning to detect search task boundaries

The important thing is not to stop questioning. Curiosity has its own reason for existing. One cannot help but be in awe when he contemplates the mysteries of eternity, of life, of the marvelous structure of reality.

Albert Einstein

As search log segmentation is the first step in many search task models, analyses of search patterns, and user assisting applications (Chapter 2), in this chapter we propose a neural model to learn to detect task boundaries in search query logs. Previous models for task boundary detection rely on clicked URLs and surrounding queries to properly segment the query logs, determining if adjacent queries are part of the same search task or not. However, waiting for clicked URLs or future consecutive queries could render the use of these methods unfeasible in user assisting applications that require model results in realtime. The model proposed in this chapter uses only pairs of adjacent queries and their time span, generating results suited for realtime user assisting applications, with improved accuracy over existing approaches. We also show the advantages of fine-tuning the proposed model for adjusting the architecture to a small annotated collection.

3.1 Introduction

Search systems are an essential component of the interactions of users with the World Wide Web. They are crucial to help users access the ever-increasing amount of information available. A wide range of needs and desires from users are converted to queries and submitted to available search engines (Hearst, 2009). Both automatic and manual analysis of search interactions from users enables the modeling of users' search patterns. Extracted patterns help to personalize search engine interactions. In turn, personalization allows the modification of search results depending on the user, the retrieval of advertisement according to user interests, the suggestion of queries related to user information needs, and other tasks designed to support the user while she performs her search tasks (Du et al., 2018; Hearst, 2009).

Search system logs register the queries users run in the search engines to complete their search tasks. Mining those logs allows the identification of search tasks. Search session segmentation is the first step in multiple methods for search task identification(Hearst, 2009; Lucchese et al., 2013; Mehrotra and Yilmaz, 2017). In session segmentation, a sequential log of search queries is partitioned into smaller sequences of queries. The boundaries for the query log partitions lie in pairs of adjacent queries. To determine if a query pair is a boundary, one may use time spans between the queries (Hagen et al., 2013; Lucchese et al., 2013; Mehrotra and Yilmaz, 2017; Wang et al., 2013a). If the time span is larger than a certain threshold, the query pair is considered a session boundary, which means that each query belongs to a different session. More sophisticated approaches use heuristics-based models (Gomes et al., 2019) or neural networks (Du et al., 2018) to determine the boundaries in the sequential query log. However, the use of clicked URLs (Gomes et al., 2019) or adjacent queries (Du et al., 2018) in the search log - in both backward and forward directions - represents a limitation in practical setups, especially in user supporting applications that require modeling on the fly. In such applications, waiting for the clicked URL or future queries to populate the model input might be unfeasible.

Hence, we propose a bidirectional recurrent neural network (RNN) architecture that segments pairs of adjacent queries based on semantic representations of queries and time spans between queries to provide temporal information. The segmentation model determines if adjacent pairs of queries represent a boundary between search tasks, without relying on clicked URLs, character-level representations, or surrounding queries for optimum performance. Furthermore, we test the segmentation architecture in a fine-tuning setup (Angermueller et al., 2016) to know how the model adapts to a small log dataset.

3.2 Related limitations

To support users during the information seeking process, it is crucial to segment query logs correctly according to their search tasks. Some methods based on time spans between queries have been proposed to extract task-based sessions from query logs (Lucchese et al., 2013). However, when analyzing search logs, multiple information needs overlap because users tend to solve various search problems during their interactions with search systems (Hagen et al., 2013; Hearst, 2009; Lucchese et al., 2013). Because of this multitasking, a time-based session might contain more than one search task (Lucchese et al., 2013; Sen et al., 2018).

Supervised models and heuristics based methods has been proposed as well (Section 2.5). But such approaches have some limitations. Heuristics-based methods (Hagen et al., 2013; Gomes et al., 2019) have several manually set thresholds in their rules, making it challenging to automatically adapt to other labeled datasets if needed. Should an update is needed, a manual analysis of the new dataset is required to update the thresholds of the rules, while machine learning models could be automatically fine-tuned or retrained.

Likewise, the use of character-level information (Du et al., 2018; Hagen et al., 2013; Gomes et al., 2019; Lucchese et al., 2013) can misrepresent queries because lexically similar queries can have entirely different semantic contents. A difference in semantic content implies that lexically similar queries could pertain to non-related search tasks (Zhang et al., 2019). A related problem arises when evaluating similarities in queries based on term likelihoods (Mehrotra and Yilmaz, 2017). Also, the need for clicked URLs (Gomes et al., 2019) or adjacent queries to properly segment search logs (Du et al., 2018) could be problematic. Some user supporting applications need on the fly results, thus, they can not wait for clicked URLs or future user queries to provide forward context. Likewise, when dealing with simple search tasks like fact-finding, typically, a single query could solve the information need (Hearst, 2009); thus, there is no related context available. Similarly, the average amount of queries per search task is less than four, on the basis of the data released by widely used search engines (Gayo-Avello, 2009; Völske et al., 2019). For instance, there is an average of 3.2 queries per task (Hagen et al., 2013) and 3.5 queries per task (Sen et al., 2018) in publicly available search task datasets.

Because of these limitations, we propose a bidirectional RNN (BiRNN) approach that relies on a semantic representation of queries in search logs, adding time spans between queries to provide temporal information. The proposed session segmentation architecture can be fine-tuned to automatically adapt it to new datasets and does not need character level representations, clicked URLs, or adjacent queries in the search log to achieve its maximum performance.

3.3 A new task segmentation approach

Our proposed BiRNN-based architecture (Figure 3.1) processes search query vector representations and learns to identify task boundaries in pairs of suc-

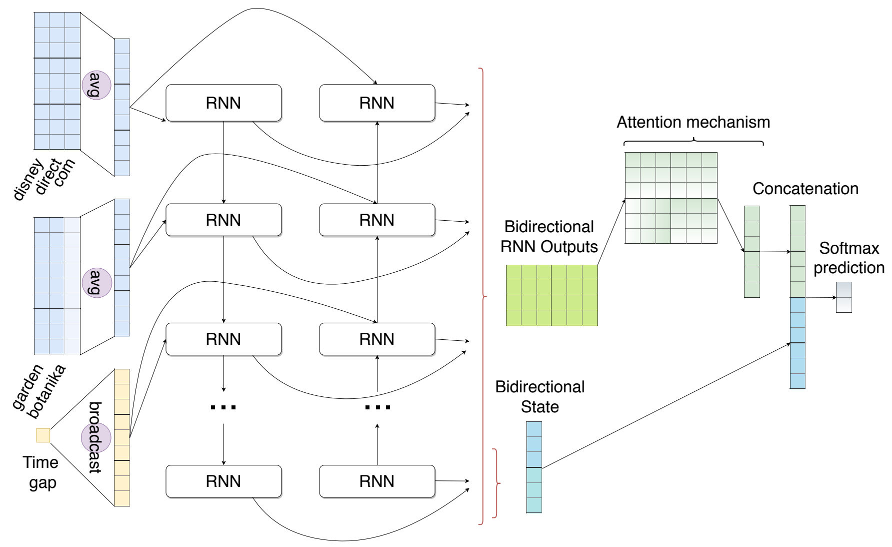


Fig. 3.1: Segmentation architecture with a bidirectional recurrent neural layer along with an attention mechanism. Input samples include the query pairs and the time span between them. Query examples in the image come from the first two entries in the CSTE dataset.

cessive queries. The architecture relies on a bidirectional recurrent layer, a commonly used layer for processing sequential information (Angermueller et al., 2016; Graves, 2012; Liu, 2017). Furthermore, some studies in the field of Neural Machine Translation showed that intermediate output states from recurrent layers could significantly improve the performance of the initial models (Luong et al., 2015). Therefore, following these conclusions, we include an attention mechanism in the segmentation architecture to leverage the information from the intermediate output states. Additionally, completely different queries from a lexical standpoint might represent very similar information seeking needs. For example, "constantinople" and "istanbul archeology" could represent semantically related information needs because Istanbul and Constantinople refer to the same city, but there is no lexical similarity between the two queries (Hagen et al., 2013; Sen et al., 2018). These discrepancies could affect the ability of methods that rely on character-level information to extract task patterns from query logs. Thus, we do not use character-level information to encode queries. Instead, we use word embeddings to capture the semantic meaning of the queries.

3.3.1 Input representation

We use word embeddings to represent queries in the search log (Figure 3.1). As queries are short texts, every query is mapped to a unique vector, which allows us to keep both the syntactic and semantic information of the query content (Sen et al., 2018). Thus, we define the vector representation of the query $\boldsymbol{q_i}$ as the average of all its word embeddings $\boldsymbol{w_i}$ in order to get a single vector per query, a query representation approach used in several recent studies (Gomes et al., 2019; Mehrotra and Yilmaz, 2017; Sen et al., 2018; Zamani and Croft, 2016a,b). Also, we concatenate the time span in seconds between the timestamps of a query pair: $d_{time\_span} = timestamp(q_{i+1}) - timestamp(q_i)$, where *timestamp* is a function that gets the timestamp in seconds of a given query. If needed, the d_{time_span} value is broadcast[1] to match the input dimensions.

For the experiments to evaluate the model with adjacent queries for context, each adjacent query vector is appended to the query pair representation, respecting the sequential order in which queries appear in the search log. In this case, the input contains $\boldsymbol{q_{-m}}, \ldots, \boldsymbol{q_0}, \boldsymbol{q_1}, d_{time\_span}, \ldots, \boldsymbol{q_n}$, where m is the number of backward queries and n is the number of forward queries.

3.3.2 Recurrent neural network

An RNN is a type of neural network ideally suited to process sequential information. In contrast with previous methods (Lucchese et al., 2013; Mehrotra and Yilmaz, 2017; Sen et al., 2018; Wang et al., 2013a), RNNs tend to scale adequately and perform well when dealing with sequential data (Angermueller et al., 2016). RNNs store information from input sequences by using iterative function loops (Graves, 2012; Rajaraman and Ullman, 2011). To store information, RNNs have a hidden state vector, which works as a memory while the network processes the sequence in the forward direction (Rajaraman and Ullman, 2011). BiRNNs were proposed to process the input sequence simultaneously in a forward and backward direction. To do so, they compute forward hidden states and backward hidden states. Concatenating forward and backward hidden states enables the generation of outputs that leverage information from preceding and following steps in the sequence (Bahdanau et al., 2015; Luong et al., 2015).

In our proposed approach, we add an attention mechanism (Bahdanau et al., 2015; Luong et al., 2015) at the output of the bidirectional RNN (Figure 3.1). The forward and backward hidden states of the bidirectional recurrent layer are concatenated to the attention mechanism output before applying dropout. The proposed BiRNN-based model uses a fully connected

[1] `https://docs.scipy.org/doc/numpy-1.15.0/user/basics.broadcasting.html`

layer that takes the dropout result as input (Figure 3.1). Then, Softmax is used to generate the results in the fully connected layer, indicating if the two queries are part of the same task or not.

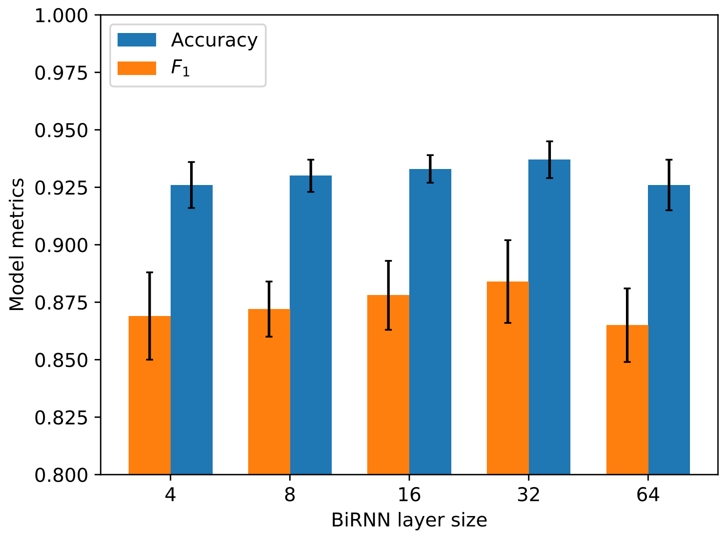


Fig. 3.2: Performance of the segmentation architecture considering different numbers of recurrent units for the bidirectional layer. Results were calculated using LSTM units and WSMC12 dataset.

3.3.3 Configurations of the proposed BiRNN approach

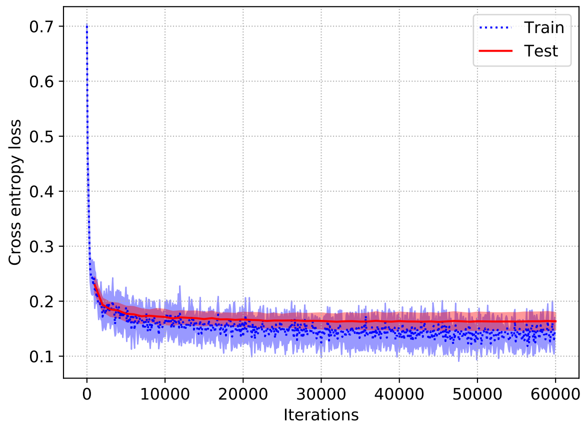
Four alternative configurations are studied for the proposed BiRNN approach as a result of two hyperparameters: time span location and recurrent unit type. The time span d_{time_span} can be placed in the initial layer (Figure 3.1) or the concatenation layer (Du et al., 2018), at the attention mechanism output. The latter is inspired by previous work (Severyn and Moschitti, 2015), where the late concatenation of extra features improved performances in text classification. Regarding recurrent unit types, in practical setups, the memory in the standard RNN only works effectively with information very close to the step that the network is processing. Thus, LSTM networks were proposed to enhance the performance of standard RNNs with long term information. LSTMs have the ability to save important information, forget information that is not relevant, and focus on the parts of the sequence that better serves the overall network performance. A two-tier configuration provides the ability to focus. It comprises a hidden state vector, which works as the working memory, and a cell state vector, which works as the long term memory. A popular variant of the LSTM network is the Gated Recurrent Unit (GRU), a simpler architecture to compute and implement (Cho et al., 2014). It relies on a single vector for memory purposes, decreasing the number of parameters needed (Rajaraman and Ullman, 2011; Cho et al., 2014). In the proposed architecture, we consider both types of recurrent units for the bidirectional layer.

3.4 Results and discussion

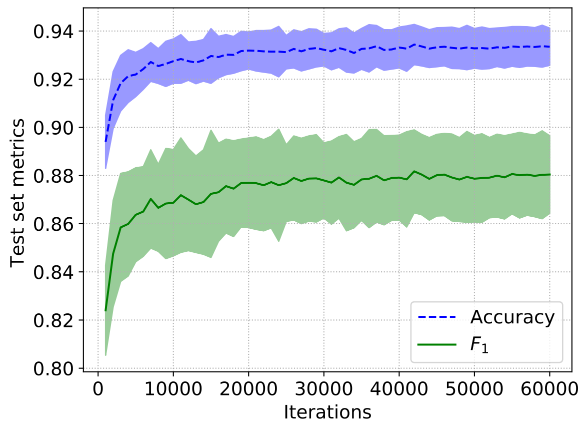
3.4.1 Technical considerations

Query	Timestamp	Task label
update media player	2006-03-09 09:14:24	2
update media player 9 series	2006-03-09 09:15:40	2
jim rome	2006-03-09 09:17:13	1
bank of america	2006-03-10 02:01:42	3
bank of america	2006-03-10 23:50:33	3
jt the brick	2006-03-11 00:05:18	1
storm watch	2006-03-11 01:45:50	12
weather	2006-03-11 01:46:34	12
dailyracingform	2006-03-11 08:22:09	5
dailyracingform	2006-03-11 08:22:09	5
search with plate numbers	2006-03-11 09:00:35	13
plate searches	2006-03-11 09:55:33	13
jim rome	2006-03-11 10:18:25	1
license plate	2006-03-11 19:22:35	13
bank of america	2006-03-11 19:29:06	3
plate search	2006-03-11 19:42:41	13
plate search	2006-03-12 00:06:49	13

Table 3.1: Entries in the WSMC12 dataset from user with ID 1045635. Dashed lines represent search task boundaries.



(a)



(b)

Fig. 3.3: Training curves for the bidirectional architecture. (a) Loss curves for both training and testing sets. (b) Accuracy and F_1 results for the testing set during the training of the system.

Two datasets were considered in this chapter: the WSMC12 and the CSTE datasets (Section 2.1). While task labels in the WSMC12 dataset correspond to search tasks inside user search sessions, with query timestamps (Table 3.1), task labels in the CSTE dataset correspond to cross-session search tasks, without grouping queries by user information or query timestamps. Therefore, to ensure the sequential order of queries in the CSTE dataset, query timestamp information comes from (Lucchese et al., 2013). Moreover, *Accuracy* and F_1 score enable the evaluation of the different approaches, performing 10-fold cross-validation. The Student's paired t-test provided the test for statistical significance.

We used the GloVe publicly available pre-trained word vectors (Pennington et al., 2014) for computing the search query vector representations. GloVe is a word vector space used in several recent works for short text representation and query encoding (Edo-Osagie et al., 2019; Marivate and Sefara, 2019; Rakib et al., 2018; Zamani and Croft, 2016a,b; Wang et al., 2016). The pre-trained word vectors have a dimension of 300 and are based on web corpora with 42 billion tokens from Common Crawl (Pennington et al., 2014). Word vectors have a coverage of 96.2% for the WSMC12 dataset and 84.5% for the CSTE dataset.

The proposed segmentation architecture is implemented using the Tensorflow deep learning framework – an advanced dataflow system that provides one of the most efficient implementations for RNNs (Angermueller et al., 2016). To train the segmentation model, we minimized the cross-entropy loss with the Adam optimizer (Figure 3.3). The Adam optimizer used the default TensorFlow parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. The learning rate was set to 10^{-4}, batch size to 256, layer size to 32, training steps to 6 x 10^4, and dropout to 0.3. To initialize the weights in all the layers, we used (Glorot and Bengio, 2010) initialization scheme. The bidirectional layer size was determined by testing several sizes with recurrent units and the WSMC12 dataset (Figure 3.2). The model performance improved from an accuracy of 0.926 at 4 recurrent units, to an accuracy of 0.937 at 32 recurrent units. However, a further increase in size affects performance. Accuracy falls to 0.926 at 64 recurrent units. Because of these results, we set the bidirectional layer size to 32 recurrent units.

3.4.2 Evaluation results

3.4.2.1 Comparison of results

We compared the proposed approach to the state-of-the-art HBSSM approach as a baseline. We also considered other supervised machine learning approaches like logistic regression, Support Vector Machine (SVM) with linear kernel and Radial Basis Function (RBF) kernel (γ=2, C=1), K-Nearest Neighbors (n=3), Naive Bayes, Gaussian Process Classification (GPC) with ($\kappa[\gamma$=1]), Random Forest, AdaBoost, Decision Tree, and Quadratic Discriminant Analysis (QDA) (Table 3.2). We also evaluated the proposed model in several context scenarios (Table 3.3), using CA-LSTM (Du et al., 2018) as a reference. Additional to the query pair representation (Section 3.3.1), we provided query texts and clicked URLs, when available, to the baseline HBSSM approach so that it was possible to compute all the heuristics specified for the model. We also flattened the query pair representation for methods that

Segmentation model	**WSMC12**		**CSTE**	
	Accuracy	F_1	Accuracy	F_1
Logistic regression	0.733	0.238	0.634	0.322
K-Nearest Neighbors	0.865	0.701	0.708	0.460
SVM, linear kernel	0.735	0.022	0.669	0.232
SVM, RBF kernel	0.742	0.055	0.671	0.105
Naive Bayes	0.773	0.306	0.643	0.223
QDA	0.323	0.428	0.534	0.553
Random Forest	0.862	0.742	0.725	0.502
AdaBoost	0.862	0.739	0.721	0.504
GPC	0.905	0.811	0.720	0.417
Decision Tree	0.882	0.777	0.759	0.541
HBSSM	0.886	0.813	0.656	0.627
BiRNN LSTM - time at AL	0.921	0.861	0.784	0.651
BiRNN GRU - time at AL	0.927	0.867	**0.789**	0.648
BiRNN LSTM	0.931	0.875	0.788	**0.663**
BiRNN GRU	**0.937**	**0.884**	0.751	0.604

Table 3.2: Model performance for WSMC12 and CSTE datasets. Architecture configurations include GRU recurrent units, LSTM recurrent units, and time span concatenation at the attention layer (AL). Differences between results have $p \leq 0.05$ for the Student's t-test.

required a unidimensional vector as input, ensuring that all methods got the query pair and its time span as input information.

Our proposed segmentation architecture obtains very satisfactory results, surpassing the remaining methods used for comparison. Moreover, GPC – the best performing method from the traditional machine learning approaches – gets results close to the HBSSM performance and matches the accuracy of CA-LSTM, even exceeding it in some context scenarios. Nonetheless, results from our proposed segmentation architecture outperform the previous recurrent model in all context scenarios. The trained BiRNN GRU architecture is also faster than the HBSSM baseline. We measured the query pair processing time using a CPU-only virtual instance with a CPU frequency of 2.397 GHz, taking the average time per query pair for all the pairs in the WSMC12 dataset. The trained BiRNN GRU architecture took around 15ms per query pair, while the HBSSM approach took around 61ms. This difference in execution time represented a speedup of 4 for our proposed architecture.

Also, we analyze the model with both GRU and LSTM recurrent units, as the decision of which to choose depends on the task and the dataset (Chung et al., 2014). Replacing the GRU cells with LSTMs decreases the accuracy of the segmentation architecture with the WSMC12 dataset. Accuracy drops from 0.937 to 0.931 (Table 3.2). Similar behavior is observed in the fine-tuning results for the CSTE dataset (Table 3.5), suggesting that the simpler GRU cell (Cho et al., 2014) is a better choice for the search segmentation model. Additionally, given the smaller number of parameters per recurrent unit, GRU models have another advantage: a faster training time. One training step with GRU cells takes around 20.15s, while one training step with LSTM cells takes around 31.68s, representing a speedup of 1.6 for the GRU model.

Likewise, concatenating the time spans to the attention mechanism decreases the performance of the proposed architecture. Accuracy drops from 0.937 to 0.927 with the WSMC12 dataset. With the smaller CSTE dataset, the best results are obtained when performing fine-tuning with time span information at the input sample representation as well (Table 3.5). Thus, the best scenario happens when the query vectors and the time span information are part of the input sample representation. This configuration allows the computation of hidden state vectors inside the bidirectional layer that rely upon both the query pair semantic content and the time between them. Similarly, the bidirectional output vectors depend on both semantic and time information to feed the attention mechanism.

No. of queries	**2**	**4**	**6**	**8**	**10**
CA-LSTM	0.863	0.878	0.904	0.898	0.903
BiRNN LSTM	0.929	0.921	0.920	0.915	0.912
BiRNN GRU	**0.935**	0.920	0.927	0.919	0.917

Table 3.3: Segmentation model accuracy for the WSMC12 dataset, with additional adjacent queries for recurrent architectures. Differences between results have $p \leq 0.05$ for the Student's t-test.

3.4.2.2 Impact of query context

We also analyzed how the model behaves when adding adjacent queries for session segmentation. We computed query vectors for each adjacent query and appended the query vectors to the original pair, respecting the order in which they appeared in the original search log. Half of the adjacent queries precedes the query pair, while the remaining half follows the query pair in the chronological search log. When the input includes adjacent queries for

Context	**No. of queries**	**Accuracy**	F_1
Backward - forward	2	0.935	0.882
	4	0.920	0.853
	6	0.927	0.868
	8	0.919	0.852
	10	0.917	0.844
Backward only	1	0.933	0.877
	2	0.934	0.878
	3	0.927	0.863
	4	0.928	0.865
	5	0.924	0.859
No context	0	**0.937**	**0.884**

Table 3.4: Model metrics using additional adjacent queries in the search log for providing context. We consider several numbers of adjacent queries in the WSMC12 dataset, selecting both backward and forward contexts or backward context only.

context, we do not see any improvement in the accuracy of the segmentation architecture. Although the addition of adjacent query vectors still exceeded other baseline methods, overall, none of the results improved the performance of the query pair without context (Table 3.3). Such a result is essential because the average amount of adjacent queries per search task is around three. Similarly, simple information needs like fact-finding are usually solved with only one query (Hearst, 2009). Furthermore, forward context can be a critical drawback, especially when supporting user information needs. Applications like query suggestion must respond on the fly. They cannot wait for future queries to provide context to the model.

The same pattern was observed when we examined only backward queries for context (Table 3.4), a strategy similar to the approach in Bestlink SVM (Wang et al., 2013a), where only previous queries in the sequential log were considered. In all tests, the segmentation model gave the best performance when the input had the query pair and the time span between them.

3.4.2.3 Fine-tuning with a smaller dataset

Finally, we wanted to know the impact of pre-training the segmentation architecture before training the model with a small dataset. Scarce data poses challenges to deep learning architectures, and fine-tuning is one of the alternatives to deal with small datasets (Angermueller et al., 2016; Lin et al., 2015). Table 3.5 presents the results for the fine-tuning experiments. When performing 10-fold cross-validation on the CSTE dataset, without doing fine-tuning, we got an accuracy of 0.751. After pre-training with the WSMC12 dataset, we performed 10-fold cross-validation on the CSTE dataset, obtaining an accuracy of 0.803. Thus, the BiRNN architecture configuration with the time span at the input and GRU recurrent units gave the best performance when fine-tuning. Overall, it represents the best performance with the smaller CSTE dataset when we consider all the methods tested. The results mentioned above highlight the ability of the model to perform well when processing datasets with a limited number of samples, a common scenario because of the difficulty for obtaining large search query logs and the cost associated with labeling them (Wang et al., 2013a).

	Segmentation model	**Accuracy**	F_1
No pre-training	BiRNN LSTM - time at AL	0.784	0.651
	BiRNN GRU - time at AL	0.789	0.648
	BiRNN LSTM	0.788	0.663
	BiRNN GRU	0.751	0.604
Pre-training	BiRNN LSTM - time at AL	0.787	0.649
	BiRNN GRU - time at AL	0.782	0.644
	BiRNN LSTM	0.769	0.644
	BiRNN GRU	**0.803**	**0.665**

Table 3.5: Fine-tuning the recurrent architecture to realize segmentation on the CSTE dataset. The model is pre-trained using the WSMC12 dataset.

3.5 Summary

In this chapter, we proposed a bidirectional RNN architecture with an attention mechanism for segmenting search query logs by identifying task boundaries on them. Our proposed segmentation model processes query pairs in search logs to determine if a boundary is present or not. Experimental results showcase the improvement of the proposed architecture over the baseline method, outperforming the other approaches used for comparison. Once trained, the recurrent model is also several times faster than the heuristics-based baseline. Furthermore, there is no need for clicked URLs, character-level representations, or additional queries surrounding the query pair to provide context to the model. This result is especially relevant given the mean number of queries per task in datasets from widely used search engines and the fact that information needs like fact-finding and other simple tasks are usually solved with only one query. Even when testing with additional context to the query pair, segmentation architecture results continue to be above all the other approaches. Also, the proposed segmentation model performs well when fine-tuning with a smaller query log dataset. Fine-tuning performance is useful given the scarcity of publicly available labeled collections from search engines and the cost of labeling large search logs.

Chapter 4
Extracting Search Tasks from Query Logs Using a Recurrent Deep Clustering Architecture

Tout ce qu'un homme peut
imaginer, d'autres hommes
peuvent le rendre réel.

Jules Verne

The neural model in Chapter 3 learns to detect search task boundaries from user interactions. However, it is a supervised method, requiring labeled datasets for training the model. Likewise, task boundaries are not enough to extract search tasks from user interactions. The steps users perform to carry out search tasks are not necessarily sequential. Those steps can appear interleaved in the search query log, reflecting the fact that users multitask when accessing search systems. In several search query datasets (Section 2.1), it is possible to observe that users can switch to other search tasks, and after a certain period, come back to continue the previous search task. In addition, different users can run queries to carry out the same information need.

Consequently, in this chapter, we extend the neural architecture to propose a model that can extract search tasks from user interactions. Most existing search task extraction methods use graph-based or non-parametric models, which grow as the query log size increases (Section 2.4). Deep clustering methods offer a parametric alternative, but most deep clustering architectures fail to exploit recurrent neural networks for learning text data representations. Therefore, we propose a recurrent deep clustering model for extracting search tasks from query logs. The proposed architecture leverages self-training and dual recurrent encoders for learning suitable latent representations of user queries, outperforming previous deep clustering methods. It is also a parametric approach, which offers the possibility of having a fixed-sized architecture for analyzing increasingly large search query logs.

4.1 Introduction

Users carry out their search tasks running groups of related queries in available search systems, fulfilling a wide range of information needs and desires (Hearst, 2009). Query logs record the queries that users submit to search engines. Therefore, proper extraction of search tasks from query logs helps to support users while they fulfill their information needs, facilitating multiple goals like query term prediction, query recommendation, advertisement, results ranking depending on the search task, query-task mapping, and prediction of user satisfaction based on search tasks (Du et al., 2018; Mehrotra and Yilmaz, 2017; Sen et al., 2018; Völske et al., 2019).

Along with the search queries that users submit, query logs also contain timestamps and other user-related information. Initially, query logs were segmented using the time between query timestamps to establish a session boundary and delimiting search tasks. If the time between a pair of subsequent queries was above a certain threshold in minutes, a boundary was established, signaling the end of a search task (Lucchese et al., 2013). However, according to multiple analyses of search query logs, users tend to interleave search tasks in a single time session. Also, some tasks are performed during multiple time sessions (Lucchese et al., 2013; Mehrotra and Yilmaz, 2017; Sen et al., 2018). Hence, clustering models have been utilized to extract search tasks by grouping semantically related queries.

Recent models for search task extraction rely on graph-based methods or non-parametric approaches (Du et al., 2018; Lucchese et al., 2013; Mehrotra et al., 2016; Mehrotra and Yilmaz, 2017; Sen et al., 2018), which grow as search query logs increase in size, making them more computationally expensive as the number of queries increases. By contrast, deep clustering methods (Aljalbout et al., 2018; Min et al., 2018) offer a parametric alternative to learn latent representations of query log entries and simultaneously cluster them into interrelated groups of search queries.

Most existing deep clustering approaches do not exploit the modeling power of recurrent neural networks (RNNs), which are widely used for natural language processing (NLP) and sequential data processing (Graves, 2012; Lipton et al., 2015; Mitchell, 2019). Therefore, we propose a recurrent deep clustering (RDC) model to extract search tasks from query logs. RDC leverages the modeling power of recurrent neural networks in a dual encoder configuration, along with self-training, to learn a suitable latent space of user queries and simultaneously cluster them in groups of search tasks.

4.2 Related limitations

The need for large query log datasets that are cleaned and labeled by humans represents a challenge for supervised task extraction models (Du et al., 2018;

Mitchell, 2019; Wang et al., 2013a). As unsupervised learning approaches do not rely on labels (Murphy, 2012), they could represent a better alternative for search task extraction. Clustering is an unsupervised learning approach that groups related items using abstract similarities or learns new categories by analogy to existing ones (Mitchell, 2019).

Several clustering methods has been proposed proposed to extract search tasks from query logs (Section 2.4). Nonetheless, graph-based and non-parametric models grow as the size of datasets increases (Murphy, 2012), becoming more computationally expensive. For example, the number of leaves in BRTs (Mehrotra and Yilmaz, 2017) is directly related to the number of queries in the search query log; for graph-based methods, every entry in the search query log ends up being a node in the underlying graph. Likewise, the representation of the data is crucial for the subsequent results of clustering methods. High dimensional data tends to affect clustering methods because distances in high dimensional spaces are less effective. Dimensionality reduction methods have been widely used, including linear methods, non-linear methods, and spectral methods. Nevertheless, the latent representation obtained from dimensionality reduction can affect clustering performance; thus, deep neural networks are a viable alternative to compute latent representations (Aljalbout et al., 2018; Min et al., 2018; Murphy, 2012) for input data, without performing dimensionality reduction as a preprocessing step.

Deep neural networks can be used to simultaneously learn latent representations and cluster data, using a method commonly known as deep clustering (Aljalbout et al., 2018; Min et al., 2018). Also, in contrast with graph-based methods and non-parametric approaches, deep clustering models do not grow with the size of the search query log (Murphy, 2012). Deep clustering appeared initially in acoustic separation and then spread to other areas of research (Min et al., 2018). Before deep clustering appeared, research focused on data representation and clustering methods independently. However, learning latent representations is at the heart of deep clustering.

Models in deep clustering rely on several neural network architectures, including autoencoders, variational autoencoders (VAEs), feedforward neural networks, convolutional neural networks, deep belief networks, and generative adversarial networks (GANs) (Aljalbout et al., 2018; Min et al., 2018). All architectural variations are trained to learn cluster friendly representations, combining representation learning and clustering. Deep neural networks are trained to minimize the clustering loss, optimizing the network weights for improving the predicted labels of input samples. Both GAN and VAE based architectures are generative. They do not only learn to cluster inputs; they are also able to generate samples from the clustering categories (Aljalbout et al., 2018; Min et al., 2018).

Existing deep clustering models fail to exploit RNNs for learning latent representations of text data samples. Text data naturally fits the sequential modeling power of recurrent neural networks (Graves, 2012). Because of this, RNNs have been widely used for processing text data in NLP, generating state-of-the-art results in multiple applications (Graves, 2012; Mitchell, 2019; Zhang et al., 2019). Our proposed architecture differs from prior deep clustering methods (Aljalbout et al., 2018; Min et al., 2018) by using RNNs in a dual encoder configuration (Yang et al., 2019) to simultaneously learn latent representations of user queries and cluster them in groups of search tasks. Also, in contrast with other approaches (Du et al., 2018; Lucchese et al., 2013; Mehrotra and Yilmaz, 2017; Sen et al., 2018), it provides a parametric model, which preserves its size despite the query log length.

4.3 Search task extraction

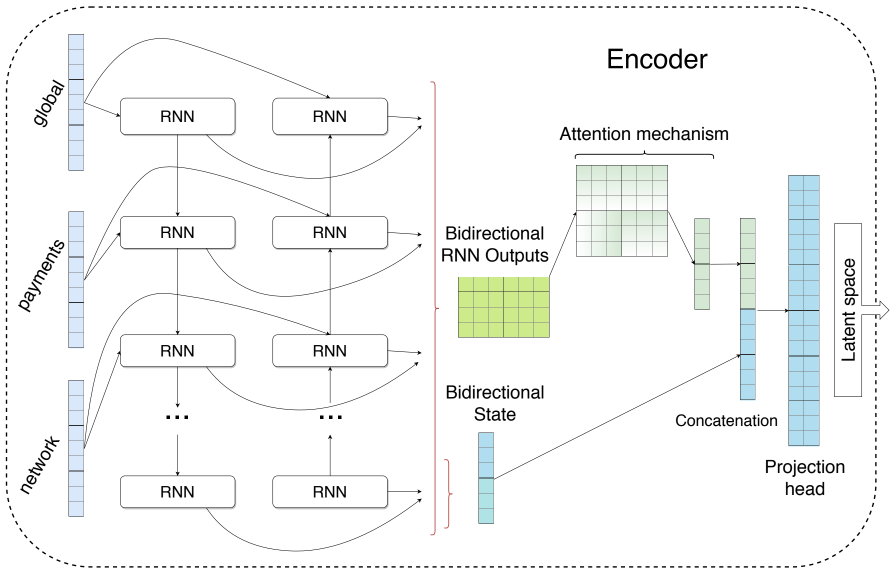


Fig. 4.1: Recurrent neural network encoder for learning latent representations of queries. The encoder comprises a bidirectional recurrent layer, an attention mechanism, and a projection head.

The proposed RDC model has an RNN encoder as the central component of its architecture. The architecture uses a dual encoder setup (Yang et al., 2019) (Section 2.2), a widely used configuration in representation learning, neural machine translation, and other NLP applications (Chen et al., 2020; Yang et al., 2019, 2020). The recurrent encoder comprises a bidirectional recurrent layer, an attention mechanism, and a projection head (Chen et al., 2020). Input queries comprise a list of word embeddings $q_i = [w_1, w_2, w_3, ...]$.

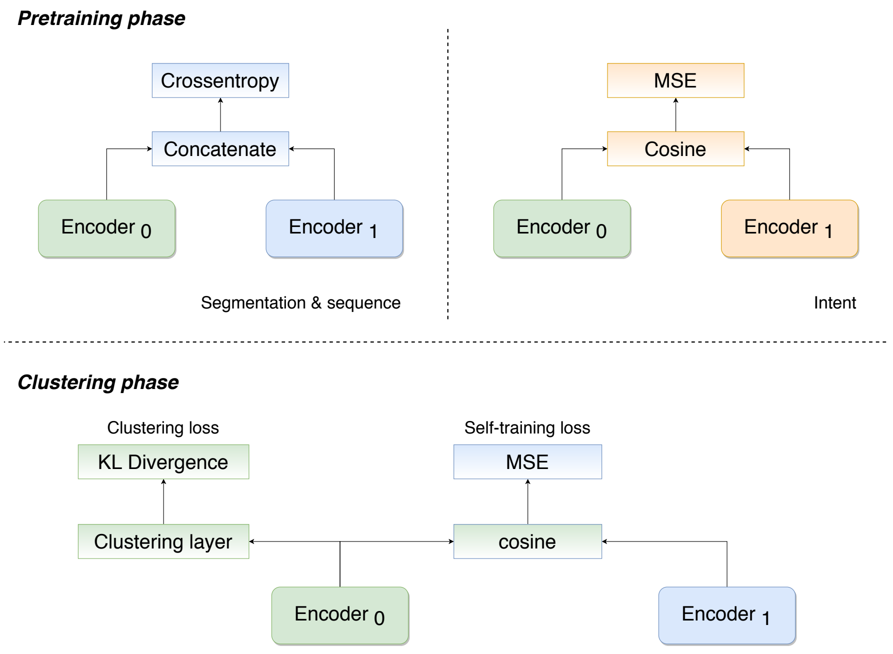


Fig. 4.2: Pretraining and clustering phases for search task extraction using a dual encoder configuration.

To form the input query's latent representation, we concatenate the output of the attention mechanism and the hidden state of the bidirectional recurrent layer, passing the concatenated tensor through a projection head (Figure 4.1). Regarding recurrent unit types for the encoder, we consider both Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) (Chung et al., 2014).

Two phases optimize RDC in tandem: a pretraining phase and a clustering phase (Guo et al., 2017; Min et al., 2018; Xie et al., 2016). Each phase has its loss; thus, we adapt the architecture of the model depending on the loss that we are optimizing (Figure 4.2).

4.3.1 Pretraining phase

Deep clustering methods tend to pretrain neural network layers before the clustering phase, which allows the initialization of the latent representation for input samples (Aljalbout et al., 2018; Min et al., 2018). During the pre-training phase, we optimize the encoder with a supervised objective. We use the dual encoder configuration (Yang et al., 2019) to pretrain the recurrent encoders according to the following objectives:

- *Segmentation.* In this supervised pretraining objective, the recurrent encoders are trained using the search task segmentation approach (Chapter 3). This pretraining objective determines if two adjacent queries in a chronologically ordered query log are part of the same search task or not. The expected output of this objective is binary, and we use cross-entropy to compute the pretraining loss $\mathcal{L}_P$.

- *Sequence.* The sequence pretraining objective determines if a pair of queries appear adjacent in a chronologically ordered query log or not. Similar to the segmentation objective, the expected output of this objective is also binary, and we use crossentropy to calculate the pretraining loss $\mathcal{L}_P$.

- *Intent.* For the intent pretraining objective, queries representing the user's intent for the same search task are close in the latent representation space (Zhang et al., 2019). Therefore, we compute the cosine proximity between encoder outputs and use the Mean Squared Error (MSE) (Tan et al., 2018) between predicted cosine proximity and expected cosine proximity to calculate the pretraining loss $\mathcal{L}_P$. The expected cosine proximity is set to one for pairs of queries pertaining to the same search task, zero otherwise.

4.3.2 Clustering phase

Once the foregoing objectives have been used to pretrain the recurrent encoders, we discard the layers on top of the encoders that we used during pretraining, changing the pretraining objective to the clustering objective. The objective loss for the clustering phase $\mathcal{L}_O$ comprises the clustering loss $\mathcal{L}_C$ and the self-training loss $\mathcal{L}_S$ (Guo et al., 2017; Luo et al., 2017).

For the clustering loss, following previous work (Aljalbout et al., 2018; Min et al., 2018; Xie et al., 2016), we connect the pretrained encoders' output to a clustering layer, where the Student's t-distribution provides a kernel to compute the soft assignments for the data points representing the user queries. The Student's t-distribution is a heavy-tailed distribution that helps to preserve distances of data points when mapping from a high dimensional to a low dimensional space. When mapping data points, the distances in the high dimensional space require larger distances in the low dimensional space to remain equivalent. Hence, the natural gaps existing between clusters in a high dimensional space can easily disappear when mapping to a lower dimensional space. One way to deal with this phenomenon is to use a heavy-tailed distribution to model distances in the low dimensional space. Compared to a

Gaussian distribution, the Student's t distribution has much heavier tails. For that reason, it is used in the t-Distributed Stochastic Neighbor Embedding (t-NSE) (Van der Maaten and Hinton, 2008), a widely popular technique for visualizing high dimensional data. Thus, by using the Student's t distribution in the low dimensional space for the clustering layer, it is possible to preserve the natural gaps that exist in the high dimensional space (Xie et al., 2016; Van der Maaten and Hinton, 2008).

Additionally, the soft assignments should match an auxiliary target distribution, which is designed to emphasize high confidence data point assignments, strengthen predictions, and normalize the loss contribution from each cluster by using the soft cluster frequencies. Therefore, we compute the clustering loss by calculating the Kullback–Leibler (KL) divergence between the Student's t-distribution and the auxiliary target distribution. Formally, given a query q_i and initial cluster centroids μ_j, the clustering loss $\mathcal{L}_C$ is calculated as follows (Guo et al., 2017; Xie et al., 2016):

$$z_i = encoder_0(q_i) \tag{4.1}$$

$$s_{ij} = \frac{(1 + \|z_i - \mu_j\|^2)^{-1}}{\sum\limits_{j'} (1 + \|z_i - \mu'_j\|^2)^{-1}} \tag{4.2}$$

$$f_j = \sum_i s_{ij} \tag{4.3}$$

$$p_{ij} = \frac{s_{ij}^2 / f_j}{\sum\limits_{j'} s_{ij'}^2 / f_{j'}} \tag{4.4}$$

$$\mathcal{L}_C = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{s_{ij}} \tag{4.5}$$

where p_{ij} is the auxiliary target distribution and f_j are the soft cluster frequencies. K-means generates the initial cluster centroids μ_j from the pre-trained encoder representation. The Student's t-distribution in equation 4.2 has one degree of freedom (Xie et al., 2016).

For the self-training loss, we use the dual encoder configuration along with back translation (Edunov et al., 2018), a self-training technique for unsupervised data augmentation (Xie et al., 2020) that preserves the semantics of the query encodings. Adding noise to query encodings can be ineffective for creating augmented samples because the resulting samples hardly match variations from real case scenarios. Hence, to create realistic augmented samples, strong data augmentation methods focus on creating modifications that match real variations. For instance, in computer vision, it is common to use cropping, rotation, or scaling to create augmented samples for images (Chen et al., 2020; Zoph et al., 2020). As queries are short texts, we use back translation to create realistic augmented samples for entries in the search query log (Figure 4.3).

Back translation creates paraphrases for a search query while preserving the semantics of the original query (Edunov et al., 2018). Formally, given a query q_i, with augmented sampled b_i, and because the semantics from back translation remains the same, target cosine proximity $t_i = 1.0$, the self-training loss $\mathcal{L}_S$ is calculated as follows (Guo et al., 2017; Min et al., 2018; Xie et al., 2020):

$$u_i = encoder_0(q_i) \tag{4.6}$$

$$v_i = encoder_1(b_i) \tag{4.7}$$

$$p_i = \frac{u_i v_i}{|u_i|\,|v_i|} \tag{4.8}$$

$$\mathcal{L}_S = \sum_i (p_i - t_i)^2 \tag{4.9}$$

During the clustering phase, both the cluster centroids and the dual encoder weights are updated by optimizing the objective loss $\mathcal{L}_O$ (Aljalbout et al., 2018; Guo et al., 2017; Min et al., 2018):

$$\mathcal{L}_O = \mathcal{L}_S + \gamma \mathcal{L}_C \tag{4.10}$$

where γ is a constant. The range of γ is $0.0 < \gamma < 1.0$ to help the model preserve the semantic space of the query encodings with the optimization of $\mathcal{L}_S$, while simultaneously optimizing $\mathcal{L}_C$ to improve the clustering performance.

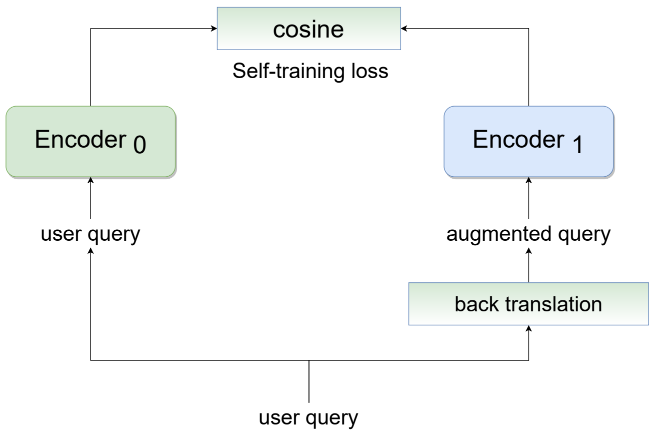


Fig. 4.3: Calculation of self-training loss using the dual encoder configuration, together with back translation to produce realistic augmented queries.

4.4 Experimental setup

Metrics to evaluate the performance of the models include the unsupervised accuracy (ACC), the Normalized Mutual Information (NMI), and the Adjusted Rand Index (ARI) (Section 2.2.3). The Student's paired t-test provides the test for statistical significance. To evaluate the effectiveness of RDC, we compare its performance with the following methods:

- Deep Embedded Clustering (DEC) combines feature extraction with autoencoders and clustering. It learns clustering centers by first pretraining the autoencoder on the input dataset to learn a latent representation. Then, DEC discards the decoder part of the autoencoder and uses the encoder to calculate input representations. DEC uses K-means to initialize cluster centroids, then, it minimizes the clustering loss by minimizing the KL divergence (Equation 4.5) (Chang et al., 2017; Xie et al., 2016).

- Improved Deep Embedded Clustering (IDEC) extends DEC by including the decoder part of the autoencoder during the clustering. Doing so aims to preserve the original structure of the input data in the latent representation space. To include the encoder, IDEC uses a loss to simultaneously optimize the clustering on the encoder output and the representational accuracy of the decoder output (Guo et al., 2017).

- Point Symmetry-based Deep Clustering (SymDEC) replaces the euclidean distance that DEC uses for computing the clustering loss with the point symmetry-based distance, improving the results when clustering datasets with symmetrical input samples (Moreno, 2018).

- Deep adaptive clustering (DAC) joins feature extraction and clustering into a single neural method. To extract features, DAC relies on a deep convolutional neural network and adds a constraint, so that resulting labels converge to a one-hot encoding. The constraint assumes that a pair of input samples either pertain to the same cluster or pertain to a different cluster. Input sample similarities are unknown beforehand; thus, an adaptive approach inspired by curriculum learning (Bengio et al., 2009) is proposed. First, only pairs of input samples with similarities above or under a threshold are considered. With those pairs of images, the weights of the convolutional network are updated using back-propagation. As the training advances and the model improves, more pairs meet the threshold criteria. When the model converges, all pairs of input samples are part of the loss computation, and the loss stabilizes. Once the loss stabilizes, it selects the label with the highest value inside the one-hot vector to determine the cluster of the input sample (Chang et al., 2017).

- Chimera network (Luo et al., 2017; Wang et al., 2018) uses stacked layers of bidirectional LSTMs for audio separation models. This stacked recurrent model can handle problems like speaker-independent multi-speaker speech separation and music source separation. The Chimera architecture comprises four bidirectional layers, a dense layer to compute the vectors in the latent space, and two heads for multi-task learning: one head for unsupervised source separation and other head for supervised time-frequency mask inference. We replace the multi-task learning heads with the clustering layer in Section 4.3.2 to adapt the architecture for search task extraction from query logs.

For reference, we also include results for k-means (Tan et al., 2018), Density-based spatial clustering of applications with noise (DBSCAN) (Ester et al., 1996), and Hierarchical Agglomerative Clustering (HAC) (Murtagh and Legendre, 2014; Ward Jr, 1963). Scikit-learn[1] with default parameters provides the implementation for k-means, DBSCAN, and HAC. We also use publicly available implementations for DEC[2], IDEC[2], SymDEC[2], DAC[3], and Chimera[4] with the best performing hyperparameters reported for each method.

Two datasets are considered for evaluating RDC performance: the CSTE and the TRECQTM datasets. For pretraining, two datasets are considered as well: Sequence and Segmentation pretraining objectives use the WSMC12 dataset, which has timestamps so that we can guarantee a chronologically ordered query log. The Intent pretraining objective uses the WHQTM dataset (Section 2.1).

The GloVe publicly available pre-trained word vectors[5] provide the representation for the search queries (Pennington et al., 2014). We use the same query representation for all the methods under testing. To train the RDC model, we use the Adam optimizer (Kingma and Ba, 2014). The learning rate is set to 10^{-4}, batch size to 256, and dropout to 0.3. The bidirectional layer contains 32 recurrent units, and the projection head has two feedforward layers, one with 512 units and the other with 256 units. Using the Google Cloud Translation API[6], we perform the back translation augmentation for the self-training loss (Equation 4.9). Back translation is realized offline for practical purposes, using English (en) - French (fr) (Xie et al., 2020) to create the augmented samples.

[1] `https://scikit-learn.org`

[2] `https://github.com/XifengGuo/DEC-DA`

[3] `https://github.com/HongtaoYang/DAC-tensorflow`

[4] `https://github.com/leichtrhino/ChimeraNet`

[5] `http://nlp.stanford.edu/data/glove.42B.300d.zip`

[6] `https://cloud.google.com/translate`

Dataset	Method	Pretraining	ACC	NMI	ARI
CSTE	k-means	None	0.395	0.670	0.231
	DBSCAN	None	0.199	0.343	0.027
	HAC	None	0.407	0.719	0.310
	DEC	Autoencoder	0.362	0.684	0.345
	IDEC	Autoencoder	0.347	0.681	0.348
	SymDEC	Autoencoder	0.337	0.652	0.325
	DAC	None	0.318	0.644	0.344
	Chimera	None	0.387	0.707	0.339
	RDC	Sequence	**0.420**	**0.735**	**0.355**
	RDC	Segmentation	0.408	0.730	0.354
	RDC	Intent	0.331	0.641	0.334
	RDC	None	0.415	0.734	0.355
TRECQTM	k-means	None	0.219	0.535	0.050
	DBSCAN	None	0.026	0.105	0.001
	HAC	None	0.276	**0.613**	0.086
	DEC	Autoencoder	0.097	0.419	0.019
	IDEC	Autoencoder	0.097	0.418	0.018
	SymDEC	Autoencoder	0.104	0.396	0.022
	DAC	None	0.095	0.368	0.025
	Chimera	None	0.214	0.523	0.061
	RDC	Sequence	**0.285**	0.594	0.094
	RDC	Segmentation	0.246	0.566	0.080
	RDC	Intent	0.187	0.508	0.055
	RDC	None	0.284	0.590	**0.095**

Table 4.1: Clustering performance for CSTE and TRECQTM datasets, including RDC and other methods used for comparison. Differences in RDC results against all baseline methods have $p \leq 0.05$ for the Student's t-test.

4.5 Results and discussion

Results for RDC with several pretraining configurations appear in Table 4.1. RDC outperforms all the other deep clustering methods used for comparison, for both the CSTE and the TRECQTM datasets. RDC also outperforms reference methods like k-means and DBSCAN. When comparing clustering performance against HAC, we find that RDC outperforms HAC when extracting short-lived search tasks, while in long search tasks, it improves over HAC in two out of three metrics ($p \leq 0.05$). The CSTE dataset has mostly short-lived search tasks because the average number of user queries per task is 3.2, while the TRECQTM dataset has an average of 28.2 user queries per task, reflecting behaviors like exploration, specification, or paraphrasing that users undertake in long search tasks (Zhang et al., 2019). These results are essential because short-lived search tasks, including fact-finding, browsing, or transactions, can account for up to 85% of all the entries in a search query log (Hearst, 2009).

When comparing RDC with autoencoder-based models, such as DEC, IDEC, and SymDEC, the results are higher in all the metrics used for assessing clustering performance; we observe the same behavior when considering DAC, which uses convolutional neural networks. This outperformance reflects the advantage of using the modeling power of recurrent neural networks for learning representations of search queries. Chimera, a stacked recurrent architecture, also outperforms deep clustering models based on autoencoders and convolutional neural networks. However, RDC has a better clustering performance than Chimera in the three metrics used for comparison. Similarly, RDC has a more straightforward configuration than Chimera because RDC only uses two bidirectional recurrent layers for the dual encoder setup, while Chimera uses a stack of four bidirectional recurrent layers.

Self-training with back translation for queries renders pretraining effects negligible. Indeed, back translation using English (en) - French (fr) is a strong data augmentation technique. It augments data samples while preserving the semantics of the original queries. For instance, “effects of tide on columbia river”gets translated to “effets de la marée sur la rivière Columbia”, and then back translated to “tidal effects on the columbia river”; “farm houses for rent in broom county”gets translated to “Maisons de ferme à louer dans le comté de broome”, and then back translated to “farms for lease in broom county”. Sometimes back translation corrects spelling, for instance “the cost of haveing a horse in new york”gets translated to “le coût d’avoir un cheval à new york”, and then back translated to “the cost of having a horse in new york”, but in general, the semantics remain the same, so the query encoding space is preserved during the clustering phase by minimizing the self-training loss.

Consequently, although the best pretraining scheme for the RDC models is the Sequence objective, surpassing the results of both Segmentation and Intent objectives, it represents no change when compared against RDC with no pretraining. For the CSTE dataset, accuracy is only 0.5% higher, and NMI is only 0.1% higher ($p = 0.8$); ARI has no change at all. We observe a similar behavior with the QTMT dataset. In some cases, pretraining can even end up hurting performance, as we can see with the Intent pretraining objective. Preceding results are in agreement with previous work about the effect of pretraining neural architectures (Zoph et al., 2020), where self-training with strong data augmentation diminishes the effect of pretraining, making it negligible. Therefore, it is possible to discard neural network pretraining,

Dataset	Cell	Pretraining	ACC	NMI	ARI
CSTE	LSTM	Sequence	0.410	0.730	0.354
		Segmentation	0.399	0.718	0.352
		Intent	0.320	0.632	0.334
		None	0.409	0.729	0.354
	GRU	Sequence	0.420	0.735	0.355
		Segmentation	0.408	0.730	0.354
		Intent	0.331	0.641	0.334
		None	0.415	0.734	0.355
TRECQTM	LSTM	Sequence	0.278	0.592	0.096
		Segmentation	0.217	0.544	0.070
		Intent	0.160	0.483	0.044
		None	0.268	0.586	0.092
	GRU	Sequence	0.285	0.594	0.094
		Segmentation	0.246	0.566	0.080
		Intent	0.187	0.508	0.055
		None	0.284	0.590	0.095

Table 4.2: Comparison between LSTM and GRU cells for the recurrent layers of RDC. Results for the CSTE and TRECQTM datasets include all the pretraining alternatives.

an essential result because pretraining needs labeled datasets, which can be challenging to create (Wang et al., 2013a), while self-training with back translation is unsupervised.

Regarding recurrent units, the decision of which to choose depends on the task and the dataset (Chung et al., 2014); therefore, we analyze the RDC model with both GRU and LSTM cells (Table 4.2). Replacing the LSTM cells with GRUs generates a slight decrease in model performance for CSTE and TRECQTM datasets. The biggest difference happens with the TRECQTM dataset, using intent pretraining, where changing GRUs to LSTMs makes accuracy decrease 2.7%, NMI 2.5%, and ARI 1.1% ($p \leq 0.05$). These changes imply that less computationally expensive GRUs are a better choice for the RDC architecture than LSTMs, although changes observed for the metrics are low, especially with the sequence or no pretraining configurations, which are the best performing setups for the RDC model.

4.6 Summary

This chapter presented RDC, a recurrent deep clustering method for extracting search tasks from query logs. The proposed method leverages self-training and dual recurrent encoders to find latent representations for user queries, clustering them into search task groups. Experimental results show the proposed clustering method outperforms prior deep embedding clustering architectures in all the metrics used for testing. Also, RDC offers a parametric architecture for search task extraction, which preserves its size despite changes in the query log size. This size preservation represents an advantage compared to non-parametric methods and graph-based models that grow with the query log size, making them more computationally expensive as the number of queries in the search log grows.

Chapter 5
A multilingual approach for unsupervised search task identification

The best way to get a good idea is to get a lot of ideas.

Linus Pauling

Parametric models like RDC (Chapter 4) tend to have lower modeling results than non-parametric alternatives. When comparing parametric and non-parametric machine learning models, there is a trade-off between model size and accuracy (Murphy, 2012). Parametric models remain constant regardless of the size of the input dataset, but they tend to be less accurate. On the other hand, non-parametric models grow as the size of the input dataset grows. However, they tend to be more accurate. Additionally, the use of GloVe word vectors for query representation limits RDC to English queries only. On top of that, we need to estimate the number of search tasks beforehand, so that we can initialize the clustering layer.

Therefore, in this chapter, we propose an unsupervised, multilingual model for search task identification. Search system logs contain queries in multiple languages, but most existing methods for search task identification are not multilingual. Some methods rely on search context for custom embeddings or external indexed collections that support a single language, making it challenging to support the multiple languages of queries run in search systems. Other methods depend on supervised components and user identifiers to model search tasks. The supervised components require labeled collections, which are challenging and costly to get in multiple languages (Section 2.4). Also, the need for user identifiers renders these methods unfeasible in user agnostic scenarios. Hence, the proposed approach in this chapter is user agnostic, enabling its use in both user-independent and personalized scenarios. Furthermore, the multilingual query representation enables us to address the existing trade-off when mapping new queries to the identified search tasks.

5.1 Introduction

To support users during the different steps of the information seeking process, it is crucial to correctly group queries in search logs according to their search tasks. Mining query logs from search engines enable the automatic modeling of search tasks. Precise modeling of search tasks is required for user assisting applications like query term prediction, query recommendations, user modeling based on tasks, personalization in e-commerce, and results ranking (Hearst, 2009; Mehrotra and Yilmaz, 2017; Völske et al., 2019), which enhance search system support for helping users to fulfill their information needs.

Most unsupervised search task identification methods rely on custom training of word embeddings using search collections or external indexed collections to provide semantic similarities for search queries (Lucchese et al., 2013; Mehrotra and Yilmaz, 2017; Sen et al., 2018). Unfortunately, these methods cannot support user queries in multiple languages. Other supervised approaches for search task identification require collections of manually labeled data to train the models (Wang et al., 2013a; Du et al., 2018), which are challenging to create because of the cost of manual labeling (Wang et al., 2013a) and the long-tail nature of search queries (Zhang et al., 2019).

We propose a multilingual method for unsupervised search task identification. The proposed approach combines graph clustering methods (Lucchese et al., 2013; Sen et al., 2018) with recent general language models (Cer et al., 2018; Yang et al., 2020) for obtaining query representations in a multilingual semantic vector space. The proposed search task identification approach is independent of user identifiers, enabling the modeling of search tasks in user agnostic or personalized applications. We also address the existing trade-off between accuracy and query time (Völske et al., 2019) that arises when mapping new incoming queries to identified search tasks.

5.2 Related limitations

Search logs provide fine-grained details at the query level, enabling the characterization and classification of individual queries according to the information need they are related to (Hearst, 2009). Hence, it is possible to cluster related queries in the search log to model the tasks that users perform on search engines to fulfill their information needs. Existing methods (Section 2.4) like Bestlink SVM (Wang et al., 2013a) and CA-LSTM (Du et al., 2018) require a supervised component to perform task identification. CA-LSTM also employs user identifiers to determine the adjacent queries needed to provide context to the recurrent architecture. However, user agnostic scenarios do not have user identifiers available. Also, the context required from adjacent queries could not be available, especially when dealing with simple search tasks like fact-finding, which tend to have a single query (Hearst, 2009). BRTs (Mehrotra and Yilmaz, 2017) and LDA-Hawkes (Li et al., 2014a) depend on user identifiers and time sessions to compute the user/time affinity. Similarly, some graph clustering methods (Lucchese et al., 2013) use time and user sessions to group the queries before the clustering. Heuristics task identification methods (Gayo-Avello, 2009; Hagen et al., 2013) rely on user information and timestamps to identify the search tasks, along with several manually set thresholds. Furthermore, relying on time sessions to identify tasks (Gayo-Avello, 2009; Hagen et al., 2013; Mehrotra and Yilmaz, 2017) could be misleading. According to multiple analyses of search query logs (Lucchese et al., 2013; Mehrotra and Yilmaz, 2017; Sen et al., 2018), users tend to multitask during single time sessions and some complex tasks extend during multiple sessions.

Likewise, both QRY-VEC (Sen et al., 2018) and BRTs (Mehrotra and Yilmaz, 2017) use a custom training word embedding model. The custom training performed using the tempo-lexical context (Sen et al., 2018) can avoid topic shifting (Rekabsaz et al., 2017), but unfortunately, there are not enough labeled collections to train multilingual word embeddings using such context. Also, the use of an external index (Lucchese et al., 2013; Sen et al., 2018) requires time-consuming index access at the retrieval model (Hagen et al., 2013) and similar to the issue present in custom training word embeddings, there are not enough corpus for a multilingual clustering of queries. Furthermore, several methods use cosine similarity between query vectors to prune edges in the clustering graph (Sen et al., 2018) or compute the query affinities (Hagen et al., 2013; Mehrotra and Yilmaz, 2017). However, the angular similarity has a better performance than the cosine similarity in semantic textual similarity (STS) between sentence pairs (Cer et al., 2018).

5.3 Task identification approach

The proposed unsupervised approach uses a multilingual query representation and a graph based clustering method to group queries related to the same search task. In contrast with previous methods (Du et al., 2018; Hagen et al., 2013; Lucchese et al., 2013; Mehrotra and Yilmaz, 2017; Sen et al., 2018; Wang et al., 2013a), it supports queries in multiple languages. Also, it does not utilize user identifiers (Du et al., 2018; Hagen et al., 2013; Mehrotra and Yilmaz, 2017) and has no supervised components (Du et al., 2018; Wang et al., 2013a). In this section, we cover the multilingual query representation and explain the graph based clustering method.

5.3.1 Multilingual query representation

Pretrained word embeddings have been released in several languages (Grave et al., 2018; Mikolov et al., 2013). However, using pre-trained word vectors can generate topic shifting (Rekabsaz et al., 2017; Zhang et al., 2019) and requires an additional phase to detect the language of the query to correctly select the pre-trained model to compute the multilingual query vector. Instead, Multilingual Universal Sentence Encoder (MUSE) (Yang et al., 2020) provides universal representations for sentence embeddings suited to information retrieval tasks (Zhang et al., 2019).

MUSE has models based on transformers (Vaswani et al., 2017) or convolutional neural networks (CNNs). We use the transformer-based model. It is more computationally expensive than the CNN based model; however, it is more accurate on several tasks, including sentence retrieval, bitext retrieval, and retrieval question answering. The transformer-based model relies on the encoder part of the transformer architecture. It takes into account context-aware embeddings and averages together the results from the encoder to produce one vector per sentence. Training is based on question-answer pairs, translation pairs, and the Stanford Natural Language Inference (SNLI) corpus. Furthermore, MUSE utilizes SentencePiece, a language-independent subword tokenizer, to process the input query text. SentencePiece covers above 99% of possible tokens in all languages. Likewise, MUSE supports queries in sixteen languages: Arabic (ar), Chinese PRC (zh), Chinese Taiwan (zh-tw), Dutch(nl), English(en), German (de), French (fr), Italian (it), Portuguese (pt), Spanish (es), Japanese (ja), Korean (ko), Russian (ru), Polish (pl), Thai (th), and Turkish (tr).

5.3.2 Graph Based Clustering

Existing clustering methods for search task identification rely on lexical similarities (Lucchese et al., 2013) or cosine distances between word embeddings (Sen et al., 2018; Mehrotra and Yilmaz, 2017). However, the angular similarity (Cer et al., 2018) has been used in recent research (Chidambaram et al., 2018; Yang et al., 2020) to better discriminate text representations in natural language processing. In particular, the angular similarity has been found to perform better in STS between sentence pairs than the cosine similarity (Cer et al., 2018); thus, we use it to compute the similarity between query pairs. Given two queries q_i, q_j, with multilingual vector representations v_i, v_j, the angular similarity S_{ang} is defined as follows (Cer et al., 2018):

$$S_{ang}(v_i, v_j) = -\arccos\left(\frac{v_i v_j}{|v_i|\,|v_j|}\right) \tag{5.1}$$

Algorithm 1 MGBC algorithm

```
Input: Query log $Q$ Output: Task labels $L$
  $V \leftarrow \{\}, E \leftarrow \{\}, G(V,E) \leftarrow (V,E)$
  for all $q_i \in Q$ do
      $v_i \leftarrow multilingual\_vector(q_i)$
      $V \leftarrow V \cup \{v_i\}$
  end for
  for all $v_i, v_j \in V$ do
      $\mathbf{e}_k \leftarrow S_{ang}(v_i, v_j)$
      $E \leftarrow E \cup \{\mathbf{e}_k\}$
  end for
  for all $\mathbf{e}_k \in E$ do
      if $\mathbf{e}_k < \eta$ then
          $E \leftarrow E \setminus \{\mathbf{e}_k\}$
      end if
  end for
  for all $\mathcal{C}_i \in G(V,E)$ do
      $task_i \leftarrow i$
      for all $v_k \in \mathcal{C}_i$ do
          $L[v_k] \leftarrow task_i$
      end for
  end for
```

The Multilingual Graph Based Clustering (MGBC) relies on the multilingual query representation to encode queries in the search logs (Algorithm 1). Once the queries are converted into vectors in the multilingual semantic space, a weighted undirected graph $G(V,E)$ is created, where query vectors are nodes in the graph and S_{ang} is the weight for the edges connecting the queries. After creating the fully connected graph, the graph is pruned by filtering out edges with $S_{ang} < \eta$, where η is a threshold optimized during the clustering process: $\eta = k/10, 0 < k \leq 10, k \in \mathbb{N}$. The connected components $\mathcal{C}$ in the graph after the pruning process represent the search tasks in the query log. Every connected component receives a unique task label $task_i$, which becomes the label for all the nodes pertaining to the connected component $\mathcal{C}_i$ (Chen and Ji, 2010; Lucchese et al., 2013; Nascimento and De Carvalho, 2011; Sen et al., 2018).

5.4 Results and discussion

Following recent work (Du et al., 2018), we use the F_β score, with $\beta = 1$ for the balanced metric, and $\beta = 0.6$, which gives more weight to the precision of the search task identification (Section 2.2.3). The Student's paired t-test

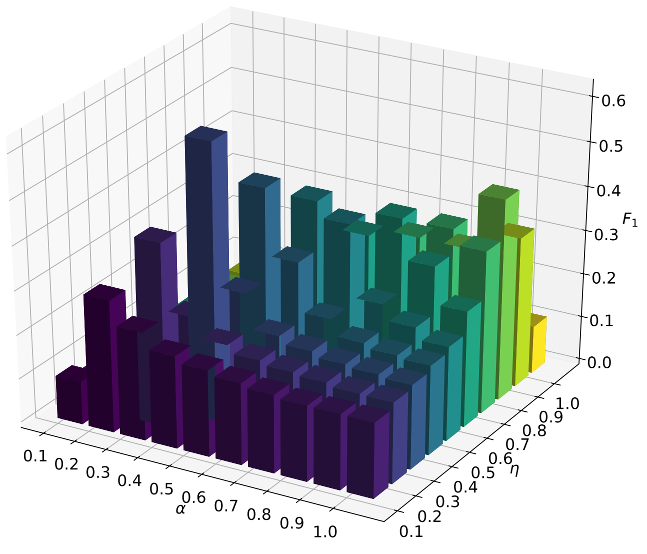


Fig. 5.1: Grid search for α, the parameter for the convex combination of similarities, and η, the threshold for the MGBC clustering method.

provides the test for statistical significance. For evaluating the effectiveness of the proposed approach, we consider a user agnostic search task identification and a personalized search task identification. We also address the existing trade-off when mapping queries to identified search tasks.

Identification method	α	η	F_1	$F_{0.6}$
QC-WCC	0.8	0.4	0.471	0.428
QRY-VEC word2vec	0.6	0.5	0.473	0.441
QRY-VEC tempo-lexical	0.6	0.7	0.538	0.488
MGBC	0.4	0.3	**0.624**	**0.695**

Table 5.1: Clustering performance for the CSTE dataset with ground-truth search task labels. Differences of MGBC results against the baseline have $p \leq 0.05$ for the Student's t-test.

5.4.1 User agnostic search task identification

Language	**ISO 639-1**	η	F_1	$F_{0.6}$
Arabic	ar	0.9	0.447	0.395
Chinese PRC	zh	0.8	0.480	0.473
Chinese Taiwan	zh-tw	0.18	0.482	0.476
Dutch	nl	0.8	0.449	0.431
English	en	0.8	0.456	0.437
German	de	0.8	0.450	0.432
French	fr	0.8	0.484	0.547
Italian	it	0.8	0.452	0.434
Portuguese	pt	0.8	0.458	0.438
Spanish	es	0.8	0.450	0.432
Japanese	ja	0.8	0.453	0.436
Korean	ko	0.9	0.451	0.396
Russian	ru	0.8	0.449	0.429
Polish	pl	0.8	0.460	0.524
Thai	th	0.8	0.444	0.427
Turkish	tr	0.8	0.429	0.401

Table 5.2: Search task identification results for all the supported languages of the MGBC approach. Results for the CSTE dataset have $p \leq 0.05$ between languages.

The dataset for evaluating the clustering approach in a user agnostic scenario is the CSTE dataset. As a baseline, we use the state-of-the-art QRY-VEC (Sen et al., 2018) method, an unsupervised task identification method that uses custom trained tempo-lexical embeddings, averaging the embeddings for each word in the query to compute a single vector per query. We also include results from QC-WCC (Lucchese et al., 2013).

To compare to the QRY-VEC method, we use the same index similarity S_{ind} than the baseline (Sen et al., 2018), which is based on the ClueWeb12B dataset (Callan, 2012). We adjust the angular similarity S_{ang} in equation 5.1 by the use of a convex combination of both angular and index similarities. The similarity between queries q_i, q_j with multilingual vectors v_i, v_j becomes (Lucchese et al., 2013; Sen et al., 2018):

Language	**ISO 639-1**	η	F_1	$F_{0.6}$
Arabic	ar	0.70	0.595	0.648
Chinese PRC	zh	0.70	0.658	0.667
Chinese Taiwan	zh-tw	0.70	0.632	0.604
Dutch	nl	0.70	0.594	0.577
English	en	0.70	0.597	0.544
German	de	0.70	0.550	0.542
French	fr	0.60	0.656	0.748
Italian	it	0.80	0.559	0.492
Portuguese	pt	0.70	0.616	0.610
Spanish	es	0.80	0.641	0.593
Japanese	ja	0.70	0.697	0.737
Korean	ko	0.70	0.573	0.639
Russian	ru	0.70	0.633	0.742
Polish	pl	0.70	0.541	0.578
Thai	th	0.70	0.541	0.533
Turkish	tr	0.70	0.618	0.640

Table 5.3: Search task identification results for all the supported languages of the MGBC approach. Results for the CUSTA dataset have $p \leq 0.05$ between languages.

$$S_{ang}(v_i, v_j) = -\alpha * \arccos\left(\frac{v_i v_j}{|v_i|\,|v_j|}\right) + (1-\alpha) * (S_{ind}) \tag{5.2}$$

where α, η are parameters to be optimized during the process of clustering for search tasks. The optimization uses a grid search with $\alpha = k/10, \eta = k/10$, where $0 < k \leq 10, k \in \mathbb{N}$ (Sen et al., 2018), selecting the model with the best F_1 metric.

Results show that MGBC outperforms the baseline method in search task identification (Table 5.1, Figure 5.1). It gets better performance than both lexically based (QC-WCC) and monolingual query vectors based (QRY-VEC) methods for identifying tasks, highlighting the ability of the multilingual semantic vector space to encode queries for the modeling of search tasks.

Regarding multilingual tests, in addition to the CSTE dataset, we use the Complex User Search Task Analysis (CUSTA) dataset, which has queries mostly in French (Section 2.1). Using Google Cloud Translation API[1], we translate the CSTE and CUSTA datasets to all the languages supported

[1] https://cloud.google.com/translate/docs

by MGBC. Running the search task identification method on automatically translated queries enables the assessment of the method in multilingual task identification.

MGBC uses the angular similarity S_{ang} in equation 5.1 for the multilingual tests. In the results for the CSTE dataset (Table 5.2), F_1 metrics vary from 0.429 with the Turkish language to 0.484 with the French language, which are located around the F_1 metric of 0.456 for the English language, the original language of the dataset (Figure 5.2). Similarly, for the CUSTA dataset (Table 5.3), F_1 metrics vary from 0.541 with the Thai language to 0.697 with the Japanese language, which are located around the F_1 metric of 0.656 for the French language, the original language of the dataset. These results reflect the quality of the multilingual semantic space to represent the search tasks. Overall, no drop in performance is observed despite the use of automatic translation, suggesting an adequate performance in the sixteen languages for the search task identification approach. Although there exist variations in results for the different languages, they are explained by the expected differences in the automatic translation results.

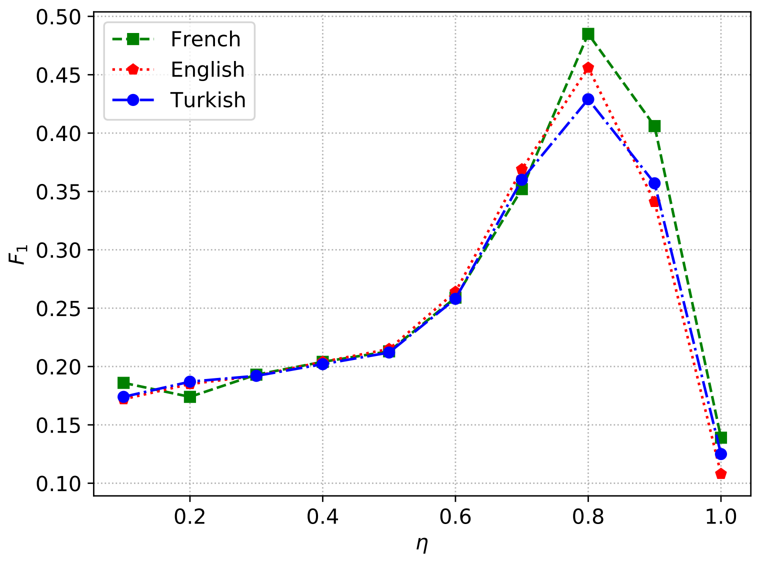


Fig. 5.2: Threshold η search for the MGBC graph clustering method, considering English, French, and Turkish languages.

5.4.2 Personalized search task identification

Identification method	F_1	$F_{0.6}$
LDA-Hawkes	0.871	0.864
BRTs	0.878	0.874
CA-LSTM	0.883	0.887
QRY-VEC	0.851	0.880
MGBC	**0.884**	**0.913**

Table 5.4: Clustering performance for the WSMC12 dataset, which has search task labels per user. The ClueWeb12B dataset provides the index similarity for comparison to the baseline method.

We use the WSMC12 dataset for the evaluation of the proposed MGBC approach in personalized search task identification. As discussed before (Section 2.1), labels in this dataset correspond to search tasks for a particular user identifier, grouping queries by user sessions. If two users run a query for the same search task, the search task label is not necessarily the same. For instance, the user with identifier 9887420 issued the query “maps” and it was labeled with search task ’2’; the user with identifier 1713103 issued the query “maps” as well, but it was labeled with search task ’22’. For that reason, the WSMC12 dataset allows the evaluation of the proposed method in a personalized setup.

Along with the QRY-VEC baseline, we include methods depending on user identifiers for the input, namely, LDA-Hawkes, BRTs, and CA-LSTM (Du et al., 2018; Li et al., 2014a; Mehrotra and Yilmaz, 2017).

Search task identification results show that MGBC matches the clustering performance of CA-LSTM (Table 5.4), even though CA-LSTM is a semisupervised approach, which uses recurrent layers trained in a supervised way. Likewise, MGBC exhibits slight improvements against the other methods used for comparison, though only with $p \leq 0.2$ when using the Student’s t-test for statistical significance. These results indicate that the proposed method matches the identification performance of existing methods in personalized setups, without requiring any supervised components or user identifiers. As MGBC does not require user identifiers as input to the model, it can perform search task identification in user agnostic scenarios as well as personalized scenarios. Indeed, user agnostic scenarios can be crucial in current efforts to address user privacy concerns in intelligent systems.

5.4.3 Mapping queries to search tasks

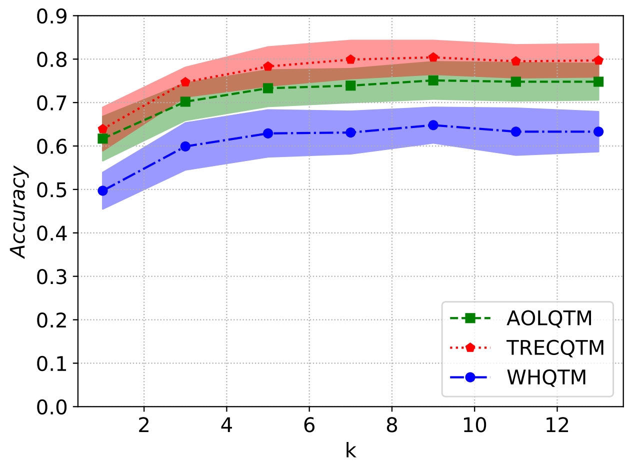
Dataset	Method	Accuracy	F_1	$F_{0.6}$	Query time
AOLQTM	Trie	0.693	0.543	0.543	0.029ms
	BM25	**0.809**	**0.689**	**0.689**	947ms
	NGT	0.751	0.608	0.607	0.308ms
TRECQTM	Trie	0.650	0.519	0.518	0.030ms
	BM25	0.791	0.688	0.688	2532ms
	NGT	**0.804**	**0.705**	**0.704**	0.299ms
WHQTM	Trie	0.471	0.310	0.311	0.032ms
	BM25	0.621	0.453	0.454	6.572min
	NGT	**0.648**	**0.481**	**0.481**	0.368ms

Table 5.5: Model metrics for mapping queries to search tasks. Three datasets are considered to evaluate the performance of the NGT approach. Differences against Trie have $p \leq 0.05$ for the Student's t-test.

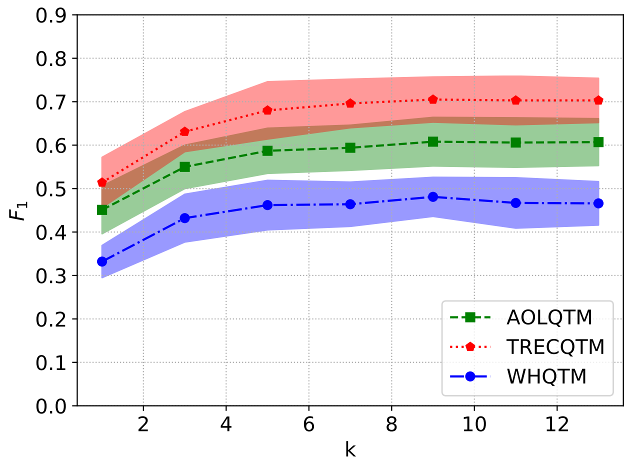
The same multilingual semantic space for query representation and the S_{ang} similarity in Equation 5.1 enables us to address the existing trade-off when mapping new incoming queries to the identified search tasks. Previously analyzed methods for mapping queries face a trade-off between accuracy and execution time. The most accurate method uses an inverted index approach based on a BM25 retrieval model; however, its average time per query is much slower than a Trie data structure implementation, which is the fastest method (Völske et al., 2019). To address this trade-off, we utilize the Neighborhood Graph and Tree (NGT) approximate nearest neighbor method (Iwasaki and Miyazaki, 2018), along with S_{ang} and multilingual query vectors. NGT[2] is a high-speed approach that supports large volumes of data in high-dimensional spaces.

Three benchmark datasets have been proposed to evaluate the mapping of new incoming queries: the AOLQTM dataset, the TRECQTM dataset, and the WHQTM dataset (Section 2.1). For comparison, we use publicly available implementations for the Trie data structure[3] and the BM25 retrieval model[4] with default parameters (Völske et al., 2019; Yang et al., 2020). Experiments run on a virtual machine instance with 8 CPUs of 3GHz and 60GB of RAM.

[2] https://github.com/yahoojapan/NGT
[3] https://github.com/google/pygtrie
[4] https://github.com/nhirakawa/BM25



(a)



(b)

Fig. 5.3: Test NGT in the multilingual semantic space with multiple values of k nearest neighbors. Results for AOLQTM, TRECQTM, and WHQTM datasets include (a) Accuracy (b) F_1.

We compute time per query as the average time for mapping 10^4 queries, while accuracy is measured using a leave-one-out evaluation, independently selecting 100 random queries from the dataset and repeating the evaluation during 50 runs (Völske et al., 2019).

We test several values of k nearest neighbors for NGT (Figure 5.3), finding $k = 9$ as the best performing setup. NGT is several times faster than the inverted index based on BM25 (Table 5.5), keeping average times per query below half a millisecond. The speedup obtained with NGT does not imply a deterioration in the accuracy metrics for TRECQTM and WHQTM datasets. Also, AOLQTM differences are much lower than the Trie data structure drop in metrics. Similarly, NGT is more accurate than the Trie data structure in all the three datasets; nonetheless, the latter continues to be faster in terms of average time per query.

5.5 Summary

The proposed MGBC multilingual search task identification approach enables the modeling of search tasks from query logs, supporting queries in sixteen languages. Experiments show that the proposed approach outperforms baseline identification methods. Also, MGBC is user-independent, enabling its use in both user agnostic and personalized search task identification applications. Moreover, the same multilingual semantic space and query similarity of MGBC can be used with NGT nearest neighbor method to address the existing trade-off when mapping new queries to identified search tasks. NGT provides metrics at the same level of the BM25 retrieval model results; however, it is several times faster, keeping query response times below half a millisecond, a crucial aspect for running on the fly applications for supporting search engine users.

Chapter 6
Modeling User Search Tasks with a Language-agnostic Unsupervised Approach

All models are wrong, but some models are useful.

George Box

Although MGBC supports queries in sixteen languages (Chapter 5), users around the world use many more languages when accessing search systems. Also, there are multiple languages with not enough datasets to train language models, arising the need for utilizing language-agnostic representations for queries, representations that can support queries even when the language is not part of the training dataset of the language model. Moreover, as discussed before (Section 2.4), clicked URLs provide a way to model user search intent from the relationship between queries and clicked documents (Zhang et al., 2019). For example, a similarity metric based on clicked URLs is the second most important in Bestlink SVM, after a similarity metric based on query semantics (Wang et al., 2013a).

Consequently, in this chapter we propose a language-agnostic, user intent aware approach to model search tasks from user interactions with search systems. The proposed approach leverages user intent modeling from clicked query-document pairs, latent representations of queries in a language-agnostic space, and graph-based clustering to model search tasks in an unsupervised approach. Experimental results demonstrate the proposed approach outperforms recent work in search task modeling, supporting user queries in multiple languages. It can also produce search task modeling results in the order of milliseconds, an essential aspect for conversational systems and user support applications requiring realtime results.

6.1 Introduction

Conversational AI systems are becoming increasingly popular because of advances in speech recognition, natural language understanding, text-to-speech synthesis, and the availability of digital personal assistants (Khatri et al., 2018; Thomas et al., 2020; Venkatesh et al., 2018). Personal assistants like Amazon Alexa, Apple Siri, Google Assistant, and Microsoft Cortana are now available in smartphones, tablets, desktops, and dedicated smart speakers (Khatri et al., 2018; Zamani and Craswell, 2020). Consequently, the increasing popularity and availability of conversational systems make conversational information seeking a major emerging area of research (Anand et al., 2020; Zamani and Craswell, 2020).

In conversational information seeking and other search systems, modeling the search tasks that users perform to satisfy their information needs is a crucial step (Mehrotra and Yilmaz, 2017; Sen et al., 2018). Search task modeling is a step in the process to make search systems more coherent, natural, engaging, and conversational (Khatri et al., 2018; Rosset et al., 2020; Venkatesh et al., 2018). Multiple user supporting applications benefit from search task modeling, including conversational question suggestion, personalization in e-commerce, product recommendations, query term prediction, query suggestions, query reformulation, and results ranking (Hearst, 2009; Mehrotra and Yilmaz, 2017; Rosset et al., 2020; Tamine et al., 2020; Völske et al., 2019). Even informative conversations with digital assistants can benefit from correctly modeling the search tasks, as the subjective perception of the quality in the conversation is strongly related to the accurate tracking of the topic (Venkatesh et al., 2018).

As discussed before, users around the world access search systems in multiple languages, making it essential to process users' requests with language-agnostic models. Also, search systems and user supporting applications require realtime responses when processing user information needs. For instance, multimodal search in conversational systems runs multiple processes in parallel, post-processing their outputs to generate a message answering the user request; hence, modeling can not exceed the timeout periods set on the search system (Zamani and Craswell, 2020). Similarly, user clicks are strongly related to the user intent (Zhang et al., 2019). Different queries with similar clicked URLs can pertain to the same information need (Mehrotra and Yilmaz, 2017), and analyzing clicked URLs can help disambiguate queries (Craswell et al., 2020a).

Our contributions in this chapter are threefold. First, we propose a language-agnostic search task modeling (LASTM) approach to model search tasks from user interactions with search systems. Second, given the relationship between clicked URLs and user intent, we propose a user intent modeling technique leveraging a large scale query - clicked document collection in the query latent space. Third, to enable the utilization of LASTM in conversational search systems and user supporting applications requiring responses on the fly, we propose a realtime method for mapping incoming queries to the modeled user search tasks directly on the query latent space.

6.2 Related limitations

User interactions with search systems enable modeling the search tasks that users perform to satisfy their information needs (Hearst, 2009). In particular, search query logs can be mined for search task modeling using methods such as heuristics-based models, semi-supervised clustering, bayesian approaches, and graph-based clustering (Section 2.4). However, most search task modeling methods (Du et al., 2018; Hagen et al., 2013; Li et al., 2014a; Lucchese et al., 2013; Mehrotra et al., 2016; Mehrotra and Yilmaz, 2017; Sen et al., 2018; Wang et al., 2013a) are monolingual. Although MGBC supports several languages through MUSE, it can only process queries in sixteen languages. Additionally, when using ClueWeb12B for calculating query similarities, MGBC can only support user queries in English.

By the same token, most search task modeling methods (Du et al., 2018; Li et al., 2014a; Lucchese et al., 2013; Mehrotra et al., 2016; Sen et al., 2018) fail to take into account clicked URLs when processing search query logs, even though clicked URLs have a critical correlation to the user intent (Zhang et al., 2019). Also, conversational information seeking systems and multiple applications supporting users search efforts require results on the fly. Building models from scratch when a user submits a query could create large processing times, forcing search systems to trigger timeout intervals (Zamani and Craswell, 2020). Similarly, waiting for forward queries to provide context (Du et al., 2018) can render models unfeasible in realtime setups. Also, some models requiring user identifiers (Du et al., 2018; Hagen et al., 2013; Li et al., 2014a; Mehrotra and Yilmaz, 2017) can not be used in user-independent modeling scenarios (Craswell et al., 2020a; Sen et al., 2018).

6.3 User search task modeling

LASTM is an unsupervised method that leverages latent representations of queries in a language-agnostic space, user intent modeling from clicked query-document pairs, and graph-based clustering to model user search tasks. It can also produce a realtime mapping of queries to modeled search tasks. In contrast with previous work (Du et al., 2018; Hagen et al., 2013; Li et al., 2014a; Lucchese et al., 2013; Mehrotra et al., 2016; Mehrotra and Yilmaz, 2017; Sen et al., 2018; Wang et al., 2013a), our proposed approach supports multiple languages through a language-agnostic latent space. The proposed approach is also independent of user identifiers, enabling modeling of search tasks in both user-independent and personalized scenarios. It also differs from prior methods (Du et al., 2018; Li et al., 2014a; Lucchese et al., 2013; Mehrotra et al., 2016; Sen et al., 2018) by leveraging clicked URLs to model user intent (Zhang et al., 2019) in the query latent space.

6.3.1 Language-agnostic query representation

As users worldwide submit queries in different languages to satisfy their information needs, Language-agnostic BERT Sentence Embedding (LABSE) (Feng et al., 2020) provides the sentence embeddings to represent user queries in a language-agnostic latent space. Using a 12-layer transformer architecture (Devlin et al., 2019; Vaswani et al., 2017) in a dual configuration, LABSE takes the transformer's hidden state for the last token in the sentence to generate the query representation.

The query representation using LABSE has the ability to perform zero-shot cross-lingual transfer, supporting queries in languages that are not part of the training dataset. When performing tests with the TAOEBA dataset (Artetxe and Schwenk, 2019), LABSE obtains an 83.7% accuracy, while the baseline Language-agnostic Sentence Representations (Artetxe and Schwenk, 2019) gets 65.5%, even though more than 30 languages in the TAOEBA dataset were not part of the LABSE training data (Feng et al., 2020).

We use the cosine proximity (Feng et al., 2020; Sen et al., 2018) to compute the similarity between query representations in the language-agnostic latent space. Formally, given a pair of queries q_i, q_j with latent representations u_i, u_j, the similarity between query representations S_{lat} is calculated as follows (Feng et al., 2020; Sen et al., 2018):

$$S_{lat}(u_i, u_j) = \frac{u_i u_j}{|u_i|\,|u_j|} \tag{6.1}$$

6.3.2 User intent modeling

User clicks play a critical role in modeling user intent – the information need the user wants to satisfy by performing the search task (Zhang et al., 2019). Query term match between queries for the same information need can be very low; even different queries pertaining to the same search task can have similar clicked URLs (Mehrotra and Yilmaz, 2017; Zhang et al., 2019). Also, analysis of clicked URLs can help disambiguate queries, revealing which documents users clicked when performing their search tasks (Craswell et al., 2020a).

To model user intent, we use the Open Resource for Click Analysis in Search (ORCAS) (Craswell et al., 2020a), a collection containing 18.8 million clicked document - query pairs for 10.4 million unique queries. Clicked docu-

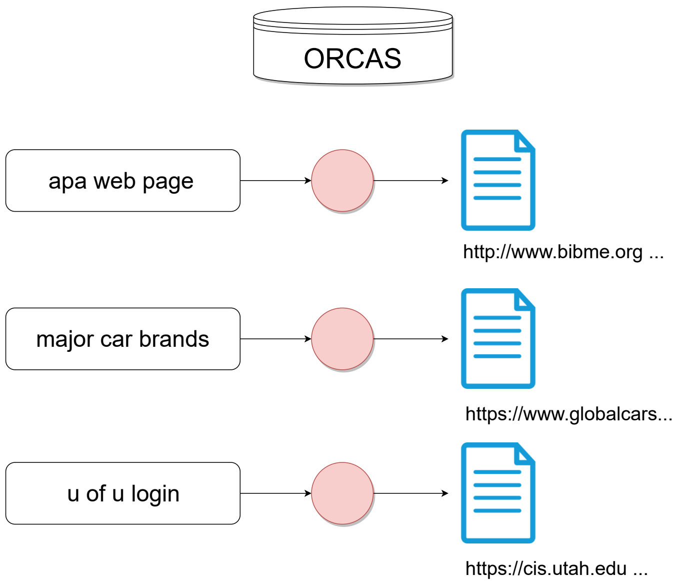


Fig. 6.1: User intent database with clicked document – query pairs from the ORCAS dataset (Craswell et al., 2020b). Queries are encoded in a language-agnostic semantic space using LABSE (Feng et al., 2020).

ments are represented using the TREC document identifier in the TREC Deep Learning document collection (Craswell et al., 2020b). We encode queries in ORCAS in the language-agnostic latent space (Feng et al., 2020), creating a user intent database $\mathcal{D}_M$ with clicked document – query pairs (Figure 6.1). To retrieve the most relevant documents for a given user query in the database, we use Scalable Nearest Neighbor (ScaNN) (Guo et al., 2020), a state-of-the-art method for large-scale retrieval tasks.

In retrieval tasks, it is usual to learn latent representations of queries and documents in the same latent space. Then, we encode the query and look for documents in the latent space by performing the inner product against the query, selecting the documents that are as close as possible to the query. Such documents will have the maximum possible inner product against the query, representing the most relevant results from the documents' database. This procedure is commonly known as MIPS (Maximum Inner Product Search).

Computing the inner product of a query against all the documents in the database can become unfeasible, especially for large databases. Thus, there exist two possibilities to retrieve relevant results more quickly. One possibility is partitioning the latent space so that the number of inner product operations is reduced to the data points in the closest partition. For partitioning, there exist methods like local sensitive hashing, graph search, and tree search.

The other possibility is to compute the inner products faster, improving the scoring rate.

A strong alternative to improve the scoring rate is quantization. It makes scoring data points faster, improving the rate of inner product calculations. It also uses the memory bandwidth more efficiently because it can handle more data points per memory access, maximizing CPU utilization. Morevoer, quantization produces effiicency gains in terms of storage size, using more efficiently the disk or memory space where the data points are stored.

Prior research in quantization focused on optimizing the reconstruction error of the quantization method for all points in the database. However, it is better to assign a higher weight to the most relevant items while safely reducing the weight for irrelevant items. This intuition produces a weighted error optimization, commonly known as anisotropic vector quantization, where items close to the query receive a higher weight in the error computations than items orthogonal to it. By doing so, anisotropic vector quantization improves the relevance of MIPS results. ScaNN uses anisotropic vector quantization, improving over NGT (Section 5.4.3) and several other approximate nearest neighbor methods (Guo et al., 2020). Therefore, we use ScaNN to retrieve results from the ORCAS document collection.

Even though ORCAS has queries exclusively in English, doing MIPS directly on the language-agnostic latent space enables user intent modeling in any language LABSE can support. Hence, we can leverage the existing relationship between clicked URLs and user intent (Zhang et al., 2019) by searching the $\mathcal{D}_M$ database.

Formally, given a database $\mathcal{D}_M = \{m_i\}_{i=1,2,\ldots,n}$ formed from a clicked query-document dataset $\mathcal{D}_Q$ with n data points, where each data point $m_i \in \mathbb{R}^p$ is the latent representation of the query $q \in \mathcal{D}_Q$ in the p-dimensional language-agnostic latent space, we want to find the k most relevant documents $\{d_j\}_{j=1,2,\ldots,k} \in \mathcal{D}_M$ for the user query $u \in \mathbb{R}^p$. Therefore, we search for the k points with the maximum inner product with the user query u as follows (Guo et al., 2020):

$$MIPS(\mathcal{D}_M, u) = \{d_j\}_{j=1,2,\ldots,k} = \arg \max_{m_i \in \mathcal{D}_M} \langle u, m_i \rangle \tag{6.2}$$

Given a user query pair q_i, q_j with latent representations u_i, u_j, the similarity based on user intent S_{int} is calculated using the Jaccard coefficient for the top thousand relevant documents in the database $\mathcal{D}_M$ (Sen et al., 2018):

$$D_i = MIPS(\mathcal{D}_M, u_i) \tag{6.3}$$

$$D_j = MIPS(\mathcal{D}_M, u_j) \tag{6.4}$$

$$S_{int}(u_i, u_j) = \frac{|D_i \cap D_j|}{|D_i \cup D_j|} \tag{6.5}$$

6.3.3 Unsupervised search task modeling

We now integrate user intent modeling and language-agnostic query representations with graph-based clustering (Chen and Ji, 2010) to model search tasks (Algorithm 2). First, we encode queries in the latent space (Section 6.3.1); every query embedding becomes a node in the weighted graph. Then, we compute the similarities between pairs of queries to create the edges of the weighted graph. The similarity between queries S_{qry} is a convex combination of the similarity in the latent space S_{lat} and the similarity based on user intent S_{int}. Given a pair of queries q_i, q_j with latent representations u_i, u_j, query similarity S_{qry} is calculated as follows (Section 5.4):

$$S_{qry}(u_i, u_j) = \alpha * S_{lat}(u_i, u_j) + (1 - \alpha) * S_{int}(u_i, u_j) \tag{6.6}$$

After finishing edge weight calculations, we prune the weighted graph, deleting edges with $S_{qry} < \eta$. The resulting connected components $\mathcal{C}$ in the graph constitute the search tasks, so we assign a unique task label $task_i$ to every connected component. All the queries pertaining to a connected component receive the same task label. A grid search (Figure 6.2) optimizes parameters η and α, using $\eta = k/10, \alpha = k/10, 0 < k \leq 10, k \in \mathbb{N}$ (Chen and Ji, 2010; Lucchese et al., 2013; Sen et al., 2018).

6.3.4 Realtime mapping of new queries

Most search systems and user supporting applications require results in realtime. Applications like contextual topic modeling in conversational search (Khatri et al., 2018), query suggestion, or query reformulation can not afford to wait for large processing times. It is essential to return an answer in a few milliseconds. Hence, once the user performs a search request, we map the new incoming query to the labels extracted with Algorithm 2 so that we can model the search task in realtime. To do the mapping, we use the same MIPS method with anisotropic vector quantization (Guo et al., 2020) that we used in Section 6.3.3.

The search task database maps the latent representation of the queries in the search log $\mathcal{Q}_L$ to the extracted task labels $\mathcal{L}_T$. Formally, given a database $\mathcal{Q}_T = \{m_i\}_{i=1,2,\ldots,n}$ formed from the search query log $\mathcal{Q}_L$ with search task labels $\mathcal{L}_T$ returned from Algorithm 2, where each datapoint $m_i \in \mathbb{R}^p$ is the latent representation of the query $q \in \mathcal{Q}_L$ in the p-dimensional language-agnostic space. For an incoming query q_i, we compute the latent representation u_i; then, we retrieve the search task labels T of the k closest queries in the language-agnostic latent space using MIPS:

$$T = MIPS(\mathcal{Q}_T, u_i) \tag{6.7}$$

Algorithm 2 LASTM

```
Inputs: Search query log $\mathcal{Q}_L$, Clicked query-document collection $\mathcal{D}_Q$
Output: Task labels $\mathcal{L}_T$

  // Build database for user intent
  $\mathcal{D}_M \leftarrow \{\}$
  for all $q_i, d_i \in \mathcal{D}_Q$ do
      $x_i \leftarrow language\_agnostic\_space(q_i)$
      $\mathcal{D}_M \leftarrow \mathcal{D}_M \cup \{x_i, d_i\}$
  end for

  // Model search tasks
  $V \leftarrow \{\}, E \leftarrow \{\}, G(V, E) \leftarrow (V, E)$
  for all $q_i \in \mathcal{Q}_L$ do
      $u_i \leftarrow language\_agnostic\_space(q_i)$
      $V \leftarrow V \cup \{u_i\}$
  end for

  for all $v_i, v_j \in V$ do
      $S_{lat}(v_i, v_j) = cos(v_i, v_j)$
      $D_i, D_j \leftarrow$ document IDs for $v_i, v_j$ from $\mathcal{D}_M$
      $S_{int}(v_i, v_j) = Jaccard(D_i, D_j)$
      $\mathbf{e}_k \leftarrow \alpha * S_{lat}(v_i, v_j) + (1 - \alpha) * S_{int}(v_i, v_j)$
      $E \leftarrow E \cup \{\mathbf{e}_k\}$
  end for

  for all $\mathbf{e}_k \in E$ do
      if $\mathbf{e}_k < \eta$ then
          $E \leftarrow E \setminus \{\mathbf{e}_k\}$
      end if
  end for

  for all $\mathcal{C}_i \in G(V, E)$ do
      $task_i \leftarrow i$
      for all $v_k \in \mathcal{C}_i$ do
          $\mathcal{L}_T[v_k] \leftarrow task_i$
      end for
  end for

  return $\mathcal{L}_T$
```

Once we have the search task labels T of the k closest queries, we return the task label with the highest number of occurrences in T.

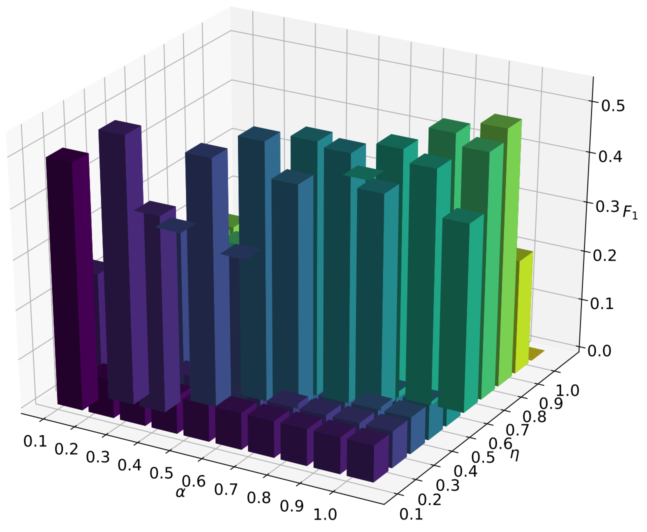


Fig. 6.2: Grid search for α, the parameter for the convex combination of similarities, and η, the threshold for the LASTM clustering method.

6.4 Results and discussion

In this section, we analyze LASTM in user independent search task modeling and realtime mapping of incoming queries. Following previous work (Du et al., 2018), we calculate model performance with the F_β score. We consider both $\beta = 1.0$ and $\beta = 0.6$ (Section 5.4). Moreover, we use open source implementations for ScaNN[1], NetworkX[2] in graph-based clustering, and the publicly available pretrained model for LABSE[3].

[1] `https://github.com/google-research/google-research/tree/master/scann`

[2] `https://networkx.github.io`

[3] `https://tfhub.dev/google/LaBSE/1`

Language	**ISO 639-1**	F_1		$F_{0.6}$	
		MGBC	LASTM	MGBC	LASTM
Arabic	ar	0.447	**0.521**	0.395	**0.490**
Chinese PRC	zh	0.480	**0.539**	0.473	**0.513**
Chinese Taiwan	zh-tw	0.482	**0.540**	0.476	**0.515**
Dutch	nl	0.449	**0.534**	0.431	**0.511**
English	en	0.456	**0.538**	0.437	**0.512**
German	de	0.450	**0.533**	0.432	**0.511**
French	fr	0.484	**0.539**	**0.547**	0.512
Italian	it	0.452	**0.540**	0.434	**0.517**
Portuguese	pt	0.458	**0.537**	0.438	**0.514**
Spanish	es	0.450	**0.541**	0.432	**0.516**
Japanese	ja	0.453	**0.522**	0.436	**0.495**
Korean	ko	0.451	**0.523**	0.396	**0.501**
Russian	ru	0.449	**0.533**	0.429	**0.508**
Polish	pl	0.460	**0.536**	**0.524**	0.512
Thai	th	0.444	**0.522**	0.427	**0.489**
Turkish	tr	0.429	**0.538**	0.401	**0.513**

Table 6.1: Search task modeling results for the CSTE dataset in all the languages supported by the MGBC method. Differences between MGBC and LASTM results have $p \leq 0.05$ for the Student's t-test.

6.4.1 Search task modeling

The CSTE dataset and the CUSTA dataset are used for experiments. As a baseline, we use MGBC (Section 5.3), calculating metrics for all the languages supported by the baseline. Queries in the CSTE dataset are in English, while queries in the CUSTA dataset are mostly in French, with very few English entries (Section 2.1). Hence, we perform machine translation with the Google Cloud Translation API[4] for evaluating LASTM in all the languages supported by MGBC.

The proposed approach improves the search task modeling performance of the baseline method in the two datasets used for testing. Using the CSTE dataset (Table 6.1), LASTM surpasses MGBC in all the languages supported by the baseline, obtaining up to 10.9% ($p \leq 0.05$) improvement in the F_1 score for the Turkish language; similarly, LASTM obtains better $F_{0.6}$ scores

[4] `https://cloud.google.com/translate`

Language	**ISO 639-1**	F_1		$F_{0.6}$	
		MGBC	LASTM	MGBC	LASTM
Arabic	ar	0.595	**0.608**	0.648	**0.665**
Chinese PRC	zh	0.658	**0.667**	0.667	**0.688**
Chinese Taiwan	zh-tw	0.632	**0.672**	0.604	**0.694**
Dutch	nl	0.594	**0.648**	0.577	**0.761**
English	en	0.597	**0.657**	0.544	**0.705**
German	de	0.550	**0.642**	0.542	**0.715**
French	fr	0.656	**0.732**	0.748	**0.750**
Italian	it	0.559	**0.604**	0.492	**0.602**
Portuguese	pt	0.616	**0.622**	0.610	**0.636**
Spanish	es	0.641	**0.643**	0.593	**0.712**
Japanese	ja	**0.697**	0.619	**0.737**	0.571
Korean	ko	**0.573**	0.563	**0.639**	0.561
Russian	ru	0.633	**0.641**	0.742	**0.754**
Polish	pl	0.541	**0.598**	0.578	**0.605**
Thai	th	0.541	**0.603**	0.533	**0.636**
Turkish	tr	0.618	**0.653**	0.640	**0.711**

Table 6.2: Search task modeling results for the CUSTA dataset in all the languages supported by the MGBC method. Differences between MGBC and LASTM results have $p \leq 0.05$ for the Student's t-test.

in fourteen out of sixteen languages, getting an improvement of up to 11.2% ($p \leq 0.05$) in the Turkish language. Furthermore, the monolingual QRY-VEC method, which supports queries in English, obtains an F_1 score of 0.538 and an $F_{0.6}$ score of 0.488 (Sen et al., 2018). Consequently, there is no loss in modeling performance when comparing LASTM to the QRY-VEC method. For the CUSTA dataset (Table 6.2), we observe improvements in fourteen out of the sixteen languages supported by MGBC; LASTM generates up to 9.2% ($p \leq 0.05$) improvement in the F_1 score for the German language and up to 18.4% ($p \leq 0.05$) improvement in the $F_{0.6}$ score for the Dutch language.

Both the similarity between query representations S_{lat} and the similarity based on user intent S_{int} contribute to the search task modeling results (Figure 6.3). In the grid search for the CSTE dataset, α values averaged 0.238 ± 0.099. For the CUSTA dataset, α values in the grid search averaged 0.731 ± 0.157. These α values indicate that the convex combination (Equation 6.6)

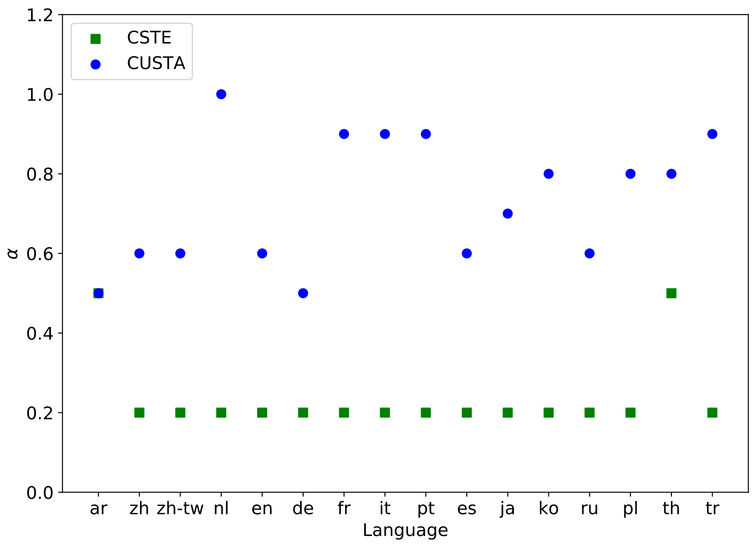


Fig. 6.3: α values for the convex combination of query similarities. Results include values after finishing the grid search for CSTE and CUSTA datasets.

effectively relies on the two similarities to compute the edges for the weighted graph.

From a language coverage perspective, the query representation for LASTM is trained with 109 languages and can perform zero-shot cross-lingual transfer to multiple more languages (Feng et al., 2020). In contrast, the baseline only supports sixteen languages, making LASTM language accessibility at least seven times larger when considering training languages only. When comparing LASTM to monolingual models, there is an improvement of two orders of magnitude in language accessibility (Figure 6.4). The improvements in modeling performance and language accessibility highlight the importance of considering user intent along with language-agnostic query representation for modeling search tasks.

6.4.2 Mapping of incoming queries

To analyze the performance of LASTM for mapping new incoming queries, we run the mapping method using the three benchmark datasets previously proposed for query-task mapping (Section 2.1): the AOLQTM dataset, the TRECQTM dataset, and the WHQTM dataset (Section 2.1). Additionally,

Dataset	Method	Accuracy	F_1	$F_{0.6}$	Query time
AOLQTM	Trie	0.693	0.543	0.543	0.029ms
	BM25	**0.809**	**0.689**	**0.689**	947ms
	MGBC	0.751	0.608	0.607	0.308ms
	LASTM	0.802	0.678	0.677	0.490ms
TRECQTM	Trie	0.650	0.519	0.518	0.030ms
	BM25	0.791	0.688	0.688	2532ms
	MGBC	0.804	0.705	0.704	0.299ms
	LASTM	**0.822**	**0.729**	**0.728**	0.481ms
WHQTM	Trie	0.471	0.310	0.311	0.032ms
	BM25	0.621	0.453	0.454	6.572min
	MGBC	**0.648**	**0.481**	**0.481**	0.368ms
	LASTM	0.558	0.389	0.389	0.982ms

Table 6.3: Realtime mapping of queries to search tasks. Differences against baseline MGBC results have $p \leq 0.05$ for the Student's t-test.

we use a leave-one-out evaluation, independently selecting one hundred random queries from the dataset and repeating the evaluation for fifty runs. Experiments run on a virtual machine instance with 8 CPUs of 3GHz and 60GB of RAM. Metrics include accuracy, F_1, $F_{0.6}$, and query time. To measure query time, we take the average time for mapping a single query, using 10^4 mappings to compute the average (Völske et al., 2019). As a baseline, we use the MGBC approach for query task mapping. MGBC combines the Neighborhood Graph and Tree approximate nearest neighbor method (Iwasaki and Miyazaki, 2018) with the MUSE latent space for query encoding (Section 5.4.3). For reference, we also include results using the Trie[5] data structure and the BM25[6] retrieval model (Völske et al., 2019; Yang et al., 2020).

Figure 6.5 depicts the optimization experiments for the number of top k results from ScaNN to consider. After running tests for $k = [1, 3, 5, 7, 9, 11, 13]$, we found that top $k = 7$ results from ScANN generates the optimal configuration, providing the best results for task mapping while keeping the time per query under a millisecond (Table 5.5). Low response time is an essential aspect for applications supporting users in realtime setups. Long answer times could affect the interaction of the search system with the users, especially in conversational and multimodal search systems, where a post-processing step is required to generate a response to the user request (Khatri et al., 2018; Zamani and Craswell, 2020). Similarly, long answer times could trigger inter-

[5] `https://github.com/google/pygtrie`

[6] `https://github.com/nhirakawa/BM25`

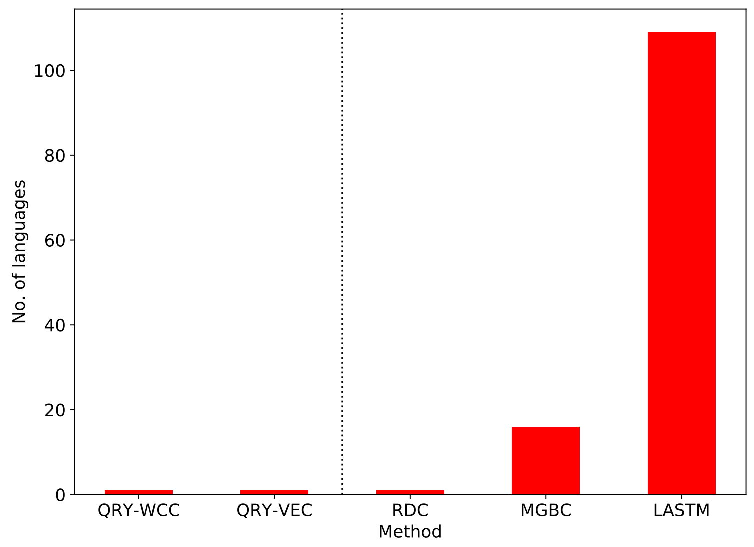
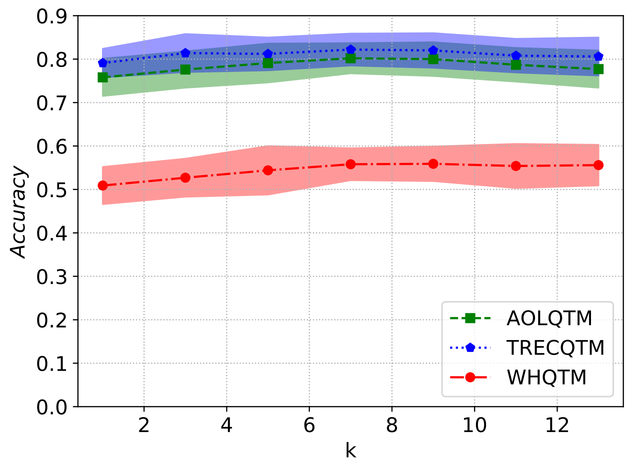


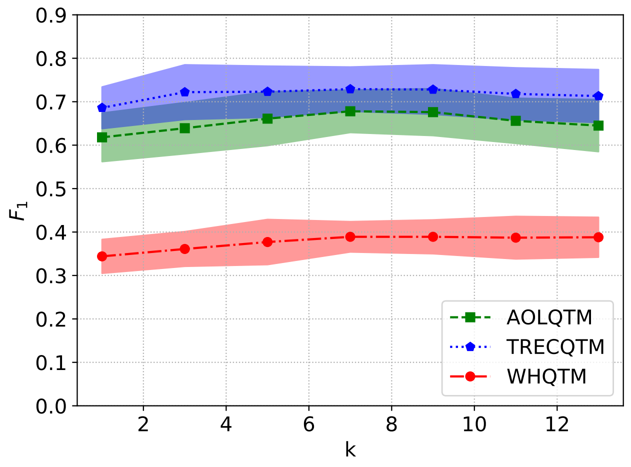
Fig. 6.4: Improvement in language accessibility when comparing LASTM against the baseline, RDC, and existing monolingual methods for search task modeling.

nal timeout intervals (Zamani and Craswell, 2020), forcing search systems to ignore search task mapping results while doing internal post-processing.

Results from query-task mapping show that the proposed method surpases MGBC in two out of the three benchmark datasets (5.5). For the TREC-based dataset, LASTM improves the F1 score of the baseline by 2.4% ($p \leq 0.05$), while keeping processing times under a millisecond. For the AOL-based dataset, LASTM surpasses the baseline method, obtaining a 7.0% improvement in the F_1 score ($p \leq 0.05$); likewise, LASTM obtains similar results to BM25, but it is faster when comparing to the BM25 implementation used for experiments. For the WikiHow-based dataset, LASTM underperforms MGBC and BM25 (Table 5.5). Regarding the number of user queries per task, we find that the TREC-based dataset has an average of 28 user queries per search task, while the WikiHow-based dataset has an average of 2 user queries per task. Hence, the WikiHow-based dataset contains mostly simple tasks, which users can solve with a few queries (Hearst, 2009). Task mapping results suggest that LASTM is better than the baseline and reference methods when mapping search tasks containing multiple queries, while MGBC is better when mapping simple search tasks in realtime.



(a)



(b)

Fig. 6.5: Search task mapping results in the language-agnostic latent space for AOLQTM, TRECQTM, and WHQTM datasets. Results include several values of top k from the ScaNN index, considering (a) Accuracy (b) F_1.

6.5 Summary

In this chapter, we proposed LASTM, an unsupervised method for modeling search tasks from user interactions with search systems. The proposed model outperforms a state-of-the-art baseline both in modeling performance as well as the number of languages it can support, highlighting the importance of language-agnostic latent spaces for query representation and the importance of considering clicked URLs to model user intent. Also, it is independent of user identifiers, enabling modeling search tasks in user-independent or personalized applications. The modeling performance of LASTM, its language-agnostic capacity, and its ability to support realtime modeling can benefit search systems and user supporting applications, constituting an essential step in the effort to make search more coherent, conversational, engaging, and natural.

Chapter 7
Conclusions

It is a peculiarity of man that he can only live by looking to the future.

Viktor Frankl

In this dissertation, we proposed four models that learn search patterns from users interactions with search systems, namely, the Language-agnostic Search Task Modeling (LASTM) approach, the Multilingual Graph Based Clustering (MGBC) method for search task identification, the Recurrent Deep Clustering (RDC) model to extract users' search tasks, and a recurrent neural architecture for segmenting search query logs.

The recurrent neural architecture learns to detect search task boundaries in pairs of subsequent queries, determining if adjacent queries in chronologically ordered query logs represent a task change or not, and taking as input the query and its timestamp only. It does not rely on the context provided by surrounding queries to achieve its maximum modeling performance. Also, it is possible to fine-tune the model, allowing the processing of small query logs. This is especially important given the scarcity of publicly available labeled datasets from search systems. Once trained, it is several times faster than a heuristics-based baseline. Additionally, as the proposed architecture does not require surrounding queries for context, it can provide segmentation results for tasks with one or few queries. The mean number of queries per task in datasets from widely used search engines is low, and information needs like fact-finding and other simple tasks are usually solved with only one query.

RDC extends the foregoing recurrent architecture to propose a model for search task extraction. Taking the recurrent architecture as the encoder for the user queries, RDC implements a dual-channel configuration that allows it to simultaneously learn query latent representations and cluster queries into groups of search tasks. The learning of query representations is performed using back translation, a strong data augmentation technique that renders negligible the effects of pretraining RDC. For that reason, the supervised pretraining can be discarded, providing an unsupervised method for search task modeling. On top of the unsupervised nature of the proposed method, it is also parametric, providing a fixed-sized query encoder regardless of the number of queries in the search system log.

MGBC provides an unsupervised, non-parametric model to identify user search tasks, combining a multilingual latent space for query representation with graph-based clustering, and using the angular distance in the latent space to group related queries. MGBC supports realtime modeling, as the same multilingual semantic space and query similarity is used to address the existing trade-off when mapping new queries to identified search tasks. For that reason, MGBC keeps query response times below half a millisecond, a crucial aspect for running on the fly applications for supporting search system users. Likewise, MGBC is multilingual, facilitating the identification of search tasks in sixteen languages, including the most used ones like Chinese, English, Arabic, French, and Spanish. Moreover, as MGBC is independent of user identifiers, it can perform search task modeling in user agnostic or personalized scenarios.

Finally, LASTM models user search tasks by leveraging a language-agnostic semantic space for query representation, along with user search intent from the relationship between queries and clicked documents. As user intent modeling is also performed in the language-agnostic semantic space, LASTM can use both semantic relatedness and user search intent to group similar queries in the search system logs. The query representation can realize zero-shot transfer learning, which allows it to support queries in languages that were not part of the training collection for the semantic space. Similar to MGBC, it is possible to get realtime modeling results by mapping incoming queries to modeled search tasks; also, the proposed model is independent of user identifiers, enabling its use in user agnostic or personalized scenarios.

Indeed, a crucial aspect in the proposed models is user privacy, which is part of broader efforts in ethics for intelligent systems. None of the proposed models rely on user identifiers as direct input to the method. Though the neural model for search task segmentation requires the queries in the log to be ordered by timestamps and grouped by user, this ordering does not require direct use of the identifiers as input to the neural model. Similarly, RDC is user agnostic. It does not require user identifiers either for deep clustering or self-training with back translation. MGBC and LASTM are user agnostic as well. They take only queries as input, and the relationship between queries and clicked documents we use to model user search intent does not utilize any user identifiers.

Another crucial aspect is realtime response times, especially for interactive search. Except for RDC, all proposed models take into account the need for realtime results in modern search systems. Most search systems need to provide results in realtime. Conversational information seeking can also implement time thresholds when aggregating results from multiple systems, so that they can provide timely responses during the interaction with users. Exceeding such limits render the system response unusable or can introduce unnecessary delays in the process users carry out to fulfill their information needs. Therefore, MGBC, LASTM, and the proposed method for search task segmentation provide modeling results under a millisecond. The neural architecture for search task segmentation can be even faster than the heuristics-based baseline once the neural network is trained. Moreover, both MGBC and LASTM rely on query-task mapping with fast approximate nearest neighbors methods for modeling search tasks in realtime.

Furthermore, language accessibility is at the core of the MGBC and LASTM approaches. As discussed before, most labeled and unlabeled datasets publicly available are in English, but users worldwide access search systems in many languages. Therefore, MGBC represents queries in a multilingual semantic space, supporting queries in sixteen languages. LASTM goes even further, using a language-agnostic semantic space for representing queries and modeling user search intent. When compared against existing monolingual modeling methods, LASTM, which supports more than a hundred languages, represents an improvement of two orders of magnitude in language accessibility.

Overall, our proposed models provide state-of-the-art performance, improving on existing methods for modeling search patterns, and taking into account user privacy, realtime responses, and language accessibility. Proposed models can be part of many user assisting applications and retrieval models, underpinning efforts of search systems to assist users while they carry out their information seeking. They can also be used to design search systems based on tasks, as well as to aid digital assistants to follow the topic in conversational search. Assisting users while they run the sequence of steps they devise to solve their information needs allows them to deal with the mental overload that information seeking requires.

7.1 Perspectives for further research

In our proposed non-parametric methods, we use existing ground-truth labels to perform the grid search, producing models as accurate as possible with respect to the ground-truth. Nonetheless, it would be interesting to explore ways to optimize the model using metrics that are independent of ground-truth labels. One possibility is the use of cluster validity indices, like the STR index (Starczewski, 2017) or the Silhouette index (Rousseeuw, 1987). Another possibility is the combined use of several cluster validity indices to optimize the non-parametric models (Saini et al., 2019), as long as it guarantees that queries inside a search task cluster are as close as possible, while search task clusters remain as separated as possible between them.

Another interesting research direction is the use of generative methods for deep clustering (Aljalbout et al., 2018; Min et al., 2018). RDC outperforms existing deep clustering methods by leveraging bidirectional RNNs and self-training to learn query representations. But comparing RDC with generative alternatives, like Variational Deep Embedding (Jiang et al., 2017) or Deep Adversarial Subspace Clustering (Zhou et al., 2018), can give us hints of the impact generative architectures can have on modeling results. If the impact is positive, it is possible to devise ways to incorporate generative alternatives to perform the clustering. Similarly, it is possible to replace the clustering layer in RDC for graph-based clustering alternatives. Doing so allows the simultaneous learning of query representations and the optimization of graphs for clustering. Besides, discrete architectures (Kaiser and Bengio, 2018; Kaiser et al., 2018) for clustering could improve the modeling of search tasks because query logs can have thousands of search tasks, and quantization is already used in extreme classification methods with a large number of classes (Guo et al., 2020).

Moreover, it is widely accepted that users multitask when fulfilling their information needs (Lucchese et al., 2013; Mehrotra and Yilmaz, 2017; Tamine et al., 2020). Most publicly available datasets with timestamps show this behaviour (Section 2.1), except for the the Complex User Search Task Analysis (CUSTA) dataset (Dosso et al., 2020). But why does the CUSTA dataset shows little to no multitasking? There are some hints about this lack of multitasking. First, "task sets" allow users to program their memory to perform a certain task, even if it is not what users normally do. For example, when looking at a book page, a user will normally read the content, but the user can program its memory to instead count the number of f's present on it (Kahneman, 2013). Thus, users can perform all the steps to solve a particular search task in tandem, even though they normally multitask. Second, a continuous train of thought requires effort. So, users should exert self-control to resist the urge to stop the mental effort that a complex task requires (Kahneman, 2013), but if the user interface allows for "cues" like social network notifications or mail access, users will have a harder time keeping focused in a particular search task (Oakley, 2014; Williams et al., 2018). Understanding this lack of multitasking can improve the design of search systems, allowing users to maintain focus, an essential behavior when solving complex search tasks.

Furthermore, conversational information seeking is increasingly becoming a major area of research (Anand et al., 2020; Zamani and Craswell, 2020). Among the many research problems in conversational search, two research questions arise. First, what approaches adapt to evolving user information needs, while at the same time, leverage past interactions with users? As discussed before, mining search query logs allow search pattern modeling, enabling multiple applications for assisting users while they satisfy their information needs. Assisting applications like conversational question suggestion (Rosset et al., 2020) and clarifying question asking (Aliannejadi et al.,

2019) can benefit from learned models. Though foregoing proposed models learn from user interactions, they use an offline training scheme. To update the models, it is necessary to retrain or fine-tune them with the updated set of interactions. However, offline learning is not dynamic. For conversational search, systems should be interactive, dynamically adapting in real-time to evolving user information needs. Conversely, learning from scratch while interacting with users can cause conversational search systems to generate unwanted responses. In such scenarios, users will likely abandon the search system (Zamani and Craswell, 2020). Hence, developing methods that learn when interacting with users in realtime, but take advantage of past interactions, is critical for information seeking in a conversational way.

Second, how to identify if users have experience in the subject matter of the information need they want to fulfill? Experience in a particular subject matter has an impact on the way users satisfy their information needs. Experienced users will use a different way to satisfy their information needs than users who need to learn about the subject matter, particularly when solving complex search tasks (Hearst, 2009; Karanam et al., 2017). Discovering whether the user has experience or not allows search systems to adapt their responses. That adaptation can be instrumental for proposing clarifying questions (Rosset et al., 2020), presenting relevant passages (Chang et al., 2020; Dai and Callan, 2020), or helping inexperienced users explore while carrying out the information seeking process (Dosso et al., 2020; Hearst, 2009). Accordingly, developing approaches to dynamically adapt to evolving users' information needs, as well as identifying the experience of users in the topic of search, can be a crucial part of the evolution of search systems towards conversational information seeking.

References

Ahmad, W. U., Chang, K.-W., and Wang, H. (2019). Context attentive document ranking and query suggestion. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 385–394.

Aliannejadi, M., Zamani, H., Crestani, F., and Croft, W. B. (2019). Asking clarifying questions in open-domain information-seeking conversations. In *Proceedings of the 42nd international acm sigir conference on research and development in information retrieval*, pages 475–484.

Aljalbout, E., Golkov, V., Siddiqui, Y., Strobel, M., and Cremers, D. (2018). Clustering with deep learning: Taxonomy and new methods. *arXiv preprint arXiv:1801.07648*.

Alpaydin, E. (2014). *Introduction to machine learning*. MIT press, Cambridge, MA, US.

Anand, A., Cavedon, L., Joho, H., Sanderson, M., and Stein, B. (2020). Conversational search (Dagstuhl seminar 19461). In *Dagstuhl Reports*, volume 9. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

Angermueller, C., Pärnamaa, T., Parts, L., and Stegle, O. (2016). Deep learning for computational biology. *Molecular systems biology*, 12(7):878.

Arguello, J. and Capra, R. (2016). The effects of aggregated search coherence on search behavior. *ACM Transactions on Information Systems (TOIS)*, 35(1):1–30.

Artetxe, M. and Schwenk, H. (2019). Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *Transactions of the Association for Computational Linguistics*, 7:597–610.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bar-Yossef, Z. and Kraus, N. (2011). Context-sensitive query auto-completion. In *Proceedings of the 20th international conference on World wide web*, pages 107–116.

Bengio, Y. (2009). Learning deep architectures for ai. *Foundations and Trends in Machine Learning*, 2(1):1–127.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48.

Blei, D. M. and Frazier, P. I. (2011). Distance dependent Chinese restaurant processes. *Journal of Machine Learning Research*, 12(8).

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Blundell, C. and Teh, Y. W. (2013). Bayesian hierarchical community discovery. In *Advances in Neural Information Processing Systems*, pages 1601–1609.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Boldi, P., Bonchi, F., Castillo, C., Donato, D., Gionis, A., and Vigna, S. (2008). The query-flow graph: model and applications. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 609–618.

Boldi, P., Bonchi, F., Castillo, C., Donato, D., and Vigna, S. (2009). Query suggestions using query-flow graphs. In *Proceedings of the 2009 workshop on Web Search Click Data*, pages 56–63.

Borisov, A., Markov, I., De Rijke, M., and Serdyukov, P. (2016). A neural click model for web search. In *Proceedings of the 25th International Conference on World Wide Web*, pages 531–541.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *arXiv preprint arXiv:2005.14165.*

Callan, J. (2012). The Lemur project and its ClueWeb12B dataset. In *Invited talk at the SIGIR 2012 Workshop on Open-Source Information Retrieval.*

Cao, H., Jiang, D., Pei, J., Chen, E., and Li, H. (2009). Towards context-aware search by learning a very large variable length hidden markov model from search logs. In *Proceedings of the 18th international conference on World wide web*, pages 191–200.

Carterette, B., Clough, P., Hall, M., Kanoulas, E., and Sanderson, M. (2016). Evaluating retrieval over sessions: The TREC session track 2011-2014. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 685–688. ACM.

Cer, D., Yang, Y., Kong, S.-y., Hua, N., Limtiaco, N., John, R. S., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., et al. (2018). Universal sentence encoder for english. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174.

Chang, J., Wang, L., Meng, G., Xiang, S., and Pan, C. (2017). Deep adaptive image clustering. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5880–5888. IEEE.

Chang, W.-C., Felix, X. Y., Chang, Y.-W., Yang, Y., and Kumar, S. (2020). Pre-training tasks for embedding-based large-scale retrieval. In *International Conference on Learning Representations*.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*.

Chen, Z. and Ji, H. (2010). Graph-based clustering for computational linguistics: A survey. In *Proceedings of the 2010 workshop on Graph-based Methods for Natural Language Processing*, pages 1–9. Association for Computational Linguistics.

Chidambaram, M., Yang, Y., Cer, D., Yuan, S., Sung, Y.-H., Strope, B., and Kurzweil, R. (2018). Learning cross-lingual sentence representations via a multi-task dual-encoder model. *arXiv preprint arXiv:1810.12836*.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Conneau, A. and Lample, G. (2019). Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems*, pages 7059–7069.

Conneau, A., Lample, G., Ranzato, M., Denoyer, L., and Jégou, H. (2017). Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Craswell, N., Campos, D., Mitra, B., Yilmaz, E., and Billerbeck, B. (2020a). ORCAS: 18 million clicked query-document pairs for analyzing search. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*. ACM.

Craswell, N., Mitra, B., Yilmaz, E., Campos, D., and Voorhees, E. M. (2020b). Overview of the TREC 2019 deep learning track. *arXiv preprint arXiv:2003.07820*.

Dai, Z. and Callan, J. (2020). Context-aware term weighting for first stage passage retrieval. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1533–1536.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186.

Diaz, F., Mitra, B., and Craswell, N. (2016). Query expansion with locally-trained word embeddings. *arXiv preprint arXiv:1605.07891*.

Dolz, J., Desrosiers, C., and Ayed, I. B. (2016). 3D fully convolutional networks for subcortical segmentation in MRI: A large-scale study. *arXiv preprint arXiv:1612.03925*, abs/1612.03925.

Dosso, C., Chevalier, A., and Tamine, L. (2020). How to support search activity of users without prior domain knowledge when they are solving learning tasks? In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*, 1st International Workshop on Investigating Learning During Web Search. ACM.

Du, C., Shu, P., and Li, Y. (2018). CA-LSTM: Search task identification with context attention based lstm. In *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1101–1104. ACM.

Dumais, S. T. (2004). Latent semantic analysis. *Annual review of information science and technology*, 38(1):188–230.

Edo-Osagie, O., Lake, I., Edeghere, O., and De La Iglesia, B. (2019). Attention-based recurrent neural networks (RNNs) for short text classification: An application in public health monitoring. In *International Work-Conference on Artificial Neural Networks*, pages 895–911. Springer.

Edunov, S., Ott, M., Auli, M., and Grangier, D. (2018). Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *The 2nd International Conference on Knowledge Discovery and Data Mining*, volume 96, pages 226–231.

Feng, F., Yang, Y., Cer, D., Arivazhagan, N., and Wang, W. (2020). Language-agnostic BERT sentence embedding. *arXiv preprint arXiv:2007.01852*.

Gayo-Avello, D. (2009). A survey on session detection methods in query logs and a proposal for future evaluation. *Information Sciences*, 179(12):1822–1843.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th international conference on artificial intelligence and statistics*, pages 249–256.

Gomes, P., Martins, B., and Cruz, L. (2019). Segmenting user sessions in search engine query logs leveraging word embeddings. In *International Conference on Theory and Practice of Digital Libraries*, pages 185–199. Springer.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press, Cambridge, MA, US.

Grave, E., Bojanowski, P., Gupta, P., Joulin, A., and Mikolov, T. (2018). Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Graves, A. (2012). *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer, Berlin, Heidelberg.

Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Guo, R., Sun, P., Lindgren, E., Geng, Q., Simcha, D., Chern, F., and Kumar, S. (2020). Accelerating large-scale inference with anisotropic vector quantization. In *Proceedings of the 37th International Conference on Machine Learning*.

Guo, X., Gao, L., Liu, X., and Yin, J. (2017). Improved deep embedded clustering with local structure preservation. In *International Joint Conference on Artificial Intelligence*, pages 1753–1759.

Hagen, M., Gomoll, J., Beyer, A., and Stein, B. (2013). From search session detection to search mission detection. In *Proceedings of the 10th Conference on Open Research Areas in Information Retrieval*, pages 85–92.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Hearst, M. (2009). *Search user interfaces*. Cambridge University Press, Cambridge, CB2 8BS, UK.

Iwasaki, M. and Miyazaki, D. (2018). Optimization of indexing based on k-nearest neighbor graph for proximity search in high-dimensional data. *arXiv preprint arXiv:1810.07355*.

Jiang, Z., Zheng, Y., Tan, H., Tang, B., and Zhou, H. (2017). Variational deep embedding: an unsupervised and generative approach to clustering. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 1965–1972.

Joshi, P., Santy, S., Budhiraja, A., Bali, K., and Choudhury, M. (2020). The state and fate of linguistic diversity and inclusion in the NLP world. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6282–6293.

Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., et al. (2017). In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, pages 1–12. ACM.

Jozefowicz, R., Zaremba, W., and Sutskever, I. (2015). An empirical exploration of recurrent network architectures. In *International Conference on Machine Learning*, pages 2342–2350.

Kahneman, D. (2013). *Thinking, fast and slow*. Farrar, Straus and Giroux, New York, NY, US.

Kaiser, Ł. and Bengio, S. (2018). Discrete autoencoders for sequence models. *arXiv preprint arXiv:1801.09797*.

Kaiser, L., Bengio, S., Roy, A., Vaswani, A., Parmar, N., Uszkoreit, J., and Shazeer, N. (2018). Fast decoding in sequence models using discrete latent variables. In *International Conference on Machine Learning*, pages 2390–2399. PMLR.

Karamanolakis, G., Mukherjee, S., Zheng, G., and Awadallah, A. H. (2021). Self-training with weak supervision. *arXiv preprint arXiv:2104.05514*.

Karanam, S., Jorge-Botana, G., Olmos, R., and van Oostendorp, H. (2017). The role of domain knowledge in cognitive modeling of information search. *Information Retrieval Journal*, 20(5):456–479.

Khan, S., Naseer, M., Hayat, M., Zamir, S. W., Khan, F. S., and Shah, M. (2021). Transformers in vision: A survey. *arXiv preprint arXiv:2101.01169*.

Khatri, C., Goel, R., Hedayatnia, B., Metanillou, A., Venkatesh, A., Gabriel, R., and Mandal, A. (2018). Contextual topic modeling for dialog systems. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 892–899. IEEE.

Kiela, D., Wang, C., and Cho, K. (2018). Dynamic meta-embeddings for improved sentence representations. *arXiv preprint arXiv:1804.07983*.

Kim, Y. (2014). Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kotov, A., Bennett, P. N., White, R. W., Dumais, S. T., and Teevan, J. (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*, pages 5–14.

Kudo, T. and Richardson, J. (2018). SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71.

Lample, G., Conneau, A., Denoyer, L., and Ranzato, M. (2017). Unsupervised machine translation using monolingual corpora only. *arXiv preprint arXiv:1711.00043*.

Li, L., Deng, H., Dong, A., Chang, Y., Baeza-Yates, R., and Zha, H. (2017). Exploring query auto-completion and click logs for contextual-aware web search and query suggestion. In *Proceedings of the 26th International Conference on World Wide Web*, pages 539–548.

Li, L., Deng, H., Dong, A., Chang, Y., and Zha, H. (2014a). Identifying and labeling search tasks via query-based Hawkes processes. In *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 731–740.

Li, Y., Dong, A., Wang, H., Deng, H., Chang, Y., and Zhai, C. (2014b). A two-dimensional click model for query auto-completion. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 455–464.

Lin, K., Yang, H.-F., Hsiao, J.-H., and Chen, C.-S. (2015). Deep learning of binary hash codes for fast image retrieval. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 27–35.

Lipton, Z. C., Berkowitz, J., and Elkan, C. (2015). A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*.

Liu, X. (2017). Deep recurrent neural network for protein function prediction from sequence. *arXiv preprint arXiv:1701.08318*.

Liu, Y., Miao, J., Zhang, M., Ma, S., and Ru, L. (2011). How do users describe their information need: Query recommendation based on snippet click model. *Expert Systems with Applications*, 38(11):13847–13856.

Lucchese, C., Orlando, S., Perego, R., Silvestri, F., and Tolomei, G. (2011). Identifying task-based sessions in search engine query logs. In *Proceedings of the 4th ACM international conference on Web search and data mining*, pages 277–286. ACM.

Lucchese, C., Orlando, S., Perego, R., Silvestri, F., and Tolomei, G. (2013). Discovering tasks from search engine query logs. *ACM Transactions on Information Systems (TOIS)*, 31(3):1–43.

Lugo, L., Moreno, J. G., and Hubert, G. (2020a). A multilingual approach for unsupervised search task identification. In *The 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM.

Lugo, L., Moreno, J. G., and Hubert, G. (2020b). Segmenting search query logs by learning to detect search task boundaries. In *The 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM.

Lugo, L., Moreno, J. G., and Hubert, G. (2021a). Extracting search tasks from query logs using a recurrent deep clustering architecture. In *Proceedings of the 43rd European Conference on Information Retrieval*. Springer.

Lugo, L., Moreno, J. G., and Hubert, G. (2021b). Extraction des tâches de recherche dans des journaux de requêtes à l'aide d'une architecture de regroupement profond récurrent. In *La 17ème édition de la Conférence en Recherche d'Information et Applications CORIA-RJCRI*. ARIA.

Lugo, L., Moreno, J. G., and Hubert, G. (2021c). Modeling user search tasks with a language-agnostic unsupervised approach. In *Proceedings of the 43rd European Conference on Information Retrieval*. Springer.

Luo, Y., Chen, Z., Hershey, J. R., Le Roux, J., and Mesgarani, N. (2017). Deep clustering and conventional networks for music separation: Stronger together. In *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 61–65. IEEE.

Luong, M.-T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Malkov, Y. A. and Yashunin, D. A. (2018). Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*.

Manikonda, L., Deotale, A., and Kambhampati, S. (2018). What's up with privacy? user preferences and privacy concerns in intelligent personal assistants. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, page 229–235, New York, NY, USA. Association for Computing Machinery.

Marivate, V. and Sefara, T. (2019). Improving short text classification through global augmentation methods. *arXiv preprint arXiv:1907.03752*.

Martens, J. and Sutskever, I. (2011). Learning recurrent neural networks with hessian-free optimization. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1033–1040. Citeseer.

Mehrotra, R. (2015). Topics, tasks & beyond: Learning representations for personalization. In *Proceedings of the 8th ACM International Conference on Web Search and Data Mining*, pages 459–464. ACM.

Mehrotra, R., Awadallah, A. H., and Yilmaz, E. (2020). Special issue on learning from user interactions. *Information Retrieval Journal*, 23(6):525–527.

Mehrotra, R., Bhattacharya, P., and Yilmaz, E. (2016). Deconstructing complex search tasks: a Bayesian nonparametric approach for extracting subtasks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 599–605.

Mehrotra, R. and Yilmaz, E. (2017). Extracting hierarchies of search tasks & subtasks via a Bayesian nonparametric approach. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 285–294. ACM.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Min, E., Guo, X., Liu, Q., Zhang, G., Cui, J., and Long, J. (2018). A survey of clustering with deep learning: From the perspective of network architecture. *IEEE Access*, 6:39501–39514.

Min, S., Lee, B., and Yoon, S. (2016). Deep learning in bioinformatics. *arXiv preprint arXiv:1603.06430*, abs/1603.06430.

Mitchell, M. (2019). *Artificial Intelligence: A Guide for Thinking Humans*. Farrar, Straus and Giroux, New York, NY, US.

Moreno, J. G. (2018). Point symmetry-based deep clustering. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1747–1750. ACM.

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT Press, Cambridge, MA, US.

Murphy, K. P. (2021). *Probabilistic Machine Learning: An introduction*. MIT Press, Cambridge, MA, US.

Murtagh, F. and Legendre, P. (2014). Ward's hierarchical agglomerative clustering method: which algorithms implement Ward's criterion? *Journal of Classification*, 31(3):274–295.

Nalisnick, E., Mitra, B., Craswell, N., and Caruana, R. (2016). Improving document ranking with dual word embeddings. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 83–84.

Nascimento, M. C. and De Carvalho, A. C. (2011). Spectral methods for graph clustering–a survey. *European Journal of Operational Research*, 211(2):221–231.

Nie, F., Wang, X., Jordan, M., and Huang, H. (2016). The constrained laplacian rank algorithm for graph-based clustering. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30.

Oakley, B. A. (2014). *A mind for numbers: How to excel at math and science (even if you flunked algebra)*. TarcherPerigree, New York, NY, US.

Pass, G., Chowdhury, A., and Torgeson, C. (2006). A picture of search. In *InfoScale*, volume 152, page 1.

Pennington, J., Socher, R., and Manning, C. D. (2014). GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding by generative pre-training.

Rajaraman, A. and Ullman, J. D. (2011). *Mining of Massive Datasets*. Cambridge University Press, New York, NY, USA.

Rakib, M. R. H., Jankowska, M., Zeh, N., and Milios, E. (2018). Improving short text clustering by similarity matrix sparsification. In *Proceedings of the ACM Symposium on Document Engineering 2018*, page 50. ACM.

Rampasek, L. and Goldenberg, A. (2016). Tensorflow: Biology's gateway to deep learning? *Cell systems*, 2(1):12–14.

Rekabsaz, N., Lupu, M., Hanbury, A., and Zamani, H. (2017). Word embedding causes topic shifting; exploit global context! In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1105–1108. ACM.

Robertson, S. (2004). Understanding inverse document frequency: on theoretical arguments for IDF. *Journal of documentation*, 60(5):503–520.

Rosset, C., Xiong, C., Song, X., Campos, D., Craswell, N., Tiwary, S., and Bennett, P. (2020). Leading conversational search by suggesting useful questions. In *Proceedings of The Web Conference 2020*, pages 1160–1170.

Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65.

Saini, N., Saha, S., and Bhattacharyya, P. (2019). Automatic scientific document clustering using self-organized multi-objective differential evolution. *Cognitive Computation*, 11(2):271–293.

Sen, P., Ganguly, D., and Jones, G. (2018). Tempo-lexical context driven word embedding for cross-session search task extraction. In *Proceedings of* *the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 283–292.

Severyn, A. and Moschitti, A. (2015). Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 373–382. ACM.

Stahl, B. C. and Wright, D. (2018). Ethics and privacy in AI and big data: Implementing responsible research and innovation. *IEEE Security Privacy*, 16(3):26–33.

Starczewski, A. (2017). A new validity index for crisp clusters. *Pattern Analysis and Applications*, 20(3):687–700.

Sun, A. and Lou, C.-H. (2014). Towards context-aware search with right click. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 847–850.

Tamine, L., Melgarejo, J. L., and Pinel-Sauvagnat, K. (2020). What can task teach us about query reformulations? In *European Conference on Information Retrieval*, pages 636–650. Springer.

Tan, P.-N., Steinbach, M., Karpatne, A., and Kumar, V. (2018). *Introduction to Data Mining*. Pearson Education, London,UK, second edition.

Tay, Y., Dehghani, M., Abnar, S., Shen, Y., Bahri, D., Pham, P., Rao, J., Yang, L., Ruder, S., and Metzler, D. (2020). Long range arena: A benchmark for efficient transformers. *arXiv preprint arXiv:2011.04006*.

Thomas, P., McDuff, D., Czerwinski, M., and Craswell, N. (2020). Expressions of style in information seeking conversation with an agent. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1171–1180.

Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Venkatesh, A., Khatri, C., Ram, A., Guo, F., Gabriel, R., Nagar, A., Prasad, R., Cheng, M., Hedayatnia, B., Metallinou, A., et al. (2018). On evaluating and comparing open domain dialog systems. *arXiv preprint arXiv:1801.03625*.

Vinh, N. X., Epps, J., and Bailey, J. (2010). Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 11(Oct):2837–2854.

Völske, M., Fatehifar, E., Stein, B., and Hagen, M. (2019). Query-task mapping. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 969–972.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. (2019a). GLUE: A multi-task benchmark and analysis platform for natural language understanding. *Proceedings of ICLR*.

Wang, H., Song, Y., Chang, M.-W., He, X., White, R. W., and Chu, W. (2013a). Learning to extract cross-session search tasks. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1353–1364. ACM.

Wang, H., Zhai, C., Dong, A., and Chang, Y. (2013b). Content-aware click modeling. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1365–1376.

Wang, P., Xu, B., Xu, J., Tian, G., Liu, C.-L., and Hao, H. (2016). Semantic expansion using word embedding clustering and convolutional neural network for improving short text classification. *Neurocomputing*, 174:806–814.

Wang, R., Nie, F., Wang, Z., He, F., and Li, X. (2019b). Scalable graph-based clustering with nonnegative relaxation for large hyperspectral image. *IEEE Transactions on Geoscience and Remote Sensing*, 57(10):7352–7364.

Wang, Z.-Q., Le Roux, J., and Hershey, J. R. (2018). Alternative objective functions for deep clustering. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 686–690. IEEE.

Ward Jr, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301):236–244.

Williams, A. C., Kaur, H., Mark, G., Thompson, A. L., Iqbal, S. T., and Teevan, J. (2018). Supporting workplace detachment and reattachment with conversational intelligence. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–13.

Xie, J., Girshick, R., and Farhadi, A. (2016). Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pages 478–487.

Xie, Q., Dai, Z., Hovy, E., Luong, T., and Le, Q. (2020). Unsupervised data augmentation for consistency training. *Advances in Neural Information Processing Systems*, 33.

Yang, Y., Abrego, G. H., Yuan, S., Guo, M., Shen, Q., Cer, D., Sung, Y.-H., Strope, B., and Kurzweil, R. (2019). Improving multilingual sentence embedding using bi-directional dual encoder with additive margin softmax. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 5370–5378. AAAI Press.

Yang, Y., Cer, D., Ahmad, A., Guo, M., Law, J., Constant, N., Abrego, G. H., Yuan, S., Tar, C., Sung, Y.-h., Strope, B., and Kurzweil, R. (2020). Multilingual universal sentence encoder for semantic retrieval. In *Proceedings of the 58th Annual Meeting of the ACL: System Demonstrations*, pages 87–94. ACL.

Zamani, H. and Craswell, N. (2020). Macaw: An extensible conversational information seeking platform. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2193–2196.

Zamani, H. and Croft, W. B. (2016a). Embedding-based query language models. In *Proceedings of the 2016 ACM international conference on the theory of information retrieval*, pages 147–156. ACM.

Zamani, H. and Croft, W. B. (2016b). Estimating embedding vectors for queries. In *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval*, pages 123–132. ACM.

Zamani, H. and Croft, W. B. (2017). Relevance-based word embedding. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 505–514. ACM.

Zamani, H., Mitra, B., Chen, E., Lueck, G., Diaz, F., Bennett, P. N., Craswell, N., and Dumais, S. T. (2020). Analyzing and learning from user interactions for search clarification. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1181–1190.

Zhang, H., Song, X., Xiong, C., Rosset, C., Bennett, P., Craswell, N., and Tiwary, S. (2019). Generic intent representation in web search. In *The 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM.

Zhang, Y., Jin, R., and Zhou, Z.-H. (2010). Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics*, 1(1-4):43–52.

Zhao, Y., Zhang, Y., Zhang, B., Gao, K., and Li, P. (2018). Recommending queries by extracting thematic experiences from complex search tasks. *Entropy*, 20(6):459.

Zhou, P., Hou, Y., and Feng, J. (2018). Deep adversarial subspace clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1596–1604.

Zoph, B., Ghiasi, G., Lin, T.-Y., Cui, Y., Liu, H., Cubuk, E. D., and Le, Q. (2020). Rethinking pre-training and self-training. *Advances in Neural Information Processing Systems*, 33.