Title
Learning from Sparse and Deficient Data and its Applications

Author
Li, Ruirui

Publication Date
2019

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

Los Angeles

Learning from Sparse and Deficient Data and its Applications

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Computer Science

by

Ruirui Li

2019

ABSTRACT OF THE DISSERTATION

Learning from Sparse and Deficient Data and its Applications

by

Ruirui Li
Doctor of Philosophy in Computer Science
University of California, Los Angeles, 2019
Professor Wei Wang, Chair

Deep learning models have been attracting substantial attention in the last few years as they successfully demonstrated remarkable performance on different tasks (e.g., classification and ranking) in various fields. The success of most deep learning models significantly depends on massive training data as these models inherently work by memorizing or distinguishing massive training instances in the data. However, such a huge amount of training data may not be available or accessible all the time due to privacy issues, user experience concerns, or corporation constraints, etc. This leads to the data deficiency issue, which corresponds to the scenarios where we have just a few training instances to accomplish a task. Even though sufficient training data could be acquired, sometimes the data could be very sparse due to the large base of elements (e.g. users and items) in the dataset. This results in the data sparsity issue, which corresponds to the scenarios where we have very limited training instances to pinpoint the characteristics of each element in the data. Both the data deficiency and data sparsity issues limit the expressiveness of strong model capacities of the deep learning models and generally lead to inferior performances on a variety of tasks.

In this dissertation, we propose several deep learning frameworks to compensate for the data deficiency and data sparsity in the context of three concrete applications, i.e., customer recommendation in location-based social networks, query recommendation in search engines, and automatic speaker recognition. The methodologies presented in these frame-

works span different research areas, including geographical influence modeling on location data, automatic data augmentation via adversarial training, comprehensive instance utilization through metric-learning-based few-shot learning, and knowledge transfer via gradient-based meta-learning. As a result, these methodologies not only tackle specific challenges in the applications mentioned above but also shed light on other relevant applications.

The dissertation of Ruirui Li is approved.

Carlo Zaniolo

Douglas Stott Parker

Yingnian Wu

Wei Wang, Committee Chair

University of California, Los Angeles

2019

To my parents,

for their endless love and supports.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

This work would not have been possible without the continuous support of my advisor, family, and friends. First, I would like to extend my sincere gratitude to my advisor, Prof. Wei Wang, who offered me the precious opportunity to continue my PhD research at UCLA. She proffered me unending support, inspiring me to play with new ideas and encouraging me to explore the direction which I am interested, for which I am heartily thankful. It was her inspiration, encouragement, and expectation that kept me focused and motivated. Second, I would also like to include my gratitude to my family. I was extremely fortunate to have parents who loved me unconditionally and supported me bottomlessly. I am truly indebted to my parents for their self-giving and untold love. This dissertation is dedicated to my family. Finally, I would like to say thanks to all my friends and colleagues at UCLA, in particular, Chelsea Ju, Jyun-Yu Jiang, and all members in the SCAI lab. They inspired my research here, made my life free of solitude and monotonicity, and left me with precious memories.

VITA

2006 – 2010	B.Sc. (Computer Science), Nanjing University, Nanjing, China
2010 – 2013	M.S. (Computer Science), University of Hong Kong, Hong Kong
2015 – 2015	Research Intern, AT&T Research Lab, Bedminster, New Jersey
2016 – 2016	Software Engineer Intern, Google, Irvine, California
2017 – 2017	Software Engineer Intern, Google, Manhattan, New York
2018 – 2018	Research Intern, Yahoo Research Lab, San Jose, California
2013 – 2019	Graduate Student Researcher, Computer Science Department, UCLA

PUBLICATIONS

Ruirui Li, Jyun-Yu Jiang, Jiahao Liu, Chu-Cheng Hsieh, and Wei Wang. Automatic Speaker Recognition with Limited Data. In *Proceedings of The 13th ACM International Conference on Web Search and Data Mining* (WSDM 2020).

Ruirui Li, Xian Wu, and Wei Wang. Adversarial Learning to Compare: Self-Attentive Prospective Customer Recommendation in Location-based Social Networks. In *Proceedings of The 13th ACM International Conference on Web Search and Data Mining* (WSDM 2020).

Ruirui Li, Jyun-Yu Jiang, Xian Wu, Hongda Mao, Chu-Cheng Hsieh and Wei Wang. Bridging Mixture Density Networks with Meta-learning for Automatic Speaker Identification. In *Proceedings of The 45th International Conference on Acoustics, Speech, and Signal Process-*

ing (ICASSP 2020, under review).

Ruirui Li, Xiusi Chen, Xian Wu, and Wei Wang. Few-Shot Learning for New User Recommendation in Location-based Social Networks. In *Proceedings of The 31th World Wide Web Conference* (WWW 2020, under review).

Ruirui Li, Liangda Li, Xian Wu, Yunhong Zhou, and Wei Wang. Click Feedback-Aware Query Recommendation Using Adversarial Examples. In *Proceedings of The 30th World Wide Web Conference* (WWW 2019).

Ruirui Li, Jyun-Yu Jiang, Chelsea J.-T. Ju, and Wei Wang. CORALS: Who are My Potential New Customers? Tapping into the Wisdom of Customers' Decisions. In *Proceedings of The 12th ACM International Conference on Web Search and Data Mining* (WSDM 2019).

Ruirui Li, Jyun-Yu Jiang, Chelsea J.-T. Ju, Cheryl Flynn, Wen-Ling Hsu, Jia Wang, Wei Wang, and Tan Xu. Enhancing Response Generation Using Chat Flow Identification. In *Proceedings of the Conversational AI and its Applications Workshop at KDD* (CAI Workshop SIGKDD 2018).

Zijun Xue, **Ruirui Li**, and Mingda Li. Recent Progress in Conversational AI. In *Proceedings of the Conversational AI and its Applications Workshop at KDD* (CAI Workshop SIGKDD 2018).

Chelsea J.-T. Ju, Jyun-Yu Jiang, **Ruirui Li**, Zeyu Li and Wei Wang. TahcoRoll: An Efficient Approach for Signature Profiling in Genomic Data through Variable-Length k-mers. *bioRxiv*, 2017. pre-print.

Chelsea J.-T. Ju, **Ruirui Li**, Zhengliang Wu, Jyun-Yu Jiang, Zhao Yang and Wei Wang.

Fleximer: Accurate Quantification of RNA-Seq via Variable-Length k-mers. In *Proceedings of The 8th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics* (ACMBCB 2017).

Ruirui Li, Xinxin Huang, Shuo Song, Jia Wang and Wei Wang. Towards Customer Trouble Tickets Resolution Automation In Large Cellular Services. In *Proceedings of 22nd International Conference on Mobile Computing and Networking* (Mobicom 2016).

Liuli Chen, Chelsea J.-T. Ju, **Ruirui Li**, Wenchao Yu, and Wei Wang. Skimdiff: Transcript-level Differential Analysis of RNA-Seq Data. In *Proceedings of The 6th International Conference on Bioinformatics Models, Methods and Algorithms* (Bioinformatics 2015).

Ruirui Li, and Wei Wang. REAFUM: Representative Approximate Frequent Subgraph Mining. In *Proceedings of The International Conference on Data Mining* (SDM 2015).

Ruirui Li, Ben Kao, Bin Bi, Reynold Cheng, and Eric Lo. DQR: A Probabilistic Approach To Diversified Query Recommendation. In *Proceedings of Tge 21st ACM International Conference on Information and Knowledge Management* (CIKM 2012).

CHAPTER 1

Introduction

The advances of deep learning technologies have a revolutionary impact on a variety of fields, as these deep learning technologies, born with strong modeling capacity, have compelling power in accomplishing various tasks. The expressiveness of their compelling modeling power largely depends on the support of massive training data. However, such massive data may not be available or accessible all the time due to data deficiency and data sparsity issues in practice. The data deficiency issue corresponds to the scenarios when there is a shortage of labeled training data. For example, this generally happens when we are serving new users in recommendation or speaker recognition tasks, as a large portion of new users usually has a very limited number of training instances. The data deficiency issue is also known as few-shot learning. The data sparsity refers to the scenarios where the density of the interactions between elements in the data is low, as the base of the elements could be very large. Therefore, there are very few interactions available to mine the characteristics of each element, which may degrade the performance of downstream applications. Dealing with the deficient and sparse data is often considered a daunting task as there is a limited number of related data records/interactions we can utilize to conduct modeling and analysis. In this dissertation, we study three different research problems in mining deficient and sparse data. The first two problems focus on leveraging auxiliary features, and adversarial training to solve recommendation tasks. The third problem focuses on analyzing acoustic characteristics of speakers and achieving effective speaker identification from limited data. For each problem, we discuss the limitations of existing approaches and propose new methods to address these challenges.

1.1 Scope of the Research

Recommendation has attracted substantial attention from researchers and has revolutionized the e-commerce industry. Various recommender systems have been developed to facilitate the matching between users with appropriate items (e.g., products, services or businesses). Two recommendation problems, prospective new user recommendation in location-based social networks and query suggestion on search engines, are summarized below. For the third research problem, we discuss the challenges and potential of recognizing speakers from limited training data.

- **Prospective new user recommendation in location-based social networks.** The proliferation of GPS-enabled devices, such as smartphones, establishes the prosperity of location-based social networks (LBSNs), which results in a tremendous amount of user check-ins. These user check-ins bring in great opportunities to understand users' preferences and facilitate matching between users and local businesses. However, the user check-ins are extremely sparse due to the huge user and business bases, which makes matching a daunting task. To make accurate recommendations in LBSNs, it is critical to have effective methods that can embrace relevant auxiliary information as extra guidance and adopt appropriate techniques to fully utilize the sparse check-in data.

- **Query suggestion on search engines.** Search engine users always endeavor to formulate proper search queries during online search. To help users accurately express their information need during search, search engines are equipped with query suggestions to refine users' follow-up search queries. The success of a query suggestion system significantly relies on whether we can understand and model user search intent accurately [1, 2]. However, user search queries are generally very short, typically with only one or two key keywords each [3]. Short queries lead to two issues. First, they are often ambiguous. Ambiguity weakens the expressiveness of queries because the search engine may not get the users' hidden search intents. This causes poor retrieval

results. The second issue is that short queries are often not specific enough. Although there is no ambiguity in the meaning of a search query, it is too general a query for the search engine to pinpoint the specific information in which the user is interested. In order to model the search intent accurately, existing query suggestion approaches usually mine search sessions and extract search query sequences as context information to aid recommendations [4–6]. However, very few take the user feedback, i.e., user click information, into account. Therefore, a user feedback-aware query suggestion system is needed. It is also essential to have an approach that can generalize well to all search queries, as many queries are tail queries, which only occur a few times in the search log.

- **Automatic speaker recognition.** Automatic speaker recognition (ASR) is a stepping-stone technology towards semantic multimedia understanding, and benefits versatile downstream applications. Most existing neural network-based ASR methods can achieve excellent recognition performance with the support of sufficient training data [7–11]. However, such sufficient training data may not be available or accessible for every user, especially for new users. In practice, a large portion of users usually has a very limited number of training instances and such data deficiency generally leads to inferior recognition performance, especially when serving new users. As a consequence, the lack of training data prevents ASR systems from accurately learning users' acoustic biometrics, jeopardizes the downstream applications, and eventually impairs user experience. Evidently, an efficient mechanism is needed to capture the complicated acoustic characteristics of users and distinguish them as accurately as possible even when facing the shortage of training data.

1.2 Contributions

In this dissertation, we emphasize the importance of the research problems mentioned above, and identify the specific challenges falling within each research problem. We propose a series

of deep learning methods to tackle these challenges.

The first research issue comes from recommending potential new users for local businesses in location-based social networks based on user check-ins. The user check-ins are highly sparse due to the large bases of users and businesses, which poses the major challenge for learning users' preferences and businesses' characteristics. To alleviate the data sparsity, we propose two approaches, i.e., *CORALS* and *SEATLE*. In *CORALS*, we strive to incorporate auxiliary features, which can provide extra guidance in addition to the user check-in records. More specifically, we propose a unified approach to learn the matching between users and businesses, which not only considers users' preferences but also incorporates geographical influence and reputation influence when making recommendations. The geographical influence quantifies how easily a user can check in at a local business based on the user's historical check-in trajectory and the location of the target business, while the reputation influence gauges the impact of the online reviews regarding the target business towards the user. In *SEATLE*, we further decompose geographical influence into geographical convenience and geographical dependency. Geographical convenience measures the relative transportation effort of a check-in from a user to a business while geographical dependency models the geographical influence among businesses, which makes the proposed approach neighborhood-aware. Moreover, metric-learning-based few-shot learning is applied to fully utilize the sparse check-in data. Results show that *CORALS* and *SEATLE* are able to recommend customers for local businesses more accurately, especially for tail businesses, that have very few user check-ins.

The second research problem addresses the caveats of offering query suggestions on search engines. User search queries are generally very short. Short queries are usually ambiguous and too general. To gather more information and model users' search intent more accurately, we propose a novel framework, *CFAN*, modeling both a user' search query sequence and click sequence to learn to represent his/her search intent. Search query sequence modeling allows us to capture a user's previous search queries in addition to the last search query while click sequence modeling endows our approach with the capability of being feedback-aware. In ad-

dition, adversarial examples are dynamically constructed to help build a robust recommender system, which is resistant to nuisance perturbations and achieves out-of-distribution generalization. Experiments, conducted on the search log collected from a real-world commercial search engine (i,e., Yahoo), show that *CFAN* can provide more favorable suggestions.

The third work focuses on modeling users' acoustic prints to achieve automatic speaker identification. We present two deep learning approaches, i.e., *AFEASI* and *MDNML*. *AFEASI* utilizes metric-learning-based few-shot learning and adversarial training to conduct speaker identification with limited training data. *MDNML* applies mixture density networks to construct users' acoustic profiles and relies on gradient-based few-shot learning to perform speaker identification with the focus of serving new users. Results demonstrate that *AFEASI* and *MDNML* can achieve superior recognition performances when facing a shortage of labeled training data.

1.3 Overview

The rest of the dissertation is organized as follows: Chapter 2 summarizes the relevant works for each research problem. Chapters 3 and 4 describe our methods on customer recommendation for local businesses in location-based social networks. Chapter 5 presents our work on query recommendation on search engines. Chapters 6 and 7 discuss the two of our works on automatic speaker recognition. Chapter 8 summarizes all the datasets used in different projects. Chapter 9 shows the experimental results of different projects in this dissertation. Chapter 10 concludes this dissertation with a summary of our work and the plan of future extensions.

CHAPTER 2

Related Work

2.1 Recommendation in LBSNs

To address the check-in sparsity issue, various ancillary information is incorporated when building recommendation models, such as POI popularity, social influence, temporal patterns, textual and visual contents [12–22]. In this part, we focus on geographical influence-oriented works.

To leverage geographical influence to improve recommendation performances, [23] first detect multiple centers for each customer based on their check-in histories. Recommendations are made by referring to the distance between the location of the business and the nearest customer center. In [13], geographical influence is modeled by a power-law distribution between the check-in probability and the pair-wise distance of two check-ins. [24, 25] utilize the kernel density estimation to study customers' check-ins and avoid employing a specific distribution. [26] exploits geographical neighborhood information by assuming that customers have similar preferences on neighboring POIs, and POIs in the same region may share similar user preferences. PACE [27] explores the use of deep neural networks to learn location embeddings and user preferences over POIs. In particular, a context graph is used to model the heterogeneous and complex feature information to address the data sparsity issue. SAE-NAD [28] applies stacked auto-encoders to provide POI recommendations. The employed auto-encoder aims to capture the non-linear and non-trivial relationships between users and POIs, and enables more complex data representations in the latent space. APOIR [29] learns the underlying check-in distribution in an adversarial manner by simultaneously training two synergistic components, i.e. a recommender and a discriminator. These two components are

co-trained alternatively to make POI recommendations.

2.2 Query Recommendation

The greatest challenge in query suggestion is to accurately understand users' underlying search intent and formulate follow-up queries which can better articulate their information needs. To tackle the challenge, various studies tap into the "wisdom of crowds" by mining search sessions.

Follow-up queries in the same search sessions tend to be semantically relevant and more expressive; therefore, they can be adopted as suggestions to help users pinpoint their information needs. The rationale behind it is that web users usually won't shift their information needs dramatically within a short period of time. Based on this assumption, various approaches are proposed. [30] models the session information into a Query Flow Graph (QFG). QFG connects queries based on their co-occurrences in search sessions. Query recommendations are generated by conducting a random walk with restarts on QFG. The Term Query Graph enriches the QFG by incorporating query term nodes. The connections between query term nodes and query nodes smooth the query flow and enhance query suggestion for tail queries [31].

The semantic relatedness between queries can also be mined by comparing their follow-up clicks. Follow-up clicks provide extra information to understand users' actual information needs in addition to the submitted queries. To utilize the click information, a Query-Click bipartite graph is widely constructed [32]. Queries are represented as vertices in the Query-Click bipartite graph and an edge connecting two queries is weighted by a transition probability, which is related to the similarity of the two queries. A hitting time algorithm, which performs a random walk, is then executed on the graph. The hitting time is the expected number of steps it takes to reach a query vertex from a starting query vertex. In [33], given an input query, queries with the smallest hitting times are recommended.

To model user search intents accurately, context-aware query suggestion methods are

proposed [34–37]. In context-aware query suggestion, the whole search query sequence is utilized to generate suggested queries instead of the very last query. [34] constructs a suffix tree on query concept levels to capture contexts and make query suggestions. The context, represented by query transitions, is modeled by a mixture variable memory Markov model in [35]. [36] models syntactic reformulations based on predefined reformulation strategies. A well-established ontology can also be leveraged to learn semantic reformulations in context [37]. Context-aware query suggestion considers more user searches in the session; therefore, it models users' information needs more accurately.

Deep neural networks have also been applied to query suggestion and demonstrated excellent performances [4–6, 38]. HRED [5], as the first study, employs a hierarchical encoder-decoder model to achieve context-aware query suggestion. The encoder first encodes query terms into query-level embeddings and query-level embeddings are further utilized to construct session-level embeddings. The decoder takes the session-level embeddings as inputs and generates suggested queries. [6] introduces the copy mechanism in addition to employing the encoder-decoder framework to generate suggested queries. The copy mechanism allows copying terms from user input queries rather than completely relying on the decoder to generate suggested queries. [38] employs a feedback memory network to model titles of user clicked URLs. The suggestions are made based on the titles of clicked URLs together with the search query. Reformulation inference network (RIN), proposed in [4], explicitly incorporates query reformulations during training. In addition, multi-task learning is applied to learn both discriminative and generative query suggestions.

Unfortunately, most existing works do not consider user-clicked suggestions when modeling user search intent. Especially for the state-of-the-art RNN-based methods [4–6, 38], all of them do not incorporate any types of clicks except [38]. The method, *CFAN* proposed in the thesis, differs from these works in that previously-clicked suggestions are explicitly incorporated as feedback. In addition, adversarial examples are dynamically constructed during the training of *CFAN*, which not only significantly improve the robustness of the model but also dramatically enhance the generic ranking performance.

2.3 Automatic Speaker Recognition

Most state-of-the-art solutions are based on the i-vector representation of speech segments [39], which contributed to significant improvements over the Gaussian Mixture Model-Universal Background Models (GMM-UBMs) [40]. Deep learning has shown remarkable success in speaker identification tasks recently. Deep speaker [10] takes filter bank coefficients as inputs, utilizes residual networks to extract audio embeddings, and employs triplet loss as the loss function to optimize the neural network. VGGVox [11] takes spectrograms as inputs. CNN based residual network is designed to extract audio embeddings. Contrastive loss is employed to optimize the training pairs in the network with pre-training using softmax classification. However, the number of training pairs can grow quadratically with the size of the dataset and elaborate pair selection heuristics are needed to make the training on large datasets feasible. Another Resnet-based model uses additive margin softmax [41] classification loss to improve the recognition accuracy in [7] and [8]. TristouNet [42] applies triplet loss [43] to learn speaker recognition. SincNet [9] utilizes convoluational neural networks to learn speaker recognition from raw audios. [44] proposes a generalized end-to-end (GE2E) loss and works on text-dependent speaker verification. The GE2E loss function updates the network in a way that emphasizes training instances that are difficult to verify at each step of the training process. [45] is the most relevant work, which leverages a prototypical network to conduct speaker recognition from limited training data.

The proposed method in this thesis, *AFEASI*, differs from the above work by focusing on speaker identification by learning from limited instances. *AFEASI* leverages metric-learning few-shot learning to achieve competitive performance with limited training instances. In addition, adversarial training is adopted to improve the generalization and robustness of the identification model.

2.4 Adversarial Examples and Learning

Training with adversarial examples has attracted significant attention in past few years because it not only improves the robustness but also the performance of the model. Adversarial examples are examples that are created by making small perturbations to the model input, designed to significantly increase the loss incurred by a machine learning model [46,47]. Most existing machine learning models are highly vulnerable to such adversarial examples [48]. Generic regularization strategies such as dropout, pretraining, and model averaging do not confer a significant reduction in such vulnerability [47]. In order to train robust models, adversarial training is proposed to train models that can correctly classify both unmodified examples and adversarial examples in classification tasks. Adversarial training not only improves the robustness to adversarial examples, but also enhances generalization performance for original examples.

Adversarial training is a novel technique for training models to improve robustness to small, approximately worst case perturbations. The adversarial training process can be viewed as minimizing the worst case error when the training instances are perturbed by an adversary. It can be interpreted as learning to play an adversarial game, trying to minimize an upper bound on the expected loss over noisy instances. Adversarial training can also be viewed as a form of active learning, where the model actively requests labels on new instances. In the case of adversarial training, the new instances are constructed by introducing perturbations to existing instances. In addition, the human labelers are replaced with a heuristic labeler that copies labels from nearby instances.

In this work, we focus on recent adversarial example works in natural language processing. Adversarial examples are generated directly by manipulating the input text in [49–51]. [49] investigates adversarial examples in the task of reading comprehension. To generate adversarial examples, distracting sentences are added to the end of paragraphs. The distracting sentences are either grammatical sentences, which are both similar to the questions and consistent to the correct answer, or a sentence with arbitrary English words. It shows that the subtle but critical difference in the paragraphs significantly confuse sixteen deep learning

models. [50] aims to fool a deep learning-based model by removing a minimum subset of words in the task of sentiment classification. [51] generates natural language adversarial examples via word replacements based on nearby contexts. Works [50] and [51] are not applicable to search queries, since queries are too short to remove terms from them or replace terms based on nearby terms. Besides these works, [52–55] construct adversarial examples by directly manipulating input embeddings. Random noise are added to input and hidden layers to introduce perturbations during training in [54]. Local distributional smoothness is proposed to promote model smoothness and help determine the adversarial direction in [52]. [52] also shows that training with adversarial perturbations outperforms the methods with random perturbations. [53] applies adversarial training in sequence models on text classification tasks and demonstrates that adversarial training improves not only the classification performance, but also the quality of word embeddings.

2.5 Few-shot Learning

Recent deep learning-based few-shot learning approaches fall into three main categories: (1) metric-based approaches [56–59], which try to learn a generalized distance metric. (2) model-based approaches [60], which use recurrent network with internal or external memory. (3) optimization-based approaches [61], which optimize model parameters explicitly for fast learning.

A siamese neural network is utilized to conduct one-shot image classification in [56]. The siamese neural network is composed of two twin networks and their outputs are jointly trained on top of a similarity function to learn the relationship between pairs of data points. The Matching Network [57] makes classification predictions by comparing the input samples with a small labeled support set. The relation network [59] is similar to the Siamese network, but differs by choosing a CNN to capture the relationship rather than a simple $L1$ distance. The prototypical network [58] defines a prototype vector to represent each class. The prototype prototype vector is calculated as the mean vector of the support data samples in each class, without any differential weighting mechanisms. A Gaussian prototypical network [62]

extends the prototypical network. Its maps an instance into an embedding vector, and an estimate of the instance quality. Together with the embedding vector, a confidence region around it is predicted, characterized by a Gaussian co-variance matrix. [63] learns edge-learnable graph neural networks to perform few-shot learning. The graph neural network formulation of few-shot learning generalizes a number of recent few-shot learning models, including siamese networks, Matching networks, and prototypical networks.

Previous few-shot learning research mainly focuses on vision learning [64], text classification tasks [65], or entity predictions on knowledge graphs [66]. To the best of our knowledge, this work proposed in this dissertation is the first research utilizing self-attentive few-shot learning in a location-based recommendation system.

CHAPTER 3

Customer Recommendation in Location-based Social Networks

3.1 Background

Recommender system has attracted substantial attention from researchers since the last decade and has revolutionized the e-commerce industry. Various recommender systems have been developed to facilitate the matching between customers with appropriate products or services, such as movies on Netflix, music on Last.fm, and merchandises on Amazon. For customers, recommendations improve user experience by providing helpful suggestions to explore and discover relevant products or services. For providers, these recommendations increase the propensity of purchases from customers.

Over the past few years, the prevalence of GPS-enabled devices, such as smart phones, establish the prosperity of location-based social networks (LBSN), such as Foursquare, Yelp, and Facebook Local. LBSN attracts millions of users to share their social friendship and their locations via check-ins. For example, an average of 142 million users check in at local businesses via Yelp every month [67]. Foursquare has 55 million monthly active users and 8 million daily check-ins on the Swarm application [68]. Facebook Local, powered by 70 million businesses [69], facilitates the discovery of local events and places for over one billion active daily users [70]. The check-ins, which contain abundant hints of user preferences on locations, allow us to identify potential new customers for local businesses.

To identify potential new customers, the most crucial thing is to understand a customer's decision-making process. However, it is a complex process, and can be influenced by multi-

ple factors. Most investigated factors are personal preferences and geographical convenience. Personal preferences are learned from customers' historical check-ins by applying collaborative filtering or matrix factorization techniques. The learned preferences, in return, help us find out new businesses which customers are interested in. In addition, check-in locations provide an ancillary resource to interpret customers' decisions from the perspective of geographical convenience. According to the Tobler's first law of geography and the law of demand, the propensity of a customer for a local business is inversely proportional to the distance between the customer and the business, which is similar to the probability of purchasing an item being inversely proportional to the cost.

There are also studies which show that customers prefer learning from local experts who know the neighborhood well and have firsthand experience [71]. This is because online reviews are becoming more and more influential in establishing and promoting the reputation of local businesses than ever before. The emergence of numerous review sites has created an unprecedented and ongoing online conversation about local businesses. Therefore, a business' reputation is more public and more accessible. Customers are able to see over the "wall" of corporate messaging at what lies behind. They can get a sense of a business' true essence through the shared experiences of other customers. These changes in marketing lead to a change in customers' habits. Customers are becoming more and more review-dependent. This is consistent with the study conducted by BrightLocal [72]. Compared with the trend in 2010, the number of people who search for local businesses before consumption doubled in 2015 and 2016. Moreover, among all the participants in the study, 92% of the customers regularly or occasionally read online reviews, which help them judge whether a local business provides good services or not. Therefore, the impact of online reviews is non-negligible and growing.

Although identifying potential new customers is crucial for local businesses, it is still a very challenging task due to the following three reasons.

- **Data Sparsity**. To know and comment on a local business, a customer has to physically visit that business. Thus, the cost is higher than that of rating a movie or a

Table 3.1: Statistics and densities of six datasets.

Dataset	#(Users)	#(Items/businesses)	#(Ratings/check-ins)	Density
Netflix	48,189	17,770	100,480,507	1.17×10^{-1}
MovieLens 20M	138,000	27,000	20,000,000	5.37×10^{-3}
last.fm	1,892	17,632	92,834	2.78×10^{-3}
Yahoo! Music	1,000,990	624,961	262,810,175	4.20×10^{-4}
Yelp	1,029,432	144,072	5,099,750	6.94×10^{-6}
Foursquare	1,219,322	422,030	693,798	1.35×10^{-6}

song online. Even if a customer makes the effort to visit the business, he/she often does not check in due to privacy or safety concerns [73], let alone writing a review. Therefore, customers' check-in data is much sparser than other rating data for movies and music. Table 3.1 shows the statistics and the densities of four well-known movie and music rating datasets, together with two LBSN datasets, i.e., the Yelp challenge dataset and the Foursquare dataset. Here the density of a dataset is calculated by the number of ratings/check-ins divided by the product of the number of users and the number of items/businesses. The densities of Yelp and Foursquare datasets are much lower than the ones of Netflix, MovieLens, Last.fm, and Yahoo! music datasets. The extremely sparse check-in data makes it challenging for us to accurately model customers' preferences.

- **Geographical Influence**. The first law of geography states that everything is related to everything else, but near things are more related than distant things [74]. Many studies show that people tend to visit nearby local businesses or explore businesses near the ones that they have visited before [13]. Therefore, a challenge is how to estimate customers' activity trajectories or zones based on the sparse check-in data. Beyond this estimation, a more challenging fact is that the geographical influence is both customer-dependent and business-dependent. If a customer owns a car, he can visit a faraway business with less effort than ones who cannot drive and rely on public

transits. On the other hand, the geographical factor has different impacts on different types of businesses. For example, customers tend to visit nearby fast-food businesses for convenience. However, they may be willing to travel farther to visit other types of businesses, such as museums, where they are more closely connected with cultures and get inspirations, or salons where they can have their hair cut and styled by professionals.

- **Reputation Influence**. Nowadays, more and more customers rely on online reviews to get a sense of the reputation of local businesses. These reviews implicitly influence customers' decisions towards visiting a business. The influence of reviews is also both customer-dependent and business-dependent. Different customers have different opinions on the same review. Moreover, similar reviews may have different impacts on different types of businesses. For example, a review such as "A bit of long wait" to a fast-food business may give other customers a very negative impression. However, the impact may be milder if the same comment is made on theme parks, such as Universal Studios.

In addition to the three challenges above, given the heterogeneous information on check-in, location, online reviews, current works also lack an integrated analysis of personal preferences, geographical influence, and business reputation when modeling customers' decisions. To the best of our knowledge, this work is the first one considering all these factors under the scenario of recommending customers for local businesses. To be more specific, the main contributions of this work are as follows:

- We propose a customer recommendation model, CORALS, which, based on historical check-in information, integrates customers' personal preferences, geographical influence, and business reputation. In addition, the model is also capable of incorporating other factors such as expenses. Moreover, the model offers high interpretability by providing the quantitative importance of incorporated factors for different types of local businesses.

- We present a comprehensive empirical evaluation of our approach against 12 recom-

mendation methods on two real-world datasets. The results show that our approach, CORALS, outperforms all baseline methods in suggesting potential new customers for local businesses in different cities.

The key to addressing the recommendation problem in LBSNs is to accurately understand customers' decision-making processes. In this work, we decompose it into three main factors: a customer i's personal preference $t_{b,i}$ over a business b, the geographical convenience $g_{b,i}$ of business b for customer i, and customer i's reliance $r_{b,i}$ on business b's reputation. In addition, the contributions of $g_{b,i}$ and $r_{b,i}$ are given by w_b^g and w_b^r, respectively. Formally, the tendency of a customer i's visiting a business b is given by:

$$t_{b,i} + w_b^g g_{b,i} + w_b^r r_{b,i}. \tag{3.1}$$

Higher tendency indicates higher check-in likelihood.

Given an observed check-in from customer i on business b, denoted by $(b,\ i)$, applying the idea of pair-wise comparisons, we sample another customer j who has not checked in at business b. Now, for a business b, we have an observed check-in $(b,\ i)$ and a sampled unobserved check-in $(b,\ j)$. It is logical to hypothesize that compared with customer j, customer i is more likely to visit business b. We construct the model by maximizing a posteriori over all observed and sampled check-ins:

$$C = \prod_{(b,i),j} p(i >_b j|\Theta)p(\Theta), \tag{3.2}$$

where Θ is a set of parameters, which define the model. $p(i >_b j|\Theta)$ gives the probability that a customer i prefers a business b more than another customer j does under the model. Formally,

$$p(i >_b j|\Theta) = \delta[(t_{b,i} - t_{b,j}) + w_b^g(g_{b,i} - g_{b,j}) + w_b^r(r_{b,i} - r_{b,j})], \tag{3.3}$$

where $\delta(x)$ is the sigmoid function:

$$\delta(x) = \frac{1}{1 + e^{-x}}. \tag{3.4}$$

In addition, the personal preference of customer i on business b is given by:

$$t_{b,i} = \boldsymbol{p}_b \cdot \boldsymbol{q}_i, \tag{3.5}$$

where $\boldsymbol{p}_b$ and $\boldsymbol{q}_i$ are business and customer vector representations in the preference hidden space, respectively. Similarly, the reputation reliance of a customer i on a business b is given by:

$$r_{b,i} = \boldsymbol{u}_b \cdot \boldsymbol{d}_i, \tag{3.6}$$

where $\boldsymbol{u}_b$ and $\boldsymbol{d}_i$ are business and customer vector representations in the reputation hidden space, respectively. Using Gaussian priors $\Theta \sim N(0, \lambda_\theta I)$ to model the parameters, we have

$$p(\Theta) = \frac{1}{\sqrt{2\pi}\sigma} e(-\frac{\|\Theta\|^2}{2\sigma^2}). \tag{3.7}$$

Substituting Equations 3.3, 3.4, 3.5, 3.6, 3.7 into the objective Equation 3.2, we can derive maximizing a posteriori as follows:

$$\begin{aligned}
C &\propto \ln \prod_{(b,i),j} p(i >_b j|\Theta) p(\Theta) \\
&= \sum_{(b,i),j} \ln \delta[(t_{b,i} - t_{b,j}) + w_b^g (g_{b,i} - g_{b,j}) + w_b^r (r_{b,i} - r_{b,j})] + \ln p(\Theta) \\
&= \sum_{(b,i),j} \ln \delta[(t_{b,i} - t_{b,j}) + w_b^g (g_{b,i} - g_{b,j}) + w_b^r (r_{b,i} - r_{b,j})] - \lambda \|\Theta\|^2 \\
&= \sum_{(b,i),j} \{\ln \delta[(\boldsymbol{p}_b \cdot \boldsymbol{q}_i - \boldsymbol{p}_b \cdot \boldsymbol{q}_j) + w_b^g (g_{b,i} - g_{b,j}) + w_b^r (\boldsymbol{u}_b \cdot \boldsymbol{d}_i - \\
&\boldsymbol{u}_b \cdot \boldsymbol{d}_j)] - \lambda \|\Theta\|^2\},
\end{aligned}$$

where λ is a set of regularization parameters for Θ. Here, the businesses' geographical convenience, $g_{b,i}$, and businesses' reputations, u_b, are inputs. These two factors will be discussed in Sections 3.2 and 3.3, respectively.

To optimize top ranked customers in the recommendation list, we apply the weighted approximate ranking strategy proposed in [75] to optimize precision@k. Algorithm 1 summarizes the optimization process. First, the parameters Θ are initialized using Normal

Algorithm 1: Parameter optimization with AdaGrad

Input: learning rate η, max iteration $iter_{max}$, regularization weights λ, max number of samples S_{max};

Output: Θ

Initialization: initialize Θ with Normal distribution $N(0,0.01)$, $iter = 0$, $\Theta_{opt} = \Theta$, $\text{err}_{opt} = err_{vali}$;

repeat

  foreach *observed check-in* (b, i) **do**

    Counter $cnt = S_{max}$;

    while $cnt > 0$ **do**

      Randomly generate an unobserved customer j;

      if $(t_{b,i} - t_{b,j}) + w_b^g(g_{b,i} - g_{b,j}) + w_b^r(r_{b,i} - r_{b,j}) > 0$ **then**

        cnt- -;

      else

        foreach *involved* θ **do**

          $\nabla\theta^{ts} = \frac{\partial J}{\partial \theta^{ts}}$;

          $n_\theta^{ts+1} = n_\theta^{ts} + (\nabla\theta^{ts})^2$;

          $\theta^{ts+1} = \theta^{ts} - \frac{\eta}{\sqrt{n_\theta^{ts}+\epsilon}}\nabla\theta^{ts}$;

        break;

  if $err_{vali} < err_{opt}$ **then**

    $err_{opt} = err_{vali}$;

    $\Theta_{opt} = \Theta$;

  else

    $\Theta = \Theta_{opt}$;

  $iter + +$;

until $iter > iter_{max}$;

Return Θ_{opt};

distributions. The optimization process is iterative. In each iteration, it goes through each observed check-in in the training set. For each observed check-in (b, i), we sample a random customer j who has not visited business b. If the preference order between i and j on business b is correctly predicted using the current Θ, we randomly sample another customer to find a violation. This process repeats at most S_{max} times until we find such a violation. Once we find a violation, we update the corresponding parameter θ, $\theta \in \Theta$. After iterating through each check-in in the training set, we evaluate the performance using the validation set. If the performance increases, we accept the updates on Θ. Otherwise, we reject the updates. This step helps us avoid adopting over-fitting parameters on the training data. The optimization terminates when *iter* reaches the maximum number of iterations.

As we mentioned, since many businesses and customers have limited numbers of check-ins, the check-in data is extremely sparse. However, the parameters of these businesses and customers can be immensely useful and informative to the problem we want to optimize. To effectively leverage the sparse data, AdaGrad [76] is proposed to give a higher learning rate to the parameters that are more sparse in the data. We adopt this concept to adjust the learning rate adaptively for each individual parameter θ, which is shown in lines 10-13 of Algorithm 1. AdaGrad modifies the general learning rate η at each time step ts for every parameter θ based on the past gradients that have been computed for θ. n_θ^{ts+1} records the sum of the squares of the gradients with respect to θ up to the ts^{th} time step[1]. ϵ is a smoothing term that avoids division by zero. In this way, AdaGrad makes it such that parameters that are more sparse in the data have a higher learning rate which translates into a larger update for that parameter.

3.2 Geographical Convenience Inference

In this section, we discuss how to infer the geographical convenience of a business b for a user i, i.e., $g_{b,i}$, based on customer i's historical check-ins.

[1]In one iteration, the same parameter may be optimized multiple times. Each optimization counts 1 time step.

We apply a Gaussian mixture model (GMM) [77] to make the inference. A Gaussian mixture model is a weighted sum of M component Gaussian densities:

$$p(\boldsymbol{l}|\Phi) = \sum_{m=1}^{M} \alpha_m g(\boldsymbol{l}|\boldsymbol{\mu}_m, \Sigma_m), \tag{3.8}$$

where $\boldsymbol{l}$ is a 2-dimensional location vector (i.e. latitude and longitude), α_m, $m = 1, ..., M$, are the mixture weights, and $g(\boldsymbol{l}|\boldsymbol{\mu}_m, \Sigma_m)$ are the component Gaussian densities. Each component density is a 2-variate Gaussian function of the form,

$$g(\boldsymbol{l}|\boldsymbol{\mu}_m, \Sigma_m) = \frac{1}{2\pi|\Sigma_m|^{1/2}} e^{-\frac{1}{2}(\boldsymbol{l}-\boldsymbol{\mu}_m)'\Sigma_m^{-1}(\boldsymbol{l}-\boldsymbol{\mu}_m)},$$

with mean location vector $\boldsymbol{\mu}_m$ and covariance matrix Σ_m. The complete Gaussian mixture model is parameterized by the mean location vectors, covariance matrices and mixture weights from all component densities. These parameters are further collectively notated by Φ. For a particular customer, given a sequence of his N check-in locations, represented by N location vectors $L = \{\boldsymbol{l}_1, .., \boldsymbol{l}_N\}$, the GMM likelihood, assuming conditional independence between the location vectors, can be written as:

$$p(L|\Phi) = \prod_{n=1}^{N} p(\boldsymbol{l}_n|\Phi).$$

We use the Expectation-Maximization (EM) [78] algorithm to estimate the parameters. The EM algorithm begins with an initial model Φ, to estimate a new model $\bar{\Phi}$, such that $p(L|\bar{\Phi}) \geq p(L|\Phi)$. The new model then becomes the initial model for the next iteration and the process is repeated until convergence. In each EM iteration, re-estimation Equations 3.9, 3.10, and 3.11 are used to guarantee a monotonic increase in the model's likelihood value in the E-step.

$$\text{Mixture weights: } \bar{\alpha}_m = \frac{1}{N} \sum_{n=1}^{N} p(m|\boldsymbol{l}_n, \Phi), \tag{3.9}$$

$$\text{Location means: } \bar{\mu}_m = \frac{\sum_{n=1}^{N} p(m|\boldsymbol{l}_n, \Phi) \cdot \boldsymbol{l}_n}{\sum_{n=1}^{N} p(m|\boldsymbol{l}_n, \Phi)}, \tag{3.10}$$

$$\text{Variances: } \bar{\sigma}_m^2 = \frac{\sum_{n=1}^{N} p(m|\boldsymbol{l}_n, \Phi) \cdot \boldsymbol{l}_n^2}{\sum_{n=1}^{N} p(m|\boldsymbol{l}_n, \Phi)} - \bar{\mu}_m^2, \tag{3.11}$$

In the M-step, the posteriori probability for component m is given by

$$p(m|\boldsymbol{l}_n, \Phi) = \frac{\alpha_m g(\boldsymbol{l}_n|\boldsymbol{\mu}_m, \Sigma_m)}{\sum_{m=1}^{M} \alpha_m g(\boldsymbol{l}_n|\boldsymbol{\mu}_m, \Sigma_m)}.$$

To determine the number of Gaussian components M, we apply affinity propagation [79] to cluster each customer's check-ins. The number of clusters yields the number of Gaussian components.

After the GMM construction for a customer i, given the geographical location l_b of a business b, as shown in Equation 3.8, $p(\boldsymbol{l_b}|\Phi)$ gives the geographical convenience $g_{b,i}$ of the business b for each customer i.

3.3 Business Reputation Inference

In this section, we discuss how to model the business reputation, $\mathbf{u}_b$, based on the reviews commented on the local businesses.

There are two main challenges. First, reviews differ in their lengths. Some reviews are informative and have more words while others are not. This challenge makes it difficult to model the business's reputation $\mathbf{u}_b$ in a fixed-length vector. Second, for a particular business, some reviews are older while others are more recent. They may have different influences on the reputation of the business.

To solve the first challenge, we apply a distributed memory model proposed in [80]. Figure 3.1 shows the framework for the vector learning task, which is to predict a word given other words in a context. Formally, given a sequence of training words o_1, o_2, o_3, ..., o_H, the objective of the model is to maximize the average log probability

$$\frac{1}{H} \sum_{h=k}^{H-k} logp(o_h|o_{h-k}, ..., o_{h+k}).$$

The prediction task is performed via a multiclass classifier, i.e., softmax. Then, we have:

$$p(o_h|o_{h-k}, ..., o_{h+k}) = \frac{e^{y_{o_h}}}{\sum_o e^{y_o}}.$$

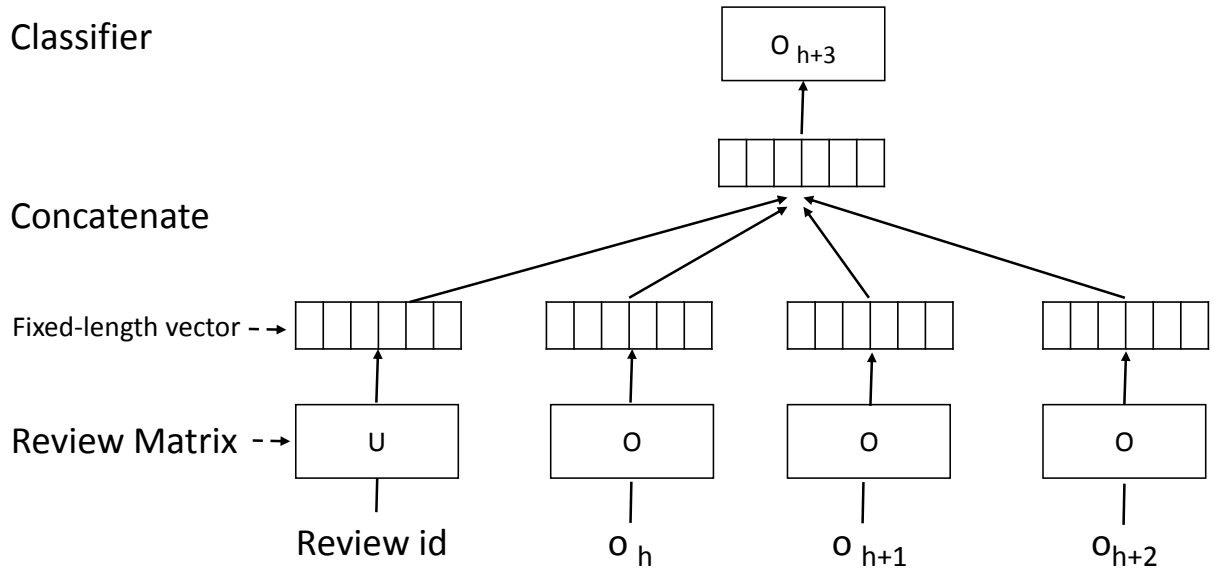


Figure 3.1: The framework for learning review vector

Each y_{o_h} is the un-normalized log probability for each output word o_h, calculated as:

$$y_{o_h} = V_0 + V z(o_{h-k}, ..., o_{h+k}, U),$$

where V_0 and V are the softmax parameters. z is constructed by a concatenation of a review vector and word vectors from O. Both review and word vectors are trained using stochastic gradient descent (SGD) and the gradient is obtained via back propagation. At each step of SGD, we sample a fixed-length context from a random review, compute the error gradient and update the parameters in the model. Once the parameters get converged, we obtain the dense representation of each review. In order to address the impact of the chronological order of the reviews, we use the vector of the most recent review as the reputation vector of the business, $\mathbf{u}_b$.

CHAPTER 4

Customer Recommendation with Few-shot Learning

4.1 Background

As an increasingly popular application of location-based services, location-based social networks (LBSNs), such as Yelp, Foursquare, and Instagram, attract millions of users to share their locations, resulting in a huge amount of user check-ins [67,68]. The availability of such unprecedented user check-ins brings in great opportunities to understand users' preferences and help businesses identify potential new customers. However, new customer predictions in LBSNs suffer severely from data sparsity. To know a business, a customer has to physically visit that business. Even if a customer makes the effort to visit the business, he/she often does not check in due to privacy or safety concerns [73]. This results in extremely sparse check-in data. Understanding customer preferences and making accurate predictions from the severely sparse data remain a daunting task.

Figure 4.1: Geographical convenience influence

To compensate for the check-in sparsity, various ancillary information, such as geographical influence, social correlations, temporal patterns, textual and visual contents, has been leveraged to improve recommendation performances in different manners [16,17,22,27,28,81,82]. For LBSNs, the geographical coordinates, i.e., the latitude and longitude, of businesses

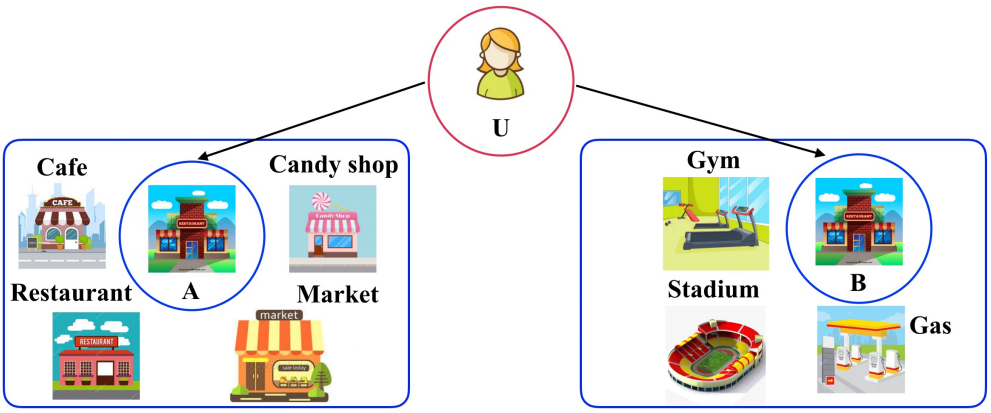


Figure 4.2: Geographical dependency influence

are the most accessible ancillary information and they are also the ones that make location-based recommendations unique compared to other recommendation tasks. However, most existing works only investigate the relationship between customers and businesses by measuring the distance of a visit from users to businesses. This leads to two limitations. First, distance may not be an accurate indicator to distinguish the transportation convenience for different users for a check-in. Second, the inter-dependencies among nearby businesses are not modeled when making recommendations.

In this work, we highlight that geographical convenience and dependency should be both incorporated to comprehensively leverage the geographical influence. Figure 4.1 and Figure 4.2 show two motivating examples for geographical convenience and dependency, respectively. In Figure 4.1, the two users are both 5 miles away from business A. Therefore, the distance indicator does not offer too much disciminative power to tell who is more likely to visit from the geographical perspective. But if we know user 1 tends to drive while user 2 relies on walking, we could gauge the actual transportation efforts more accurately based on the convenience rather than the raw distance. In Figure 4.2, two businesses A and B provide the same service and they are reachable for user u with equal transportation efforts. Without considering the neighborhood information of the two businesses, the recommendation system can barely distinguish one from the other regarding u's preference. In real-world scenarios, the neighborhood services of two businesses are never the same. In this example,

A is surrounded by a cafe, a candy shop, a market, and a restaurant, while B is near a gym, a stadium, and a gas station. If such neighborhood information is modeled when making recommendations, it can provide extra guidance to understand users' decision-making processes more comprehensively and thus make more accurate recommendations. To incorporate geographical convenience, we leverage Gaussian mixture models (GMM) [77], where each user is maintained with a profile described by a mixture of geographical activity functions. Given a business, the user profile could yield the relative transportation efforts. To incorporate geographical dependency, we utilize graph convolutional networks [83], which allow geographical features propagate among neighborhood. Therefore, the representation of a business can not only capture its own service and quality, but also those of its neighbors.

Beyond embracing geographical influence to address the sparsity issue, we also strive to seek more suitable techniques to improve the recommendations. Few-shot learning, as a contemporary approach, decomposes the training into a set of similar tasks, where transferable knowledge is learned and shared among tasks. It allows learning in sparse data much faster than otherwise possible [84]. In this work, we propose a metric-learning-based few-shot learning framework. In particular, we construct support instances and query instances, with each instance composed of one customer and one business. Support instances are labeled instances and serve as references. Query instances rely on the references to conduct reasoning. The model evolves by iterative comparisons between support and query instances. In this way, the matching between a customer and a business is optimized with explicit attention to multiple other related check-in behaviors. Therefore, the limited check-ins are comprehensively utilized to address the sparsity issue.

In addition, different from previous works, which treat historical user check-ins at a business equally, we adaptively differentiate each user check-in by modeling the check-in reason as a mixture of hidden factors, which is achieved by leveraging attention mechanisms. In a nutshell, few-shot learning allows us thoroughly utilize the sparse check-in data. Differentiation of user check-ins with attention mechanisms helps us model and understand users' decision-making processes more comprehensively.

In this work, we study the problem of potential new customer recommendation with few-shot learning. In essence, we solve three challenges: (1) how to incorporate geographical convenience and dependencies when making recommendations; (2) how to utilize few-shot learning to model a recommendation task; and (3) how to differentiate user check-in behaviors on the same business. To be more specific, the main contributions of this work are as follows:

- We decompose the geographical influence into geographical convenience and geographical dependency. The geographical convenience models the relative transportation efforts of a check-in, while the geographical dependency modeling makes our model neighborhood-aware.
- We are the first to apply meta-learning to location-based recommendation tasks and formulate the problem as few-shot learning.
- To distinguish user check-in behaviours on the same business, we introduce multiple self-attention mechanisms to explain each check-in against a set of reference check-ins.
- We present a comprehensive empirical evaluation of our approach against 13 recommendation methods on two real-world datasets. The results show that our approach, SEATLE, outperforms all baseline methods in suggesting potential new customers for businesses in different cities.

In this work, user check-ins are represented as a collection of tuples $\{(b, u)\} \subseteq B \times U$, where B and U are the business set and user set, respectively. The task of new user recommendation is to rank users given a business. The goal is to rank the true new users higher than other candidates. The candidates here are all users who have not checked in this business.

4.2 Few-Shot Learning Settings

This section describes the training of our method formulated as few-shot learning.

Following the standard few-shot learning settings [57, 58, 85], we assume access to a set of training tasks. In our problem, each training task T corresponds to the new user predictions regarding a business b. During training, it learns to learn a generalized similarity metric to compare a set of user-business tuples against some references for each task, with each one designed to simulate the few-shot setting. Tasks are optimized one after another multiple times. For each task, each time k observed check-in tuples are randomly sampled as references, denoted as R. An observed check-in tuple refers to a business-user pair (b, u) where the user u did check in business u in the dataset. In addition, two query sets, i.e., a positive query set Q^+ and a negative query set Q^-, are constructed, with each set made up of c tuples. Each query in Q^+ is also an observed check-in tuple regarding b, but distinct from the ones in R. Each query in Q^- is a fake check-in tuple, where the user in the tuple did not check in b. The model thus can be optimized by comparing two similarities, one between a positive query and references, and the other one between a negative query and reference for each business. Ranking loss is applied to conduct the model optimizations, where the ranking loss measures how well the model distinguishes a positive query from a negative query regarding a set of references. The optimizations run for multiple iterations and each business is optimized more than one times. Note that for each optimization of a business, we may select different observed check-ins as references and positive queries. Similarly, we may construct different fake check-ins are negative queries. Once trained, the embeddings of all observed check-ins regarding the same business are expected close to each other in the hidden space, while the embedding of a fake check-in is expected faraway from the embeddings of observed check-ins.

4.3 Overall Framework Utilizing Few-shot Learning

In this section, we explain how to model the recommendation task as few-shot learning, how to incorporate geographical influence, and how to distinguish user check-ins in detail.

The proposed approach decomposes the recommendation problem into a set of tasks and each task involves the user recommendations with respect to only one business. Figure 4.3

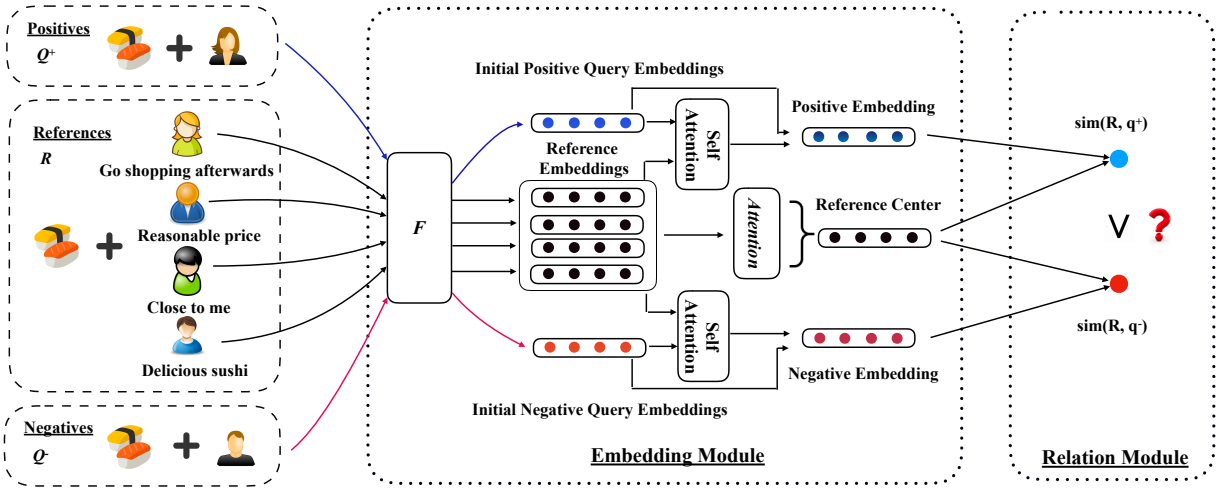


Figure 4.3: Few-shot learning framework

shows the few-shot learning framework for each task. On the left side, a reference set R and two query sets, a positive query set Q^+ and a negative query set Q^-, are constructed for a business b. The reference set is composed of k random observed check-ins, where $k = 4$ in this example. The positive query set is made up of another random selection of c observed check-ins. But the check-ins in Q^+ are mutually exclusive alternatives from the ones in R. For illustration simplicity, c is set to 1 in the example. The negative query set is constructed by building c user-business tuples, such that each user in the tuple does not check in business b. The reference set functions as the supports in the setting of few-shot learning. The two query sets, based on the reference set, jointly conduct reasoning and inference.

The framework has two modules, an embedding module and a relation module. The embedding module learns the representations of references and queries, while the relation module compares the learnt representations and optimize them in such a manner that representations of positive queries are similar to the ones of references, while representations of negative queries are dissimilar to the ones of references. In the embedding module, $F(\cdot)$ is a layer which learns the initial embeddings of reference, positive query, and negative query tuples. By going through F, each reference/positive/negative tuple is represented by a fixed-length vector. Attention is a layer which utilizes the attention mechanism to learn the representative of the references. Self Attention is another attention layer, which takes

both initial query and reference embeddings as inputs to generate the relative embeddings for queries. The relative embedding of a query learns to use references to explain the user check-in behavior encoded in the query. In the relation module, based on the learnt embeddings, we match each query in the query set to the reference representative by calculating the similarity between them, denoted as $\text{sim}(R, q)$. Then, we compare the score $\text{sim}(R, q^+)$ of a positive query q^+ with the score $\text{sim}(R, q^-)$ of a negative query q^-. Ranking loss is generated if a negative query is more similar to the reference representative than a positive query is. The model gets optimized by minimizing such ranking loss.

In the following paragraphs, we first talk about what features we use to encode a user-business tuple. Then, we explain the embedding module and the relation module.

Given a tuple (b, u), $F(\cdot)$ encodes four types of features: (1) business features, which represent its service, quality, and other business self-related factors; (2) user features, which represent his/her preference; (3) features indicating the geographical convenience of b for user u; and (4) features indicating the geographical dependencies of b, i.e., the neighborhood information of b. These four types of features collectively express how likely the user u will check in the business b.

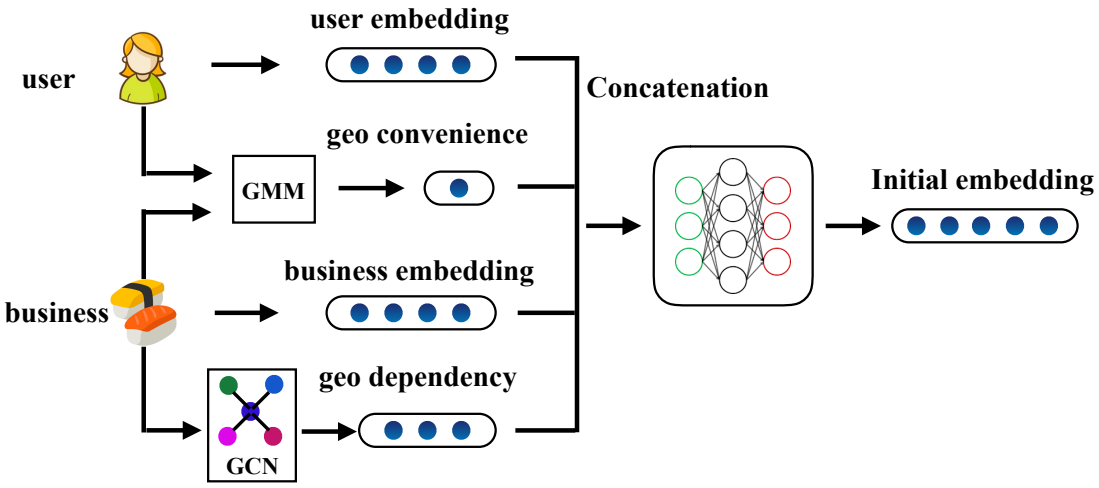


Figure 4.4: Feature constructions

Figure 4.4 illustrates the initial embedding construction for a user-business tuple. Given

a tuple, two vectors, a user embedding vector and a business embedding vector, are utilized to encode user preferences and business self-related features, respectively. A geographical convenience vector is constructed by considering the geographical location of the business and the historical check-in locations of the user. A geographical dependency vector is constructed to encode the neighborhood information of the business. These four types of information are concatenated together and then fed into a fully-connected neural network to derive the initial embedding of a user-business tuple. In Sections 4.4 and 4.5, we will present the geographical convenience modeling and geographical dependency modeling in detail, respectively.

4.4 Geographical Convenience Modeling

In this section, we discuss how to model the geographical convenience of a business b for a user u based on u's historical check-ins.

We apply the Gaussian mixture model [77] to make the inference. A Gaussian mixture model is a weighted sum of M component Gaussian densities:

$$p(\boldsymbol{l}|\Phi) = \sum_{m=1}^{M} \alpha_m g(\boldsymbol{l}|\boldsymbol{\mu}_m, \Sigma_m), \tag{4.1}$$

where $\boldsymbol{l}$ is a 2-dimensional location vector (i.e. latitude and longitude), α_m, $m = 1, ..., M$, are the mixture weights, and $g(\boldsymbol{l}|\boldsymbol{\mu}_m, \Sigma_m)$ are the component Gaussian densities. Each component density is a 2-variate Gaussian function of the form,

$$g(\boldsymbol{l}|\boldsymbol{\mu}_m, \Sigma_m) = \frac{1}{2\pi|\Sigma_m|^{1/2}} e^{-\frac{1}{2}(\boldsymbol{l}-\boldsymbol{\mu}_m)'\Sigma_m^{-1}(\boldsymbol{l}-\boldsymbol{\mu}_m)},$$

with mean location vector $\boldsymbol{\mu}_m$ and covariance matrix Σ_m. The complete Gaussian mixture model is parameterized by the mean location vectors, covariance matrices and mixture weights from all component densities. These parameters are further collectively notated by Φ. For a particular customer, given a sequence of his $\mathcal{T}$ check-in locations, represented by $\mathcal{T}$ location vectors $L = \{\boldsymbol{l}_1, .., \boldsymbol{l}_\mathcal{T}\}$, the GMM likelihood is written as:

$$p(L|\Phi) = \prod_{\tau=1}^{\mathcal{T}} p(\boldsymbol{l}_\tau|\Phi).$$

We use the Expectation-Maximization algorithm [78] to estimate the parameters. Due to the space limit, we skip the detailed optimizations here.

To determine the number of Gaussian components M, we apply affinity propagation [79] to cluster each customer's check-ins. The number of clusters yields the number of Gaussian components. After the GMM construction for a customer u, given the geographical location l_b of a business b, as shown in Equation 4.1, $p(\boldsymbol{l}_b|\Phi)$ gives the geographical convenience of the business b for each user u.

We highlight that the geographical convenience, modeled by GMM, is superior to conventional distance-based metrics, since it captures the relative geographical efforts of a visit, which is capable of distinguishing customers with different traveling flexibility more accurately.

4.5 Geographical Dependency Modeling

In this section, we show how to encode geographical neighborhood information using graphs and how to model the dependence relationship among businesses using graph convolutional networks [83].

The geographical correlations among businesses are modeled with a graph $G = (V, E)$, which encodes the geographical proximity. Each vertex $v \in V$ represents a business and an edge $e \in E$ with weight $e^{-\lambda(v_i,v_j)}$ connects every two vertices v_i and v_j, where $\lambda(v_i, v_j)$ gives the geographical distance between v_i and v_j. Formally, an adjacency matrix A is used to represent G with $A_{i,j} = e^{-\lambda(v_i,v_j)}$.

Graph convolutional network (GCN) is defined over the proximity graph, which allows us to extract and aggregate neighborhood information for each vertex. A graph convolution is defined as:

$$\boldsymbol{H}^{(\beta+1)} = \text{Sigmoid}(\boldsymbol{D}^{-\frac{1}{2}}\hat{\boldsymbol{A}}\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{H}^{(\beta)}\boldsymbol{W}^{(\beta)}), \tag{4.2}$$

with $\hat{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}$, where $\boldsymbol{I}$ is the Identify matrix, which is added to capture business' own features during feature propagations. $\boldsymbol{D}$ is the diagonal node degree matrix of $\hat{\boldsymbol{A}}$. $\boldsymbol{W}^{(\beta)}$ is

the weight matrix for the β-th layer in GCN, and $\boldsymbol{H}^{\beta}$ is the output for the β-th layer. In particular, $\boldsymbol{H}^{(0)} = \boldsymbol{X}$ and $\boldsymbol{H}^{(\tilde{\beta})} = \boldsymbol{Z}$, where $\boldsymbol{X}$ is the initial vertex feature matrix, $\boldsymbol{Z}$ is the final outputs of GCN, with $\tilde{\beta}$ indicating the number of layers in GCN.

For example, $\boldsymbol{H}_i^0$ represents the initial features of business i. By going through GCN, the information of i's neighbors gets propagated to $\boldsymbol{H}_i^{\beta}$. Therefore, $\boldsymbol{H}_i^{\beta}$ not only represents the information of business i, but also that of its nearby neighbors. There are multiple ways to construct the feature matrix $\boldsymbol{X}$ for businesses in the graph. In this work, we utilize the business service types to achieve that. The service type of a business tells whether the business is a museum, or a supermarket, etc. $\boldsymbol{X}_i$ is constructed by its corresponding service embedding.

4.6 Embedding and Relation Module

Algorithm 2 summarizes the training process. For each training epoch, we go through the tasks one by one. For each task, we aim to distinguish positive queries from negative queries regarding references. For a business b, we first sample a set of k observed check-ins as the reference set, $R = \{(b, u_1^r), ..., (b, u_k^r)\}$. Then, we sample another set of c exclusive observed check-ins as the positive query set, $Q^+ = \{(b, u_1^+), ..., (b, u_c^+)\}$. We also construct a third set of c fake check-ins as the negative query set, $Q^- = \{(b, u_1^-), ..., (b, u_c^-)\}$. After the constructions of references, positive and negative queries, we calculate the similarity between each query in $Q^+ \cup Q^-$ and the references. We expect that positive queries are closer to references, while negative queries are faraway from references in the hidden space. The representations of queries and references are learnt through two attention mechanisms. The closeness/similarity between a query and a set of references is calculated by comparing the query embedding with the embedding of the reference representative. For each optimization of a business, we randomly pair up positive queries with negative queries. Ranking loss is adopted if a negative query is closer to the references than a positive query is. In the following paragraphs, we discuss the representation learning of both queries and references, the query-reference similarity calculation, and the ranking loss function in detail sequentially.

Algorithm 2: Few-shot Training

Input: Meta-learning task set;

Output: A set of model parameters Θ;

Initialization: initialize Θ with Normal distributions;

foreach *epoch = 1:N* **do**

  Shuffle the tasks in the task set;

  foreach *T_b in the task set* **do**

    Sample a set R of observed user check-ins as references;

    Sample a set Q^+ of observed user check-ins as positive queries;

    Construct a set Q^- of fake user check-ins as negative queries;

    Learn reference and query representations;

    Calculate the representative of the references with attention;

    Calculate the similarities between references and the queries in Q^+ and Q^-;

    Calculate ranking loss $\mathcal{L} = \sum_{(q^+,q^-)} \ell$;

    Update Θ based on gradients $g \propto \nabla\mathcal{L}$;

Return Θ;

Query embedding: the query embedding is constructed by incorporating two types of information. One encodes the user business interaction behavior itself and the other one encodes the representation with attention to the references. In other words, we attempt to use references to explain the user business interaction behavior in the query. $F(\cdot)$ in Figure 4.3 yields the first part, while the second part is achieved by introducing a self-attention mechanism. The scaled dot-product attention [86] is defined as:

$$\text{Attention}(\tilde{\boldsymbol{Q}}, \tilde{\boldsymbol{K}}, \tilde{\boldsymbol{V}}) = \text{softmax}(\frac{\tilde{\boldsymbol{Q}}\tilde{\boldsymbol{K}}^T}{\sqrt{d_Q}})\tilde{\boldsymbol{V}}, \tag{4.3}$$

where $\tilde{\boldsymbol{Q}}$, $\tilde{\boldsymbol{K}}$, and $\tilde{\boldsymbol{V}}$ represent the queries, keys, and values in the attention mechanism, respectively. The attention operation calculates a weighted sum of all values, where the weight between query i and value j relates to the interaction between query i and key j. The scale factor $\sqrt{d_Q}$ is used to avoid overly large values of the inner product, where d_Q is the feature dimension of both $\tilde{\boldsymbol{Q}}$ and $\tilde{\boldsymbol{K}}$.

In our case, the self-attention operation takes the query embeddings $\boldsymbol{Q} \in \mathbb{R}^{c\times d}$ and the reference embeddings $\boldsymbol{R} \in \mathbb{R}^{k\times d}$ as inputs, converts them to three matrices through linear projections, and feeds them into an attention layer:

$$\boldsymbol{Q}^R = \text{Attention}(\boldsymbol{Q}\boldsymbol{W}^Q, \boldsymbol{R}\boldsymbol{W}^K, \boldsymbol{R}\boldsymbol{W}^V), \tag{4.4}$$

where the projection matrices $\boldsymbol{W}^Q$, $\boldsymbol{W}^K$, and $\boldsymbol{W}^V \in \mathbb{R}^{d\times d}$. The self-attention result $\boldsymbol{Q}^R$ learns the embedding of a query by comparing the closeness between the query and all references. $\boldsymbol{Q}^R$ is a weighted sum of reference embeddings, where each weight gauges the behavior similarity between the query and a reference. In this way, $\boldsymbol{Q}^R$ encodes the user-business behavior of the query explained by references.

We employ residual shortcut connection [87] to derive the final representations for queries Q, denoted as Q^{com}, as follows:

$$\boldsymbol{Q}^{com} = \boldsymbol{Q}^R + \boldsymbol{Q}. \tag{4.5}$$

The representation $\boldsymbol{Q}^{com}$ is composed of two parts, i.e., $\boldsymbol{Q}^R$ and $\boldsymbol{Q}$. $\boldsymbol{Q}^R$ reflects the relation between the query and references, while $\boldsymbol{Q}$ represents the check-in behavior itself of the

query. In particular, $\boldsymbol{Q}^R$ captures the scenario when the check-in behavior of a query can be well explained by references. The self-attention mechanism allows $\boldsymbol{Q}^R$ focus on relevant references. Most positive queries can benefit the representation learning from the self-attention mechanism. However, it may not work all the time. For example, the sampled references may not cover the reason of a check-in for a positive query. Or for negative queries, the references are not designed to explain the fake interactions at all. When the references fail to explain, $\boldsymbol{Q}^R$ encodes the information of $\boldsymbol{Q}$. In such cases, we learn from the query itself, encoded in $\boldsymbol{Q}$. The residual shortcut connection as shown in Equation 4.5 well captures the different scenarios without introducing extra parameters.

Reference embedding: for the references, we calculate the reference representative $\bar{R}$ as a weighted sum of each reference, where the weights can be derived from a second attention mechanism.

$$\alpha_i = \text{softmax}(\sigma(\boldsymbol{W}_A \boldsymbol{R}_i + \boldsymbol{b}_A)\boldsymbol{V}_C^T). \tag{4.6}$$

$$\bar{\boldsymbol{R}} = \sum_i \alpha_i \boldsymbol{R}_i. \tag{4.7}$$

Equations 4.6 and 4.7 summarize the representative calculation. Each reference $\boldsymbol{R}_i$ is first fed into a one-layer neural network, the outputs of which, together with the context vector $\boldsymbol{V}_C$, are further used to generate the importance weight α_i for each reference R_i through a softmax function. The representative $\bar{\boldsymbol{R}}$ is calculated as a weighted sum of the references based on the derived weights.

Similarity and loss function: Given a set of references R and a query q, the similarity $\text{sim}(R, q)$ between R and q is defined as the dot product between $\bar{\boldsymbol{R}}$ and $\boldsymbol{q}^{com}$. Formally,

$$\text{sim}(R, q) = \bar{\boldsymbol{R}} \cdot \boldsymbol{q}^{com}. \tag{4.8}$$

We apply hinge loss to gauge the ranking error defined on references R, a positive query q^+, and a negative query q^-:

$$\ell = \max\{0,\ \text{sim}(R, q^-) - \text{sim}(R, q^+) + \gamma\}, \tag{4.9}$$

where γ is the margin. Losses are generated only when the negative query is closer to the references than the positive query regarding a margin γ.

As shown in Figure 4.3, users in the reference set choose to check in a business due to various factors, such as the cost, the cuisine type, the convenience to get there, and the neighborhood services etc. Similarly, a user in the positive query set decides to check in the same business due to only one or a mixture of the above factors. The attention mechanism in the query embedding constructions learns to assign importance weights to the seen references. This enhances the reasoning between queries and references, which is key in few-shot learning. The attention mechanism in the reference representative constructions are shared among all tasks and it adaptively analyzes and memorizes the key factors that result in users' check-ins. The memorized knowledge are generalized and transferable among different tasks. Therefore, the knowledge gained from businesses with rich check-ins can benefit the parameter learning for businesses which provide similar services, but have fewer check-ins.

CHAPTER 5

Click Feedback-Aware Query Recommendation Using Adversarial Examples

5.1 Background

The effectiveness of keyword-based search engines, such as Google, Bing, and Yahoo, depends largely on the ability of a user to formulate expressive search queries. Expressive search queries clearly and unambiguously describe users' search intents and bring users directly to the desired information, which make search engines powerful tools for tapping into the wealth of knowledge accessible through the world-wide-web. In practice, translating human thoughts into concise sets of keywords to form queries is never straightforward [88]. This is particularly true for search engine users, who are mostly untrained casual users. In many cases, casual users have very limited background knowledge about the information they are searching for. To assist users in formulating queries, modern search engines are equipped with query suggestions [1, 5, 33, 34]. Given a user query, a query suggestion system deduces the search intent of the user by recommending a set of queries that are more expressive than the original user input. Our goal is to investigate the key concerns of existing query suggestion systems and to propose a novel approach to improve the suggestion performance.

Web search queries are usually very short, typically with only one or two keywords each [3, 89]. Short queries lead to the ambiguity issue. For example, the query "apple price" can refer to Apple stock price, the price of various Apple electronic products, or the price of an apple as a kind of fruit. Up to 23.6% of web search queries are reported to be ambiguous in [90]. The ambiguity weakens the expressiveness of queries because the search engine may

not understand the users' hidden search intents. This results in poor rankings of retrieval results and jeopardizes user experiences consequently.

A successful query suggestion system depends on understanding and modeling user search intents accurately. The user search intents lie in interactions, i.e., searches and clicks, between users and search engines. These interactions are generally partitioned into groups to form search sessions based on the time when they happened. Each search session is driven by consecutive related search queries and clicks of search engine users. In a search session, a user refines his/her original query and submits a sequence of follow-up queries to pinpoint his/her information need. Between two issued queries, the user may also click suggested queries to explore. The issued queries, together with the clicks, jointly allow us to accurately understand users' hidden search intents and come up with appropriate queries as suggestions. Unfortunately, most existing works merely consider clicked suggestions as feedback when modeling user search intents.

To conduct feedback-aware query suggestions, we mine search logs. A search log contains historical records, each of which registers the details of a web search conducted by a user. Table 5.1 shows some sample records extracted from the search log of a real search engine. Each record includes a query string, an anonymous user ID by whom the query was formulated, query submission time, the query suggestion subsequently clicked by the user (if done), and the suggested queries shown on the search page with the clicked ones highlighted in bold. The sample records show two search sessions. In the first session, user A first submitted query "apple" to the search engine, and then clicked the second query suggestion "apple stock" in the suggestion list. After that, the user issued the second query "apple price", and clicked the suggestion "apple price per share". The second search session involved a different user, B. S/he started searching with the query "disney", and then clicked "disney cartoons" in the suggestion list. He/she continued the search with the query "disney shows" and clicked "disney channel cartoons" subsequently. From the first search session, we can see that the clicked suggestion "apple stock", as feedback of the user, compensates for the ambiguity of the second issued query "apple price". It provides us with extra infor-

mation to infer the underlying search intent of the user, which is to know the stock price of Apple instead of the price of Apple electronic products, such Apple iPhone. The clicked suggestion "disney cartoons" in the second search session also implicitly indicates that the user was interested in disney cartoons shows when the user search for "disney shows". These observations motivate us to include user feedback when modeling the search intents of search engine users.

Various research works have been carried out to tackle the task of query suggestion. Among them, neural network based models have made impressive progress over the past few years [5, 6, 38]. Deep neural network models mimic the learning process of human brains. Training these neural network models generally requires a large amount of training data to achieve excellent performance. However, for search related tasks, search queries involved are very sparse, and generally follow a long-tail distribution. Thus, those tail queries, with low probabilities in the data distribution, lack abundant training supports. In addition, the suggested queries are highly relevant to each other in most cases because they are all related to the search query in semantics. Distinguishing suggestions to be clicked from the ones that won't be clicked and ranking them take extreme efforts. Therefore, there tends to be a great performance gap between training and test. The trained model can be very sensitive to unseen perturbed queries. This is because queries are very short and a subtle perturbation on a query can lead to different or even the opposite search intent. For example, considering two queries "Obama son" and "son Obama", the first query is asking who is the son of Obama, while the second one, with the two terms order exchanged, is more likely to inquire who is the father of Obama. Other perturbations include adding/deleting characters/terms from queries, typos in the query, and perturbations that are too small to be expressed on text levels but significant enough to cause the suggestion system to rank improperly. In general, human perception and cognition are robust to a vast range of nuisance perturbations. However, neural networks are currently far from achieving the same level of tolerance of such perturbations [46]. This motivates us to investigate the robustness property of modern query suggestion systems and propose an appropriate method to achieve high tolerance.

Before we proceed to describe the details of the proposed approach, we enumerate the key properties that a good query suggestion system should observe:

- [**Context-Awareness**] The recommender should be aware of the sequential search queries issued by the user. These sequential queries implicitly inform the search intent of the user.
- [**Feedback-Awareness**] The recommender should also be aware of user clicks. User clicks, as an additional information source, can potentially reduce query ambiguities and allow us to understand user search intent comprehensively and more accurately.
- [**Robustness**] The recommender should increase the generalization of the original training data and improve the resistance to nuisance perturbations to the extent as extreme as possible. This reduces the performance gap between training and test, and provides favorable rankings of suggestions robustly.

A search sequence S, represented as a sequence of search queries $< Q_1^S, Q_2^S, ..., Q_{M_1}^S >$, is submitted successively by a single user within a time interval. Each query in the search query is associated with corresponding click-through information (if happens), which is a set of clicked suggestions for that query. A clicked suggestion sequence $C, < Q_1^C, Q_2^C, ..., Q_{M_2}^C >$, is formed by ordering the clicked suggestions increasingly by the time the clicks happens. The goal of this paper is: Given a set of candidate queries $\mathcal{Q}_{\text{can}}$, we would like to generate a ranking of candidates, with each candidate $Q_{\text{can}} \in \mathcal{Q}_{\text{can}}$, so that the ones to be clicked rank as high as possible.

Take the first search session in Table 5.1 as an illustrative example. Given the search query sequence $Q = <$"apple", "apple price"$>$, the clicked suggestion sequence $C = <$"apple stock"$>$, and a set of suggestion candidates $\mathcal{Q}_{\text{can}}$ composed of "apple price targets", "iphone apple price", "apple price per share", "apple", "apple fruit price", "apple stock price", "does apple price match", and "apple price today", we would like to generate a ranking of the queries in $\mathcal{Q}_{\text{can}}$ such that the clicked suggestion "apple price per share" ranks in the first place.

Table 5.1: Sample search log records

UserID	Query String	Time	Clicked Suggestion	Suggested Queries
A	apple	2018-07-01 09:10:01	apple stock	apple store, **apple stock**, apple.com, apple iphone xs max
				apple id, itunes, icloud, google maps
A	apple price	2018-07-01 09:11:03	apple price per share	apple price targets, iphone apple price, **apple price per share**, apple
				apple fruit price, apple stock price, does apple price match, apple price today
B	disney	2018-07-03 12:12:02	disney cartoons	walt disney world, disney world tickets, disney world, disney experience
				disney orlando, **disney cartoons**, disney store, disney junior
B	disney shows	2018-07-03 12:22:20	disney channel cartoons	disney shows list, disney xd shows, disney tv shows, popular disney shows
				disney channel cartoons, list of disney tv series, 2018 disney channel shows

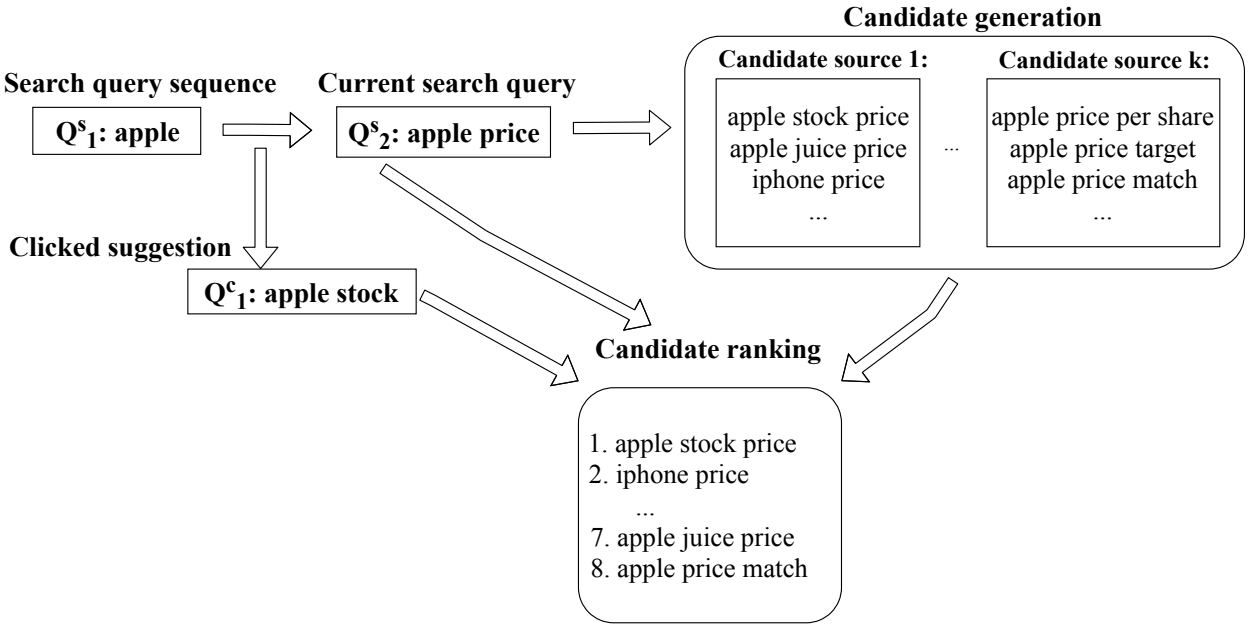


Figure 5.1: Query suggestion pipeline

In the following sections, we discuss how to make query suggestions based on the search and click interactions between users and search engines. To achieve effective and efficient query suggestions, we decompose the task into two components. First, based on the search query, we generate a set of relevant queries as suggestion candidates. These suggestion candidates are informative to cover different interpretations and aspects of the search query. Second, we rank the suggestion candidates based user's entire search query sequence and clicked feedback. Only the top ones are selected and shown to the user as query suggestions. Figure 5.1 shows the framework of *CFAN* using the search sequence <"apple", "apple

price"> as an example. After a user first searched "apple" and clicked a suggestion "apple stock", the user issued a second query "apple price" as the current search query. To make suggestions for "apple price", we first generate suggestion candidates from multiple sources. These suggestion candidates are further ranked based on entire search sequence and the clicked suggestion "apple stock".

5.2 Candidate Generation

Given a search query Q, the candidate generation component is responsible for constructing an informative set of expressive queries $\mathcal{Q}_{\text{can}}$ as suggestion candidates. An expressive suggestion candidate clearly and unambiguously describes the search intent of the user. An informative set of suggestion candidates are queries, that are diversified to cover different interpretations and aspects of the search query.

To generate such diversified informative and expressive suggestion candidates, multiple sources can be incorporated, such as search sessions and contextual information in search logs, related keywords from trending news, etc. in this work, we incorporate two main sources based on the search log. The first key source generates suggestion candidates based on query dependencies in search sessions. Given a search query, the follow-up search queries in the same search session tend to be more expressive and informative. Therefore they are adopted as suggestion candidates. As shown in Table 5.1, "apple price" is searched after "apple" in the first search session. Therefore "apple price" qualifies as a suggestion candidate for query "apple". Another key source for candidate generation is based on prefix matching between the search query and historical search queries in the search log. For example, "apple stock" qualifies as a suggestion candidate for query "apple" since the search query "apple" is a valid prefix of "apple stock".

After generating suggestion candidates, the candidates are further ranked based on a combination of statistical features, i.e., the total number of submissions within last week, last month, and last year. Only top suggestion candidates are kept for further ranking in the

Table 5.2: Training instances constructed after user A searched "apple", "apple price" and clicked "apple stock".

type	issued queries Q	clicked queries C	suggestion candidate Q_{can}
Ins^+	apple, apple price	apple stock	apple price per share
Ins^-	apple, apple price	apple stock	apple price targets
Ins^-	apple, apple price	apple stock	iphone apple price
Ins^-	apple, apple price	apple stock	does apple price match
Ins^-	apple, apple price	apple stock	apple fruit price
Ins^-	apple, apple price	apple stock	apple stock price
Ins^-	apple, apple price	apple stock	apple
Ins^-	apple, apple price	apple stock	apple price today

candidate ranking component (see Section 5.3), while others are filtered out. The purpose of filtering out these less frequent suggestion candidates is to not only generate popular and fresh candidates, but also guarantee that recommendation services respond in real time. In a nutshell, given an input query Q, the candidate generation component will return a set of relevant queries as suggestion candidates.

5.3 Candidate Ranking

In this section, we will discuss how we rank the suggestion candidates by mining the historical interactions between users and search engines. To better explain the ranking process, we first discuss the construction of training instances. Then, we go into details on how to utilize these instances to train *CFAN*.

Training instances are constructed from the historical search logs. Each instance contains a sequence of user issued queries S, a sequence of user clicked queries C, and a candidate suggested query Q_{can}. Two types of instances are constructed, i.e., positive instances and negative instances, denoted as Ins^+ and Ins^-, respectively. A positive instance is constructed if S and C jointly lead to the click of Q subsequently, while a negative instance is constructed

if a query $\tilde{Q}$ is not clicked after S and C. In particular, to generate strong negative instances, only query $\tilde{Q}$, shown together with Q after searching the last query in S, is considered as a candidate query in the construction of negative instances. Positive instances Ins^+ and negative instances Ins^- jointly define the overall matching and ranking performance among a search query sequence S, a clicked suggestion sequence C, and a set of suggestion candidates $\mathcal{Q}_{\text{can}}$. We further define such a set of Ins^+ and Ins^- as a training view, denoted as V. Table 5.2 shows the training instances of a training view constructed after user A searched queries "apple", "apple price", and clicked "apple stock" based on the search log shown in Table 5.1. In this training view, eight instances are constructed. There is only one positive instance, since the user clicked only one suggestion "apple price per share" from the suggestion list, leaving the other seven suggestions unclicked. Each of the unclicked suggestions forms a negative instance.

Note that given a sequence of search queries S and a sequence of clicked suggestions C, only if at least one suggested queries are clicked after the submission of the last query in S, a training view V will be constructed. Positive instance Ins^+ and negative instance Ins^- in V are constructed based on whether the corresponding suggestion is clicked or not, respectively.

The clicks in the constructed training views allow us to tap into the wisdom of the crowds and provide a more favorable ranking of suggestions when running into the same or similar search and click scenarios. Given such scenarios, we expect the previously clicked suggestions rank ahead of those unclicked ones. To achieve this goal, Equation 5.1 shows the ranking loss function in *CFAN* over one training view V.

$$\text{Loss}_{\text{ranking}}(\Theta) = -\log(1 + \frac{\sum\limits_{\mathbf{ins}^+ \in V} g_\Theta(\mathbf{ins}^+)}{\sum\limits_{\mathbf{ins}^+ \in V} g_\Theta(\mathbf{ins}^+) + \sum\limits_{\mathbf{ins}^- \in V} g_\Theta(\mathbf{ins}^-)}), \tag{5.1}$$

where Θ is a set of parameters defining the ranking model, and $g_\Theta(\mathbf{ins})$ is a scoring function, which gives the ranking score of an instance. We use $\mathbf{ins}/\mathbf{ins}^+/\mathbf{ins}^-$ to represent the embedding of a general/positive/negative instance, respectively. By minimizing Equation 5.1 over all training views, $g_\Theta(\mathbf{ins}^+)$ is trained to have a high ranking score for clicked suggestions,

while for unclicked suggestions, $g_\Theta(\mathbf{ins}^-)$ is trained to have a low score.

In the following paragraphs, we will discuss how to represent an training instance Ins using searched queries S, and clicked queries C, and a target suggestion Q. After that, we will discuss how to model the ranking function $g_\Theta(\mathbf{ins})$, and the training and optimization of *CFAN*.

5.3.1 Search Intent Encoder

As we mentioned in the introduction, a user's search queries and clicked queries jointly help pinpoint his/her information need. A good search intent encoder is expected to incorporate both of them when modeling search intents. Note that both user search queries and clicked queries are inherently a sequence of queries; therefore, we first discuss how to model a single query through a query encoder. Then, we move on to the discussion of sequential query modeling through a query sequence encoder.

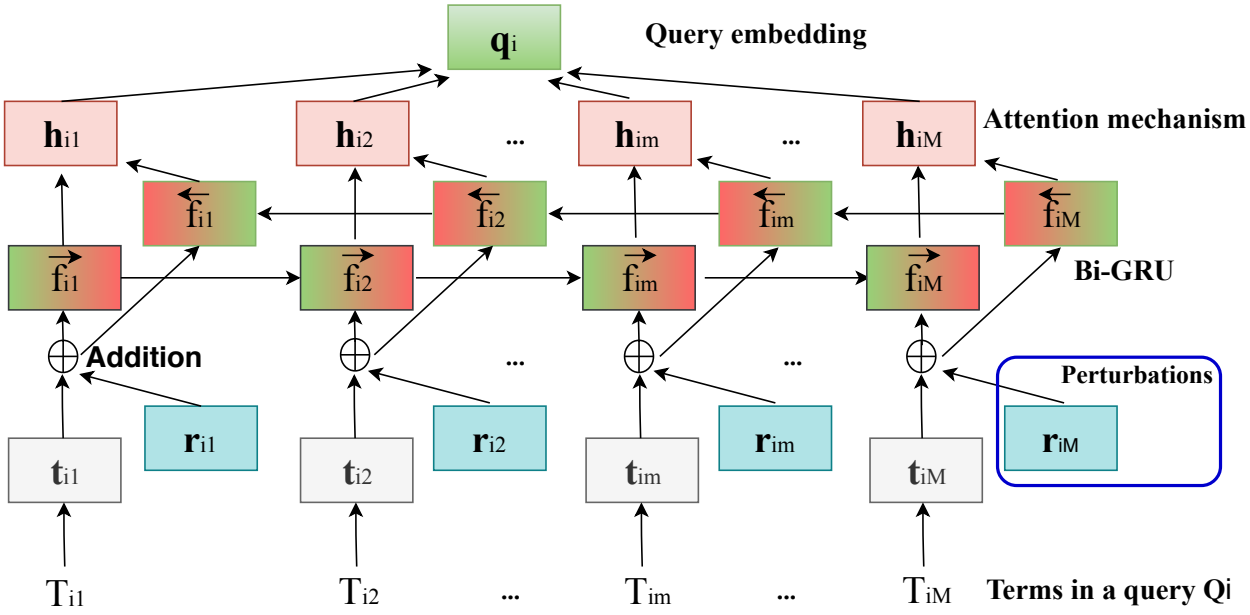


Figure 5.2: Query encoder with perturbed embeddings.

[**Query Encoder.**] Generally, queries contain different numbers of terms. In addition, different terms in a query contribute unequally to the meaning of the query. For example,

to search for the height of Barack Obama, users may formulate queries such as "how tall is Barack Obama", "height of Barack Obama", or even "Obama height". These three queries contain five, four, and two terms, respectively. Consider term importance in the first query, "tall", "Barack", and "Obama", these three terms, contribute more to understand the search intent of the query than the other terms, i.e., "how" and "is", which are generally less informative. In this work, we apply bidirectional GRU [91] and attention mechanism [86] to derive embeddings for a query Q.

Figure 5.2 shows the architecture of the query encoder. Given a query Q_i, composed of a sequence of terms $< T_{i1}, ..., T_{im}, ..., T_{iM} >$, where M is the number of terms in Q_i, we first embed each term T_{im} to an vector $\boldsymbol{t}_{im}$ through an embedding matrix as shown in Equation 5.2. To train robust models, adversarial examples are introduced by adding perturbed embeddings $\boldsymbol{r}$ into $\boldsymbol{t}$ (See more details in Section 5.3.2). We then use a bidirectional GRU to get the query embedding by summarizing information from both directions for terms in a query. The bidirectional GRU contains a forward GRU $\overrightarrow{f}$ which reads query Q_i from T_{i1} to T_{iM} and a backward GRU $\overleftarrow{f}$ which reads from T_{iM} to T_{i1}:

$$\boldsymbol{t}_{im} = Emb_t(T_{im}), m \in [1, M]. \tag{5.2}$$

$$\overrightarrow{\boldsymbol{f}}_{im} = \overrightarrow{GRU}(\boldsymbol{t}_{im}), m \in [1, M]. \tag{5.3}$$

$$\overleftarrow{\boldsymbol{f}}_{im} = \overleftarrow{GRU}(\boldsymbol{t}_{im}), m \in [M, 1]. \tag{5.4}$$

We obtain a representation $\boldsymbol{h}_{im}$ for each term T_{im} in query Q_i by concatenating the forward hidden state $\overrightarrow{\boldsymbol{h}}_{im}$ and backward hidden state $\overleftarrow{\boldsymbol{h}}_{im}$, i.e., $\boldsymbol{h}_{im} = [\overrightarrow{\boldsymbol{h}}_{im}; \overleftarrow{\boldsymbol{h}}_{im}]$, which summarizes the information of the whole query but centered around term T_{im}.

Since not all terms in a query contribute equally to the search intent embedded in the search query, attention mechanism is applied to extract such informative contributing terms and aggregate the embeddings of these terms to construct a query vector.

$$u_{im} = tanh(\boldsymbol{W}_w \boldsymbol{h}_{im} + b_w). \tag{5.5}$$

$$\alpha_{im} = \frac{\exp(u_{im}^T u_w)}{\sum_m \exp(u_{im}^T u_w)}. \tag{5.6}$$

$$\boldsymbol{q}_i = \sum_m \alpha_{im} \boldsymbol{h}_{im}. \tag{5.7}$$

Equations 5.5, 5.6, and 5.7 show the attention process on the query term level. The term embedding $\boldsymbol{h}_{im}$ is first fed into a one-layer MLP to get u_{im}. Then, u_{im} is further used to derive the normalized importance weight α_{im} through a softmax function. Finally, the query vector $\boldsymbol{q}_i$ is computed as a weighted sum of the term embeddings based on the weights.

[**Query Sequence Encoder.**] A query sequence is composed of a sequence of queries $< Q1, ...,Q_i, ..., Q_N >$, where N is the number of queries in the sequence. Both user search queries and clicked queries are essentially query sequences. Given a query sequence $< Q1, ...,Q_i, ..., Q_N >$, the query sequence encoder is expected to encode the hidden search intent in it. Similar to the structure of query encoder, the query sequence encoder contains a bidirectional GRU layer and an attention layer. The differences are two-fold. First, the query sequence encoder takes query embeddings as inputs. Second, there are no perturbations in query sequence encoder. Formally, a bidirectional GRU is first applied to encode the query sequence:

$$\overrightarrow{\boldsymbol{f}}_i = \overrightarrow{GRU}(\boldsymbol{q}_i), i \in [1, N], \tag{5.8}$$

$$\overleftarrow{\boldsymbol{f}}_i = \overleftarrow{GRU}(\boldsymbol{q}_i), i \in [N, 1], \tag{5.9}$$

The representation of a query Q_i in the sequence is constructed by concatenating $\overrightarrow{\boldsymbol{f}}_i$ and $\overleftarrow{\boldsymbol{f}}_i$, i.e., $\boldsymbol{h}_i = [\overrightarrow{\boldsymbol{h}}_i; \overleftarrow{\boldsymbol{h}}_i]$. $\boldsymbol{h}_i$ summarizes the neighbor queries around query Q_i but still focus on the meaning of query Q_i.

To reward queries that pinpoint the search intent of the user, query level attention mechanism is applied.

$$u_i = tanh(\boldsymbol{W}_s \boldsymbol{h}_i + b_s). \tag{5.10}$$

$$\alpha_i = \frac{\exp(u_i^T u_s)}{\sum_i \exp(u_i^T u_s)}. \tag{5.11}$$

$$\boldsymbol{s} = \sum_i \alpha_i \boldsymbol{h}_i. \tag{5.12}$$

Equations 5.10, 5.11, and 5.12 summarizes the attention process on query level. Finally, $\boldsymbol{s}$ gives the query sequence embedding that captures the search intent embedded in sequential

queries $< Q1, ..., Q_i, ..., Q_N >$. In a nutshell, given a query sequence, we first use query encoder to derive the embedding for each query in the sequence. Then, we use query sequence encoder to further construct the embedding of the query sequence.

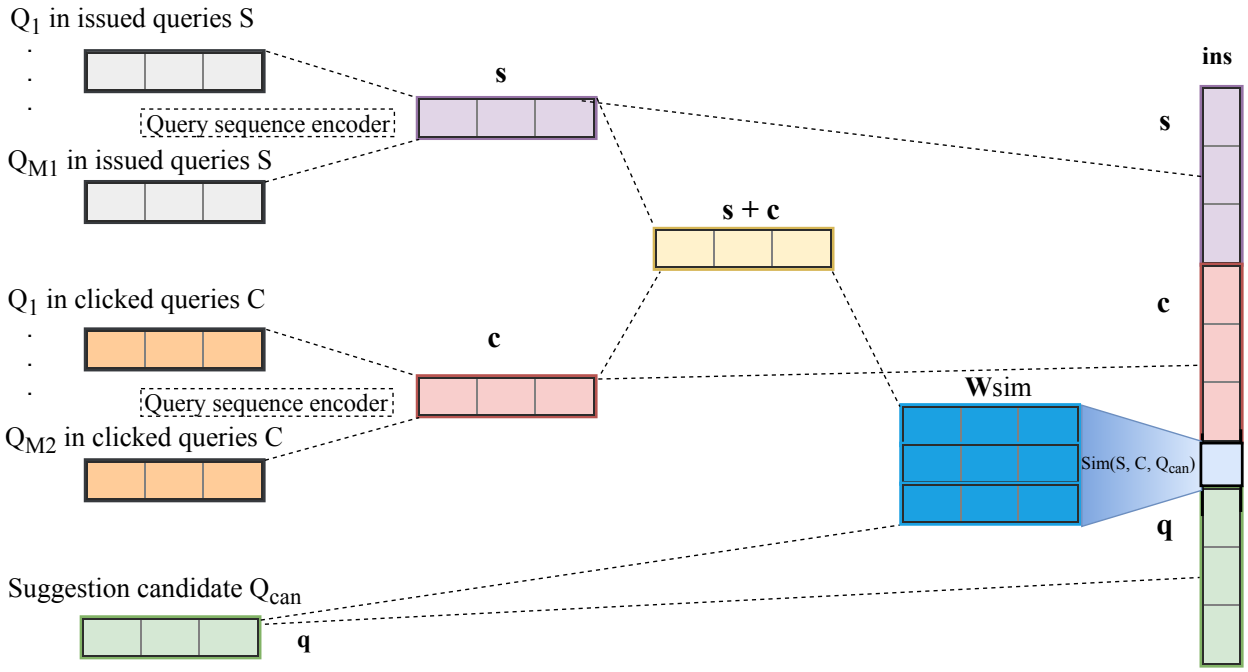


Figure 5.3: Training instance construction based on search sequence S, clicked sequence C, and suggestion candidate Q_{can}.

Given a training instance Ins, which is composed of a search query sequence S, a clicked query sequence C, and a suggestion candidate Q_{can}, we use the same query sequence encoder to derive the embeddings for S and C, denoted as $\boldsymbol{s}$ and $\boldsymbol{c}$, respectively, and we use a query encoder to encode the candidate Q_{can}, with its embedding denoted as $\boldsymbol{q}$. We use the concatenation of $\boldsymbol{s}$, $\boldsymbol{c}$, and $\boldsymbol{q}$, i.e., $[\boldsymbol{s}; \boldsymbol{c}; \boldsymbol{q}]$ to get the embedding **ins** to represent the instance Ins. Figure 5.3 summarizes the process.

5.3.1.1 Matching query sequences and suggestion query

Given a sequence of issued queries S, a sequence of clicked queries C, and a suggestion candidate Q_{can}, we aim to explicitly derive the matching among S, C, and Q_{can}. We follow

approaches [92] and [93]. We define the matching score among $\boldsymbol{s}$, $\boldsymbol{c}$, and $\boldsymbol{q}$ as follows:

$$\mathrm{Sim}(S, C, Q_{\mathrm{can}}) = (\boldsymbol{s} + \boldsymbol{c})^T \boldsymbol{W}_{\mathrm{sim}} \boldsymbol{q}, \tag{5.13}$$

where $\boldsymbol{W}_{\mathrm{sim}} \in \mathbb{R}^{|\boldsymbol{s}| \times |\boldsymbol{q}|}$ is a similarity matrix. In this module, $\boldsymbol{s} + \boldsymbol{c}$ explicitly combines the search intent embedded in search queries S and clicked suggestions C. We seek a transformation of the candidate query Q_{can} that is the closest to $\boldsymbol{s} + \boldsymbol{c}$. The similarity matrix $\boldsymbol{W}_{\mathrm{sim}}$ is a parameter of the network and is optimized during training. $\mathrm{Sim}(S, C, Q_{\mathrm{can}})$ is then explicitly appended to **ins** as a feature.

5.3.2 Learning and Optimization

Adversarial training is a novel technique for training models to improve robustness to small, approximately worst case perturbations. The adversarial training process can be viewed as minimizing the worst case error when the training instances are perturbed by an adversary. It can be interpreted as learning to play an adversarial game, trying to minimize an upper bound on the expected loss over noisy instances. Adversarial training can also be viewed as a form of active learning, where the model actively requests labels on new instances. In the case of adversarial training, the new instances are constructed by introducing perturbations to existing instances. In addition, the human labelers are replaced with a heuristic labeler that copies labels from nearby instances. In the following paragraphs, we will discuss how we construct adversarial instances and apply adversarial training on *CFAN*.

When learning *CFAN* without adversarial examples, *CFAN* is optimized by learning only the ranker, formulated as finding a set of parameters Θ, that minimize an empirical ranking loss as shown in Equation 5.1 for a given set of training views. In adversarial training, we construct adversarial perturbations on training instances in order to cause a misbehavior of *CFAN* on a training view V. The misbehavior refers to the event that *CFAN* ranks unclicked suggestions ahead of clicked suggestions. To construct such adversarial perturbations, we introduce a binary classifier, which learns the labels of training instances in V. For each instance, the loss function $\mathrm{Loss}_{\mathrm{cls}}(\Theta)$ of the classifier focuses on reducing the binary cross-

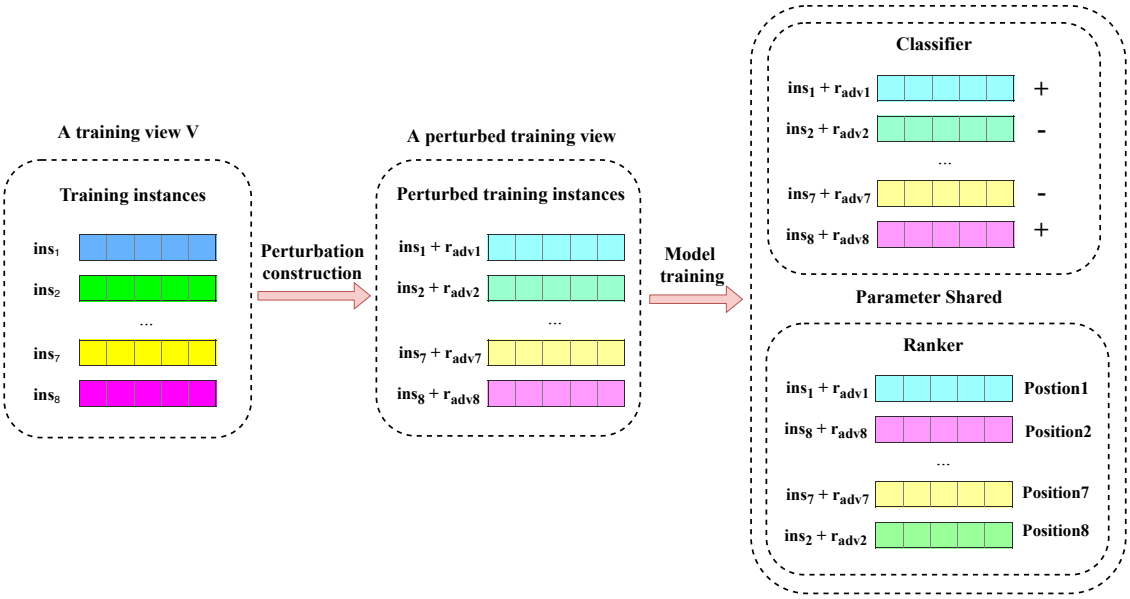


Figure 5.4: The training framework of *CFAN*.

entropy [94] between the predicted probabilistic score $\hat{y}^{\Theta}_{\mathbf{ins}}$ and the gold standard $y_{\mathbf{ins}}$ as follows:

$$\text{Loss}_{\text{cls}}(\Theta) = -[y_{\mathbf{ins}} \log(\hat{y}^{\Theta}_{\mathbf{ins}}) + (1 - y_{\mathbf{ins}}) \log(1 - \hat{y}^{\Theta}_{\mathbf{ins}})], \tag{5.14}$$

where $\hat{y}^{\Theta}_{\mathbf{ins}} = g_{\Theta}(\mathbf{ins})$.

To increase the classification difficulty, we aim to identify a bounded perturbation $\boldsymbol{r}_{\text{adv}}$ for each $\mathbf{ins}$ such that the classification loss on $\mathbf{ins} + \boldsymbol{r}_{\text{adv}}$ is as large as possible. The adversarial loss of the classifier becomes:

$$\text{Loss}_{\text{cls adv}}(\Theta) = -[y_{\mathbf{ins}} \log(\hat{y}^{\Theta}_{\mathbf{ins}+\boldsymbol{r}_{\text{adv}}}) + (1 - y_{\mathbf{ins}}) \log(1 - \hat{y}^{\Theta}_{\mathbf{ins}+\boldsymbol{r}_{\text{adv}}})], \tag{5.15}$$

where

$$\boldsymbol{r}_{\text{adv}} = \arg \min_{\boldsymbol{r}, ||\boldsymbol{r}|| \leq \epsilon} [y_{\mathbf{ins}} \log(\hat{y}^{\hat{\Theta}}_{\mathbf{ins}+\boldsymbol{r}}) + (1 - y_{\mathbf{ins}}) \log(1 - \hat{y}^{\hat{\Theta}}_{\mathbf{ins}+\boldsymbol{r}})]. \tag{5.16}$$

$\hat{\Theta}$ is the set of current parameters defining the classifier, $\boldsymbol{r}$ is a perturbation on the input $\mathbf{ins}$, $\hat{y}^{\hat{\Theta}}_{\mathbf{ins}+\boldsymbol{r}}$ gives the predicted score of the perturbed instance based on the current set of parameters, and ϵ is a parameter to bound the perturbations.

Calculating the exact value of $\boldsymbol{r}_{\mathbf{adv}}$ is not feasible, because exact minimization of Equation 5.16 with respect to $\boldsymbol{r}$ is intractable. To derive dynamic perturbations efficiently, we

apply fast gradient sign method [47] to approximate the perturbation $\boldsymbol{r}_{\mathbf{adv}}$. We linearize the cost function 5.16 around the current value of $\hat{\Theta}$, and obtain an optimal max-norm constrained perturbation of

$$\boldsymbol{r}_{\mathbf{adv}} = -\epsilon \boldsymbol{l}/||\boldsymbol{l}||_2, \tag{5.17}$$

where

$$\boldsymbol{l} = \nabla_{\mathbf{ins}} \text{Loss}_{\text{cls}}(\hat{\Theta}). \tag{5.18}$$

Based on Equations 5.17 and 5.18, the perturbation for each instance can be calculated efficiently via back-propagation. Adversarial instances can be constructed by adding the derived perturbations $\boldsymbol{r}_{\text{adv}}$ to the corresponding instance embeddings **ins**.

To train a robust suggestion system with good ranking performance, the objective of *CFAN* combines the loss functions of both adversarial classification and ranking as follows:

$$\text{loss}(\Theta) = \sum_{\text{ins} \in V} \text{loss}_{\text{cls adv}}(\Theta) + \text{loss}_{\text{ranking}}(\Theta). \tag{5.19}$$

The classification component and the ranking component share the same set of parameters. The classification component dynamically generates adversarial instances, while the ranking component generates favorable rankings of query suggestions, pushing clicked suggestions to top positions and pulling unclicked suggestions to bottom positions in ranking lists. Figure 5.4 shows the training framework of *CFAN*. At each step of training, we first identify the worst case perturbations $\boldsymbol{r}_{\text{adv}}$ for each training instance against the current model as shown in Equation 5.16. Then, perturbed instances are constructed by adding the perturbations to the corresponding instances. At last, we optimize the classifier and the ranker simultaneously on the perturbed instances through minimizing Equation 5.19 with respect to Θ.

As shown in Equations 5.17 and 5.18, easy classified instances with small classification losses tend to get large perturbations while difficult classified instances with large classification losses tend to get small perturbations. When training the classifier and the ranker simultaneously as a multi-task job in Equation 5.19, perturbed instances not only increase the difficulty for the classifier to make accurate predictions, but also drive the ranker to make improper behaviors, i.e., ranking unclicked suggestions ahead of clicked ones. These

adversarial perturbed instances regularize *CFAN* to be insensitive to changes in the inputs (i.e., **ins**) with perturbations smaller than the bound ϵ. More precisely, optimizing these adversarial perturbed instances allows *CFAN* to be aware of ranking tough instances. This is particularly important for modern suggestion systems since suggestion candidates are highly semantically relevant to the search query in most cases. Training with adversarial examples equip *CFAN* with the power to distinguish such tangled suggestion candidates, which, in return, yields a robust model with good ranking performance.

CHAPTER 6

Automatic Speaker Recognition with Metric Learning-based Few-shot Learning

6.1 Background

Within the last couple of years, voice has become one of the most ever-growing media through which people interact with their devices. For instance, over 47 million people in the United States own a smart home device while 23% of the Britons have a voice-controllable digital assistant at home in 2018 [95, 96]. To ignite the interactions between smart devices and their owners, automatic speaker recognition (ASR) plays an important role to determine the speaker identity based on a short piece of audio. Moreover, the capability of ASR comes with a wide range of applications, such as biometric authentication [97], forensics [98], and personalized services in electronics [99]. In particular, the text-independent ASR with only acoustic information is the most general and non-trial task, which can be used in everyday situations. In text-independent ASR, an arbitrary utterance from one of the known speakers in training set will be given and the system needs to identify which speaker the utterance belongs to.

Deep learning-based ASR methods are gaining popularity due to strong model capacities and superior performance [7, 8, 10, 11]. Most incremental improvements in existing deep learning methods rely on the use of deeper and more complex models with massive training data. More specifically, there are two inherent limitations for existing approaches. First, increasing model complexity is not always desirable in practice because of the greater costs of computation and storage. It thus becomes expensive to get such methods deployed in

smart devices to provide offline services. Second, acquiring sufficient labeled training data for all speakers is impractical [100] while the lack of training supports can lead to worse generalization and high vulnerability to tiny perturbations for existing deep learning-based ASR methods [101,102]. Hence, developing effective techniques for ASR with limited training data remains a daunting task.

To achieve remarkable performance with limited training data, meta-learning is one of the most promising approaches to comprehensively utilize the limited training instances. More specifically, meta-learning systematically observes how machine learning approaches perform on a wide range of similar learning tasks, and then learns to learn new tasks more efficiently [84]. In particular, few-shot learning is a contemporary meta-learning approach that introduces an auxiliary meta-learning phase to generalize and share transferable knowledge across tasks. To learn from extremely limited data, one type of few-shot learning, based on metric learning, looks to light-parametric models, which learn a distance metric among training instances rather than myriad model parameters [57]. More precisely, the essential knowledge can be learned and memorized by reasoning the distance metric between instances in a support module and a query module. Instances in the support module are labeled instances, thereby serving as references. Based on the reference instances, the query instances are then able to conduct reasoning. Finally, metric-learning-based few-shot learning models can be optimized by iterative comparisons between support and query instances such that instances from the same speaker are embedded as close to each other as possible in the hidden space and as far as possible from instances of the other speakers.

To comprehensively exploit the training instances, an alternative way is to generate augmented data based on the training set. Different from conventional methods that separately augment data apart from the training process, we construct augmented data automatically by leveraging adversarial training. In particular, we construct dynamic perturbations at the embedding level to form adversarial examples. These adversarial examples are formed by applying small but intentional perturbations to inputs from the dataset. Specifically, these adversarial examples can be treated as ultimate data augmentation as specific perturba-

tions are created to best fool the model. Accordingly, the model trained in an adversarial manner can not only learn from the original static training data but also improve based on the dynamically constructed perturbed data. As a result, adversarial training significantly improves the robustness of ASR models and achieves out-of-instance generalization while the robustness is crucial for the security-sensitive ASR task. In a nutshell, data augmentation through adversarial training provides another effective solution to thoroughly utilize the training instances and train models resistant of nuisance perturbations to achieve high generalizations in both training and test.

In this work, we study the problem of speaker identification with a shortage of training data. In essence, we address the data deficiency issue by applying few-shot learning and adversarial training. To be more specific, the main contributions of this work are as follows:

- Different from conventional neural network-based methods, which rely on the availability of a sufficient amount of training data to achieve high identification performance, we model it as a few-shot learning problem to conquer the data deficiency.
- To further improve the generalization of the model, we employ adversarial training. Adversarial examples serve as dynamic augmented data, the optimization of which results in a more generalized and robust speaker recognition system.
- We present a comprehensive empirical evaluation of our approach on a real-world dataset. The experimental results show that our approach, *AFEASI*, significantly outperforms 11 conventional baseline methods in speaker recognition.

Given a short piece of audio x and its mel frequency cepstral coefficients (MFCCs) $\boldsymbol{m}_x$ as features, the goal of this paper is to recognize the speaker identity y among a set of known speakers. In particular, in this work we focus on text-independent automatic speaker identification by leaning from limited pieces of training audios. To achieve this goal, we strive to thoroughly utilize the limited instances during training by leveraging few-shot learning and adversarial training.

6.2 Framework Overview

In this paper, a metric-learning-based few-shot learning pipeline is applied to perform N-shot learning for previously rare speakers. More precisely, the model is capable of recognizing a previously rare speaker after having examined only N examples, where N is a small number.

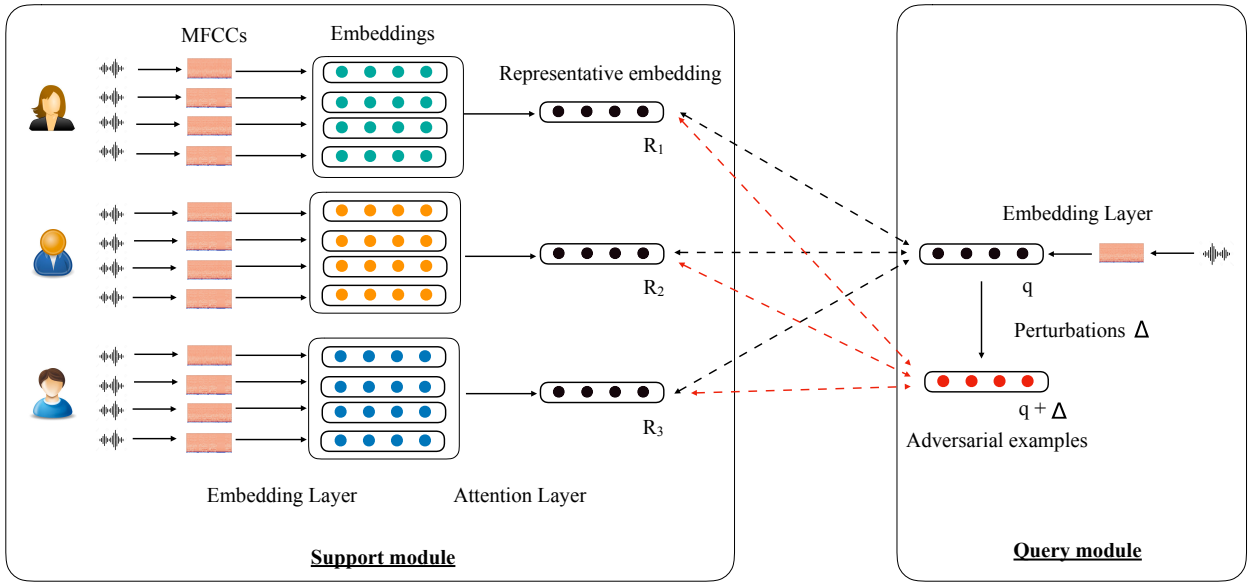


Figure 6.1: The overall framework of AFEASI.

Figure 6.1 shows the framework of AFEASI that performs speaker identification by conducting N-shot, K-way classification tasks with a support set of K different speakers and N training audio instances for each speaker in the support set. In addition, a set of query audio instances is given for prediction. Note that although Figure 6.1 shows only one query instance for illustration simplicity, AFEASI can cope with multiple query audio instances. For each audio instance x, AFEASI first extracts the mel frequency cepstral coefficients (MFCCs) [103] as acoustic features $\boldsymbol{m}_x$, thereby deriving a fixed-length vector as the audio embedding $\boldsymbol{E}_x$ with an embedding layer. Based on the embeddings of audio instances, an aggregated embedding is constructed as the representative for each speaker in the support module. *AFEASI* then optimizes the distances between the embeddings of the query instances and the representatives of the corresponding speakers so that the representatives can be applied to recognize the speaker identity. The process of the optimization can be

summarized as finding a distance metric into a space in which instances of the same speaker are embedded as close to each other as possible and as far as possible from instances of the other speakers. To further comprehensively utilize the training data, we introduce dynamic adversarial perturbations on the query instances to enhance the generalization of *AFEASI* through improving its robustness against unseen instances. To better visualize this part, adversarial learning is highlighted in red in the framework.

6.3 Embedding Representation Learning

In this section, we discuss how to construct an embedding given a piece of audio x_i.

We first convert the audio signal into frequency domains by constructing the mel frequency cepstral coefficients (MFCCs) [103] as acoustic features, which is denoted as $\boldsymbol{m}_{x_i}$. A 2D-convolutional layer is first utilized to extract informative features from the raw MFCC. Then the resulting feature maps are fed into an activation layer to introduce non-linearity. We further employ residual shortcut connection [87] to derive the representations for the audio MFCC. Equation 6.1 summarizes the key operations as follows:

$$\boldsymbol{C}_1 = \text{Relu}(\text{Relu}(\text{Conv}_1(\boldsymbol{m}_{x_i})) + \boldsymbol{m}_{x_i}), \tag{6.1}$$

where $\text{Relu}(\cdot)$ and $\text{Conv}_1(\cdot)$ are the activation layer and the 2D-convolutional layer, respectively. To comprehensively distill the local features, we repeat the above residual-based covolutional operations for H times as:

$$\boldsymbol{C}_h = \text{Relu}(\text{Relu}(\text{Conv}_h(\boldsymbol{C}_{h-1})) + \boldsymbol{C}_{h-1}), h > 1, \tag{6.2}$$

where $\boldsymbol{C}_h$ is the feature maps at the h-th convolutional layer. Finally, the embedding $\boldsymbol{E}_{x_i}$ can be constructed by flattening the feature maps $\boldsymbol{C}_H$ at the H-th convolutional layer, thereby serving as the representation of the input audio x_i.

6.4 Representative Embedding Construction

As shown in the support module of the framework, for each speaker, we aim to derive a representative embedding, which summarizes the acoustic biometric of the speaker. We develop an aggregation attention layer to learn the importance weights across each audio embedding of a particular speaker. Formally, the aggregation attention layer can be represented as follows:

$$\alpha_i = \text{softmax}(\boldsymbol{c} \cdot \tanh(\boldsymbol{W} \cdot \boldsymbol{E}_{x_i} + \boldsymbol{b})), \tag{6.3}$$

$$\boldsymbol{E}_R = \sum_i \alpha_i \boldsymbol{E}_{x_i}, \tag{6.4}$$

where $\boldsymbol{W}$ and $\boldsymbol{b}$ are the parameters for computing the attention weights α_i. Each audio embedding $\boldsymbol{E}_{x_i}$ is first fed into a one-layer neural network. Its output, together with the context vector $\boldsymbol{c}$, are further utilized to generate the importance weight α_i for each audio embedding $\boldsymbol{E}_{x_i}$ through a softmax function. The aggregated embedding $\boldsymbol{E}_R$ is calculated as a weighted sum of the audio embeddings based on the learned importance weights.

6.5 Few-Shot Learning

In this section, we discuss how to model the speaker identification task as a few-shot learning problem. A metric learning-based few-shot learning framework is employed in this work, which is composed of two modules, i.e., a support module and a query module. As shown in Figure 6.1, we first randomly sample a set of speakers from the training set as the start to construct the support module. For each speaker in the support module, we further randomly sample k pieces of his audio instances and derive the corresponding MFCCs. These MFCCs are further fed into an embedding layer so we can use a fixed length vector to represent each audio instance. To comprehensively represent the acoustic feature of a speaker, we utilize the attention mechanism to aggregate his acoustic embeddings. In the query module, we randomly select a piece of audio from a speaker, which is one of the speakers in the support

module. We feed it into the embedding layer to derive the audio embedding. We then compare the distances between the query embedding and all the representative embeddings in the support module. The distances then are utilized to measure the relegation distribution over all speakers int support module. Model is optimized by such iterative comparisons and reasoning between the support and query modules.

In the comparisons and reasoning, we seek to separate audio embeddings in such a way that embeddings from different speakers are far from each other and embeddings from the same speaker are as close as possible in the hidden space. We achieve this by leveraging metric learning. In particular, the predicted probability of query q belonging to speaker k is given by:

$$p(y_k|q) = \frac{\exp(-d(q, R_k))}{\sum_{k'} \exp(-d(q, R_{k'}))}, \tag{6.5}$$

where $d(q, R_k)$ is the euclidean distance between the embedding $\boldsymbol{E}_q$ of query q and the representative embedding $\boldsymbol{E}_{R_k}$ of speaker k.

The loss function is then defined as the cross entropy between the predictions and the ground truth.

$$L(\Theta) = -\sum_k g(y_k|q) \log p(y_k|q, S, \Theta), \tag{6.6}$$

where $g(y_k|q)$ is probability that q goes to speaker k, which can be derived from the ground truth, and S denotes a set of audio representatives in the support module.

6.6 Adversarial Training

The goal of employing adversarial training is to allow the identification system not only get optimized by the instances in the training data, but also be robust to unseen adversarial perturbations. To enhance the robustness, we enforce the model to perform consistently well even when the adversarial perturbations are presented. To achieve this goal, we further optimize the model to minimize the objective function with the perturbed parameters.

Formally, we define the objective function with adversarial examples incorporated as:

$$\begin{aligned} L_{adv}(S, q|\Theta) &= L(S, q|\Theta) + \lambda L(S, q + \Delta_{adv}|\Theta), \\ \text{where } \Delta_{adv} &= \arg \max_{\Delta, \|\Delta\| \leq \epsilon} L(S, q + \Delta|\Theta), \end{aligned} \tag{6.7}$$

where Δ denotes the perturbations on the query instances, $\epsilon \geq 0$ controls the magnitude of the perturbations, and Θ denotes the model parameters. In this formulation, the adversarial term $L(S, q+\Delta_{adv}|\Theta)$ can be treated as a model regularizer, which stabilizes the identification performance. We use λ to control the strength of the adversarial regularizer, where the intermediate variable Δ maximizes the objective function to be minimized by Θ. The training process can be expressed as playing a minimax game:

$$\Theta_{opt}, \Delta_{opt} = \arg \min_{\Theta} \max_{\Delta, \|\Delta\| \leq \epsilon} L(S, q|\Theta) + \lambda L(S, q + \Delta|\Theta), \tag{6.8}$$

where the learning algorithm for model parameters Θ is the minimizing player, and the procedure to derive perturbations Δ acts as the maximizing player, which aims to identify the worst-case perturbations against the current model. The two players alternately play the game until convergence.

Constructing Adversarial Perturbations. Given a support set S and a query q, the problem of constructing adversarial perturbations Δ_{adv} is formulated as maximizing

$$\ell_{adv}(S, q|\Delta) = \sum_i g(y_i|q) \log p(y_i|q + \Delta, S, \hat{\Theta}), \tag{6.9}$$

where $\hat{\Theta}$ denotes a set of current model parameters. As it is difficult to get the exact optimal solutions of Δ_{adv}, we employ the fast gradient method proposed in [47], a common choice in adversarial training [104, 105], to estimate Δ_{adv}. The idea is to approximate the objective function around Δ as a linear function. To maximize the approximated linear function, we need to move towards the gradient direction of the objective function with respect to Δ. With the max-norm constraint $\|\Delta\| \leq \epsilon$, we approximate Δ_{adv} as:

$$\Delta_{adv} = \epsilon \frac{\tau}{\|\tau\|}, \text{ where } \tau = \frac{\partial \ell_{adv}(S, q|\Delta)}{\partial \Delta}. \tag{6.10}$$

Learning Model Parameters. We now consider how to learn model parameters Θ. The local objective function to minimize for a query q given a support set S is as follows:

$$\begin{aligned}\ell_{adv}(S, q|\Theta) = \sum_i g(y_i|q) \log p(y_i|q, S, \Theta) \\ +\lambda \sum_i g(y_i|q) \log p(y_i|q + \Delta_{adv}, S, \Theta),\end{aligned} \tag{6.11}$$

where Δ_{adv} is obtained from Equation 6.10. We can obtain the SGD update rule for Θ:

$$\Theta = \Theta - \eta \frac{\partial \ell_{adv}(S, q|\Theta)}{\partial \Theta}, \tag{6.12}$$

where η denotes the learning rate.

Algorithm 3: Parameter optimizations

Input: Training instances D, max iteration $iter_{\max}$;

Output: Model parameters Θ

Initialization: initialize Θ with Normal distribution $N(0,0.01)$, $iter = 0$, $\Theta_{opt} = \Theta$, $L_{opt} = L_{vali}$;

repeat

  foreach *support and query* S, q **do**

    // Constructing adversarial perturbations;

    $\Delta_{adv} \leftarrow$ Equation 6.10;

    // Updating model parameters;

    $\Theta \leftarrow$ Equation 6.12;

  if $L_{vali} < L_{opt}$ **then**

    $L_{opt} = L_{vali}$;

    $\Theta_{opt} = \Theta$;

  $iter + +$;

until $iter > iter_{max}$;

Return Θ_{opt};

Algorithm 3 summarizes the training process. In each training step, we randomly construct support set S and a query q. We then construct adversarial perturbations and optimize

model parameters in a sequential order. The training involves multiple training steps and stops until reaching a certain number of training epochs. The parameters achieving the best performance on the validation dataset are utilized for evaluations.

CHAPTER 7

Automatic Speaker Recognition with Gradient-based Few-shot Learning

7.1 Background

A recent report[1] in 2018 shows that smart speakers have gained an installed user base of nearly one in every four U.S. adults or 50+ million users. These smart speakers equipped with voice recognition technology, also known as speaker identification, which answers the fundamental question "Who is speaking?" The answer to the question enables various downstream applications to provide a personalized experience.

Our work presumes new users always have very limited labeled voice data, as Google Assistant and Amazon Alexa only require a new user to repeat two to four prompts for learning his/her voice. Unlike the aforementioned research where existing users and new users are treated equally, we develop a meta-learning approach targeting to expedite the learning process for recognizing new users with limited training data. We foresee the need of expediting the learning for new users as (1) smart speakers are gaining in popularity where the report also shows that 30% of users are new in 2018 and (2) the previous research [106] displayed that the length of voice history of a user is positively correlated to his/her identification accuracy.

Our proposed solution expedites the learning by transferring the knowledge learned from the existing user base with a gradient-based meta-learning tactic. We use Mixture Density

[1]https://voicebot.ai/wp-content/uploads/2018/11/voice-assistant-consumer-adoption-report-2018-voicebot.pdf

Networks (MDNs) [107] to construct acoustic user profiles in that MDNs are gradient-friendly and can model voice utterances with arbitrary lengths so that we can then apply Model-Agnostic Meta-Learning (MAML) [108] technique to achieve expeditious learning. Our experiments demonstrate that our proposed solution, *MDNML*, when having only four seconds of voice data from new users, its accuracy outperforms the best/worst baseline methods by 3.2%/5.8%.

7.2 Bridging Mixture Density Networks with Gradient-based Meta-learning

We formulate the objective of our work as the following. Suppose the system has a set of existing users with registered voice utterances as background training data. Given a set of new users, with a short registered voice utterance for each user as enrollment, and another short testing voice utterance of a user within the new user set, the goal of this study is to recognize the speaker identity behind the testing voice utterance. For simplicity, we compare model performance based on text-independent tasks and presume that new users have very limited training data [45, 106, 109], for example, one to four seconds.

To better explain how to construct users' acoustic profiles and how to transfer profiling knowledge from existing users to new users, we illustrate the framework of *MDNML* in Fig. 7.1.

7.2.1 Mixture Density Networks

Mixture density networks (MDNs) are based on a mixture density model that combines neural networks [107]. MDNs are chosen in this work to construct acoustic profiles for users as they are inherently flexible in sense that it can model voice utterances with arbitrary lengths. Moreover, assuming the voice print of a user can be sufficiently expressed by a short period of time, each tiny time frame can contribute to one training instance for the user, leading to a relatively adequate amount of training data for new users. In addition, MDNs

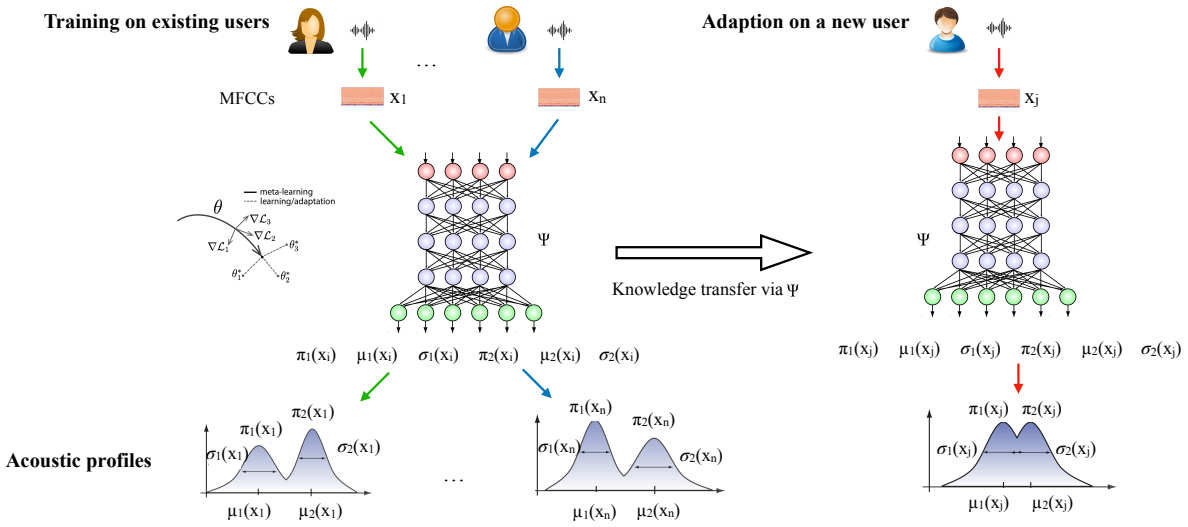


Figure 7.1: The framework of *MDNML*. During training, we learn a set of well-initialized model parameters Ψ by training acoustic profiles of all existing users. To serve new users, we construct their acoustic profiles by adapting from Ψ.

based on neural networks are gradient-friendly so that the gradient-based knowledge transfer techniques are applicable.

In this work, we utilize mel-frequency cepstral coefficients (MFCCs) [103] to represent the voice characteristics of users because MFCCs are capable of approximating the human aural systems and widely applied in various voice recognition tasks, such as speaker recognition [109–112] and speech synthesis [113–116]. More specifically, we utilize a Gaussian mixture model (GMM)-based MDN. An MDN maps a set of input MFCC features x to the parameters of a GMM (i.e., mixture weights π_m, mean μ_m, and variance σ_m^2), which in turn give a full probability density function of a MFCC feature y, conditioned on the input x and the learned model $\mathcal{M}$, $p(y \mid x, \mathcal{M})$, Formally,

$$p(y \mid x, \mathcal{M}) = \sum_{m=1}^{M} \pi_m(x) \cdot \Phi(y; \mu_m(x), \sigma_m^2(x)), \tag{7.1}$$

where M is the number of mixture components and $\pi_m(x)$, $\mu_m(x)$, and $\sigma_m^2(x)$ correspond to the mixture weight, mean, and variance of the m-th component conditioned on x.

To derive the parameters in a GMM-based MDN, MDN first converts the input x using

a multi-layer perceptron (MLP) and obtains output z as:

$$z = f_\theta(x), \tag{7.2}$$

where $f_\theta(\cdot)$ corresponds to a set of transformations in the MLP network. The total number of network outputs, i.e., the dimension of z, is $(2c+1)\times M$ where c corresponds to the dimension of the MFCC features. M corresponds to the number of mixture components in the MDN. Then, z is partitioned into three subsets $z_m^{(\pi)}$, $z_m^{(\mu)}$, and $z_m^{(\sigma)}$, which correspond to the outputs used to calculate the GMM weights, means, and standard derivations, respectively.

$$z = [z_1^{(\pi)}, ..., z_M^{(\pi)}, z_1^{(\mu)}, ..., z_M^{(\mu)}, z_1^{(\sigma)}, ..., z_M^{(\sigma)}]. \tag{7.3}$$

After the partition, each subset is passed through a set of specific transformations for conversion to the GMM weights, means, and standard derivations as:

$$\pi_m(x) = \frac{\exp(z_m^{(\pi)})}{\sum_{j=1}^{M} \exp(z_j^{(\pi)})}, \tag{7.4}$$

$$\mu_m(x) = \tanh(z_m^{(\mu)}), \tag{7.5}$$

$$\sigma_m(x) = \exp(z_m^{(\sigma)}). \tag{7.6}$$

The use of the softmax function in Equation 7.4 constrains the mixture weights to be positive and sum up to 1. Analogously, Equation 7.6 constrains the standard deviations to be positive.

During training, these density parameters are passed to a log likelihood calculator to compute the log likelihood of an MFCC feature y, which is further utilized to define the loss function for the MDN as follows:

$$L = -\sum_{n=1}^{N} \log\{\sum_{m=1}^{M} \pi_m(x) \cdot \Phi(y; \mu_m(x), \sigma_m^2(x))\}, \tag{7.7}$$

where N is the number of MFCC vectors for a user. The parameters of MDN only lie in the MLP network and these parameters are optimized in such a way that the overall negative log likelihood in Equation 7.7 is minimized.

7.2.2 Knowledge Transfer via Gradient-based Meta-learning

The effective training of MDNs relies on sufficient training data, which are usually unavailable for new users. To compensate for the data deficiency, we develop a gradient-based knowledge transfer module to leverage identification knowledge grained from recognizing existing users. More precisely, we learn a set of well-initialized model parameters over many similar tasks so that it would be easier to reach the global optimal when training a new task.

Each task corresponds to the training process of creating an acoustic profile of a user, where a profile is expressed by an MDN. We optimize a set of parameters Ψ such that when a gradient step is taken with respect to particular task t_i, the parameters θ_i, derived from Ψ, are close to the optimal parameters for task t_i, where $\theta_i = \{\pi, \mu, \sigma\}$ denotes the model parameters learned based on task t_i. Let $l(\theta_i)$ denote the loss of task t_i based on the test set of t_i. The entire loss over multiple tasks is given by:

$$L(\Psi) = \sum_{i=1} l(\theta_i). \tag{7.8}$$

To update the initialization parameters Ψ, we have:

$$\Psi \leftarrow \Psi - \alpha \nabla_{\Psi} L(\Psi). \tag{7.9}$$

To optimize each individual task t_i, we have:

$$\theta_i \leftarrow \Psi - \beta \nabla_{\theta_i} l(\Psi), \tag{7.10}$$

where α is the meta-learning rate, and β is the learning rate for each individual task, i.e., the training of a mixture density model. Algorithm 4 shows the detailed training and adaption processes of *MDNML*.

7.2.3 Speaker Identification in a Household

We now discuss how to utilize the constructed users' acoustic profiles to conduct speaker identification given a short voice utterance of a user in a household. Following GMM-UBM [40], in addition to training an acoustic profile $\mathcal{M}_i$ for each user i in the household,

Algorithm 4: Acoustic profile training and adaption

Input: learning rate α, meta-learning rate β, maximal number of iterations itr_{max}, inner update size T_{train} in training, inner update size T_{adapt} in adaption

/* Training on the existing users */
for $itr \leq itr_{max}$ **do**
  Sample a batch of existing users as U
  for user i in U **do**
    Sample a piece of audio of user i
    $\theta_i^{(0)} = \Psi$
    for $t \leq T_{train}$ **do**
      $\theta_i^{(t)} = \theta_i^{(t-1)} - \alpha \nabla_{\theta_i^{(t-1)}} L(\theta_i^{(t-1)})$
  $\Psi = \Psi - \beta \nabla_{\Psi} \sum_{i \in U} L(\theta_i^{(T)})$

/* Adaption on new users by fine-tuning*/
for new user j in U_{adapt} **do**
  $\theta_j^{(0)} = \Psi$
  for $t \leq T_{adpat}$ **do**
    $\theta_j^{(t)} = \theta_j^{(t-1)} - \alpha \nabla_{\theta_j^{(t-1)}} L(\theta_j^{(t-1)})$

we also train a household-level background acoustic profile $\mathcal{M}_{hbm}$ using the mixtures of all training utterances of the users in the household.

Given a user's short voice utterance x_j, we feed it into the universal background profile and each individual acoustic profile, with each profile yielding a vector of fitness scores. Each vector of scores indicates how well the voice utterance fit the corresponding acoustic profile. More specifically, we use $p(x_j \mid \mathcal{M}_{hbm})$ and $p(x_j \mid \mathcal{M}_i)$ to denote the scores for the household-level profile and the profile of user i in the household, respectively. Formally, the speaker identify is given by:

$$\arg\max_i f(\mathbf{1}_{>0}(p(x_j \mid \mathcal{M}_i) - p(x_j \mid \mathcal{M}_{hbm}))), \tag{7.11}$$

where $\mathbf{1}_{>0}(\cdot)$ is the vector-level indicator function and $f(\cdot)$ is a counter, which calculates the number of 1's in its input. By introducing the household-level background profile, it allows us to achieve speaker identification based on background-proof voice frames, which potentially offers stronger discriminative power.

CHAPTER 8

Datasets

We provide a detailed descriptions of the datasets used in different experiments.

8.1 Location-based Recommendation Datasets

8.1.1 Yelp Challenge Dataset

The Yelp dataset, which is publicly available[1], contains interactions between customers and businesses, with 4.1M reviews and 947K tips by 1M users for 144K businesses. For the Yelp dataset, we investigate the recommendation tasks in seven large cities.

8.1.2 Foursquare Dataset

The Foursquare dataset[2] contains interactions between customers and businesses in Los Angeles and New York.

Table 8.1 shows the statistics for the nine cities in the two datasets. For each business, its check-ins are sorted in a chronological order based on the timestamps. The first 50% of the check-ins are used as the training data. The following 20% are used for validation and the remaining 30% are used as the test data for evaluation.

[1] https://www.yelp.com/dataset_challenge

[2] https://www.dropbox.com/s/4nwb7zpsj25ibyh/check-indata.zip

Table 8.1: The statistics of business and customer in Yelp and Foursquare datasets.

Dataset	Yelp							Foursquare	
City	Charlotte	Cleveland	Las Vegas	Madison	Phoenix	Pittsburgh	Toronto	Los Angeles	New York
# of Customers	69,005	5,578	432,399	26,083	314,610	51,422	58,377	501,940	717,382
# of Businesses	10,652	9,960	282,204	3,895	43,482	8,037	20,849	215,614	206,416

8.2 Query Recommendation Dataset

8.2.1 Yahoo! Search Log Dataset

We use a search log collected from Yahoo!, which is one of the largest search engines in the world. We pre-processed the data by eliminating non-alphanumeric characters, spelling error correction, and lower-casing. Then we segmented the search log into sessions, using a standard segmentation heuristic, where intervals of at least 30 minutes idle time denote a session boundary [117]. The pre-processing process in this paper is consistent with previous studies [36, 37, 118]. To partition search sessions into training and test sets, the first 90% data are utilized for training while the remaining sessions are the test data. Among the training data, 10% of sessions are randomly sampled as the validation set for parameter tuning. Finally, there are 3,684,008 training queries within 493,864 sessions and 406,063 test queries within 55,141 sessions. Furthermore, to evaluate the performance using different context lengths, the test set is partitioned into three subsets, including *Short Context* (1 query), *Medium Context* (2 to 3 queries), and *Long Context* (4 or more queries). Here the context length refers to the number of search queries in a search session. Table 8.2 shows the statistics of queries with different context lengths in the training and test datasets.

Table 8.2: The statistics of queries with different context lengths in the Yahoo! dataset.

Dataset	Context Length		
	Short (1 query)	Medium (2-3 queries)	Long (4+ queries)
Training	284,273	105,647	103,944
Test	31,577	11,998	11,566

Table 8.3: The statistics of the LibriSpeech dataset.

Datasets	#(Female Speakers)	#(Male Speakers)	#(Total Speakers)	Total Hours	Per-speaker Minutes
LibriSpeech	125	126	251	∼100 hours	∼25 minutes

8.3 Automatic Speaker Recognition Dataset

8.3.1 LibriSpeech Dataset

The ASR experiments are conducted on the LibriSpeech dataset, which is publicly available[3]. The audio data is derived from reading audio books from the LibriVox project. Table 8.3 shows the statistics of the dataset.

We follow [45] to extract acoustic features from the raw audios. We convert all audio to streams at a 22 kHz sampling rate for consistency. The spectrograms are then generated by a sliding window protocol with a hamming window. The width of the hamming window is 25 ms with step size 10 ms. To remove the duplicated spectrograms coefficients, we further conduct discrete cosine transform. As a convention, 20 coefficients are kept at each time step as the acoustic features for the following speaker identification. The mel frequency cepstral coefficients (MFCCs) are constructed from the raw audio input without any pre-processing such as silence removal etc.

[3]LibriSpeech: `http://www.openslr.org/12`

CHAPTER 9

Experiments and Results

9.1 Customer Recommendation in LBSNs

In this section, we conduct extensive experiments on two real-world datasets, the Yelp challenge dataset and Fothe ursquare dataset, to evaluate the performance of CORALS.

9.1.1 Baselines

To compare our approach with others, the following 12 methods are adopted as baselines.

- **Weighted Regularized MF (WRMF)**. WRMF [119] minimizes the square error loss by assigning both observed and unobserved check-ins with different weights based on matrix factorization.
- **Maximum Margin MF (MMMF)**. MMMF [120] minimizes the hinge loss based on matrix factorization.
- **Bayesian Personalized Ranking MF (BPRMF)**. BPRMF [121] optimizes Area Under the Curve (AUC) based on pairs of observed check-ins and sampled unobserved check-ins.
- **CofiRank**. CofiRank [122] optimizes the estimation of a ranking loss based on Normalized Discounted Cumulative Gain (NDCG).
- **CLiMF**. CLiMF [123] optimizes a different ranking-oriented loss, i.e., Mean Reciprocal Rank (MRR) loss.

- **WARP**. In [75], Weighted Approximate-Rank Pairwise loss is proposed to optimize precision@k. WARP loss differs from AUC loss in updating parameters. WARP keeps drawing negative samples until getting a disordered prediction or reaching a cutoff value.

- k**OS**. $k-$Order Statistic loss is proposed in [124] and provides a variant that optimizes precision@k.

- **USG**. USG [13] is a collaborative filtering method. It utilizes social and geographical information to improve recommendations.

- **GeoMF**. GeoMF [125] is a geographically weighted matrix factorization model.

- **Rank-GeoFM**. Rank-GeoFM [126] incorporates geographical and temporal information to provide recommendations.

- **ASMF**. ASMF [16] utilizes geographical information, social information, and attributes of businesses to enhance the accuracy of recommendations.

- **ARMF**. ARMF [16] extends ASMF by applying ranking losses.

Among these 12 baseline methods, WRMF is a point-wise matrix factorization method while MMMF and BPRMF are pair-wise based. CofiRank, CLiMF, WARP, kOS focus on optimizing top ranked positions. USG, GeoMF, Rank-GeoFM, ASMF, and ARMF utilize additional information, such as check-in locations, social relationship, businesses' attributes, and temporal information to improve the accuracy of recommendations. All parameters in baselines are tuned based on their guidelines.

In addition to the above baselines, we also implement CORALS [1] with two other gradient-based parameter optimization strategies, i.e. SGD and RMSprop [127].

- **CORALS-SGD**. CORALS-SGD applies SGD to conduct optimizations. All parameters share the same learning rate.

[1]To distinguish the parameter learning algorithms used in CORALS and its variants, we also call CORALS CORALS-AdaGrad.

- **CORALS-RMSprop**. RMSprop [127] is applied to optimize learning rates adaptively. It addresses the issue of radically diminishing learning rates in AdaGrad.

9.1.2 Recommendation Performance

In this section, we evaluate the performances of CORALS and its variants against the 12 baseline methods. Mean Average Precision (MAP) is adopted as the evaluation metric. Given a ranked list rl of potential new customers, the average precision for a business b is:

$$ap_b = \frac{1}{\omega} \sum_{pos=1}^{|rl|} precision(pos) * rel(pos) \tag{9.1}$$

where ω is the number of new customers who visit a business b in the test set, pos denotes the position in the ranked list rl and $|rl|$ gives the total number of potential new customers in rl. Customers are ranked decreasingly based on how likely they will come in rl. $precision(pos)$ is the precision of a cut-off rank list from 1 to pos, and $rel(pos)$ is an indicator function that equals to 1 if the customer visits b in the test set, 0 otherwise. For example, three new customers visit a business b (i.e., $\omega = 3$) in the test set and they are ranked at position 2, 4, and 7 in rl, respectively. Therefore, $ap_b = \frac{1}{3}(\frac{1}{2} + \frac{2}{4} + \frac{3}{7})$. The mean average precision is the average of the average precision of all businesses.

$$MAP = \sum_{b=1}^{|B|} ap_b / |B| \tag{9.2}$$

MAP ranges from 0 to 1, and a higher value indicates a better performance in recommendation.

Table 9.1 shows the recommendation performances of different methods on the nine cities from the two datasets. The top seven rows show the performances based on the cities in the Yelp dataset, while the bottom two rows show the performances based on the cities in the Foursquare dataset. In addition, we further show the average recommendation performances for the top (10%) and tail (10%) businesses[2] in each city to demonstrate how each method

[2]Businesses are sorted based on their check-in numbers. Top businesses are the ones that have more check-ins, while tail businesses are the ones that have fewer check-ins.

Table 9.1: Recommendation performance (MAP). The upper table shows the performances of methods using only check-in information, and the lower table demonstrates the performances of methods using both check-in and heterogeneous information. `Mean` represents the average performance on all businesses in a city. `Top` represents the average performance on the top 10% businesses that have more check-ins, and `Tail` represents the average performance on the tail 10% businesses with fewer check-ins.

Method	WRMF			MMMF			BPRMF			CofiRank			CLiMF			WARP			kOS		
City	Mean	Top	Tail	Mean	Top	Tail	Mean	Top	Tail	Mean	Top	Tail	Mean	Top	Tail	Mean	Top	Tail	Mean	Top	Tail
Charlotte	0.026	0.058	0.014	0.028	0.049	0.028	0.029	0.049	0.029	0.031	0.052	0.024	0.034	0.058	0.024	0.044	0.060	0.036	0.038	0.057	0.024
Cleveland	0.041	0.086	0.029	0.039	0.072	0.025	0.040	0.073	0.030	0.043	0.078	0.042	0.050	0.081	0.043	0.055	0.077	0.046	0.053	0.085	0.041
Las Vegas	0.004	0.012	0.001	0.009	0.016	0.004	0.009	0.016	0.005	0.013	0.024	0.009	0.013	0.022	0.008	0.017	0.025	**0.015**	0.014	0.023	0.010
Madison	0.067	0.136	0.043	0.066	0.134	0.042	0.058	0.122	0.034	0.063	0.129	0.037	0.054	0.107	0.031	0.061	0.115	0.039	0.058	0.115	0.038
Phoenix	0.004	0.010	0.002	0.008	0.015	0.006	0.008	0.015	0.006	0.011	0.020	0.008	0.011	0.020	0.009	0.020	0.026	0.015	0.015	0.022	0.013
Pittsburgh	0.028	0.067	0.015	0.027	0.053	0.014	0.027	0.054	0.013	0.031	0.065	0.017	0.037	0.073	0.020	0.044	0.071	0.033	0.040	0.069	0.027
Toronto	0.009	0.019	0.005	0.011	0.018	0.009	0.011	0.017	0.009	0.014	0.025	0.011	0.014	0.022	0.011	0.021	0.031	0.019	0.019	0.029	0.015
Los Angeles	0.005	0.007	0.003	0.005	0.006	0.004	0.009	0.009	0.011	0.010	0.008	0.006	0.008	0.013	0.004	0.011	0.019	0.006	0.009	0.012	0.004
New York	0.002	0.003	0.001	0.003	0.004	0.001	0.005	0.005	0.007	0.004	0.004	0.006	0.005	0.005	0.006	0.005	0.007	0.005	0.004	0.005	0.003

Method	USG			GeoMF			Rank-GeoFM			ASMF			ARMF			CORALS-AdaGrad			CORALS-RMSprop			CORALS-SGD		
City	Mean	Top	Tail	Mean	Top	Tail	Mean	Top	Tail	Mean	Top	Tail	Mean	Top	Tail	Mean	Top	Tail	Mean	Top	Tail	Mean	Top	Tail
Charlotte	0.029	0.039	0.021	0.035	0.058	0.024	0.035	0.041	0.027	0.027	0.049	0.021	0.037	0.084	0.021	**0.056**	**0.087**	**0.050**	0.055	**0.087**	0.046	**0.056**	**0.087**	0.048
Cleveland	0.048	0.075	0.031	0.044	0.089	0.029	0.043	0.068	0.038	0.047	0.081	0.034	0.056	0.110	0.051	**0.091**	0.169	**0.059**	0.090	**0.171**	0.053	0.085	0.164	0.044
Las Vegas	0.008	0.019	0.004	0.017	0.024	0.012	0.011	0.018	0.010	**0.018**	**0.029**	0.011	0.010	0.016	0.005	0.014	0.026	0.010	0.014	0.026	0.010	0.014	0.026	0.010
Madison	0.063	0.104	0.047	0.077	0.148	0.038	0.063	0.112	0.044	0.072	0.151	0.048	0.089	0.184	0.043	0.116	0.192	0.091	**0.121**	0.210	0.095	0.118	**0.212**	**0.105**
Phoenix	0.010	0.017	0.006	0.020	0.023	0.017	0.014	0.016	0.011	0.017	0.023	0.012	0.016	0.019	0.011	**0.021**	0.029	**0.018**	0.020	**0.030**	0.016	0.020	0.029	**0.018**
Pittsburgh	0.030	0.055	0.023	0.038	0.069	0.032	0.042	0.047	0.030	0.047	0.071	0.030	0.041	0.090	0.033	**0.057**	0.115	**0.035**	**0.057**	**0.116**	**0.034**	0.055	0.115	0.033
Toronto	0.014	0.026	0.010	0.022	0.030	0.021	0.016	0.020	0.013	0.018	0.025	0.014	0.012	0.037	0.004	**0.027**	0.038	**0.025**	0.026	**0.040**	0.022	0.026	0.038	0.024
Los Angeles	0.020	0.025	0.017	0.021	0.021	0.017	0.008	0.011	0.006	0.009	0.010	0.007	0.008	0.011	0.004	0.021	**0.028**	**0.023**	**0.022**	0.025	**0.023**	0.019	0.024	0.019
New York	0.005	0.003	0.006	0.010	0.008	0.008	0.003	0.004	0.002	0.006	**0.009**	0.005	0.003	0.003	0.003	**0.012**	0.008	**0.012**	0.011	0.008	0.010	**0.012**	**0.009**	0.011

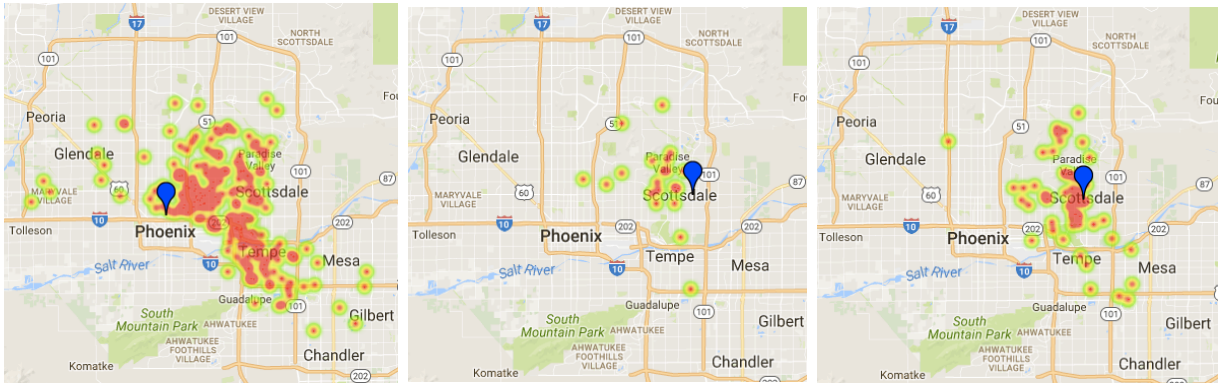
performs when there is a relatively rich or poor amount of check-ins, respectively. For example, WRMF achieves 0.026 on average for all businesses in Charlotte. It achieves 0.058 and 0.014 on average for the top and tail businesses in Charlotte, respectively. We observe that the more check-ins we have for businesses, the more accurate recommendations we can achieve. This observation applies to businesses in almost all nine cities under the 15 methods. This is because the more check-ins we have for businesses, the more accurately we can infer the style, the geographical influence, and the reputation of the businesses.

MMMF, BPRMF, CofiRank, CLiMF, WARP, and kOS achieve better recommendation performances than WRMF in general. This verifies that methods achieving low prediction errors do not necessarily have high recommendation accuracies. In other words, di-

rectly optimizing the predicted check-ins may not always provide the best recommendation lists to businesses. CofiRank, CLiMF, WARP, and kOS further outperform MMMF and BPRMF due to their optimizing strategies. They optimize NDCG, MRR, precision@k, and precision@k, respectively, which all focus on better optimizing the top-ranked customers on the list. BPRMF, which optimizes AUC, focuses on optimizing the entire list of customers. CofiRank, CLiMF, WARP, and kOS outperform USG, which shows the advantage of the learning-to-rank recommendation methods. Even without utilizing location and social information, they can accurately infer customer preferences and achieve good recommendation performances. In general, WARP achieves the best recommendation performance among the 7 methods in the upper table, where only check-in information is utilized to infer customer preference.

GeoMF, Rank-GeoFM, ASMF, and ARMF outperform WRMF, MMF, BPRMF, CofiRank, CLiMF, and kOS in general. It shows that incorporating ancillary information compensate for the sparsity issue in location-based recommendation tasks. The performance of Rank-GeoFM is not as good as the one of GeoMF. This is because Rank-GeoFM, which incorporates temporal information, intends to predict the next point of interest (POI) to visit, while the task in this work is to predict new customers for POIs. GeoMF achieves better *MAP* than ASMF and ARMF. This might be because ASMF and ARMF focus on utilizing social information, while learning geographical influence might be a better way to improve recommendation performances in location-based tasks.

CORALS-AdaGrad or its variants outperform all 12 baseline methods with few exceptions, which demonstrates the effectiveness of CORALS-AdaGrad. In particular, CORALS-AdaGrad increases the mean *MAP* by 51% and 33% against WARP and GeoMF, respectively. Bold numbers in Table 9.1 indicate the winners for the same city and the same group of the business. In summary, CORALS-AdaGrad or its variants win in all scenarios except in Las Vegas where WARP and ASMF score slightly better.



(a) Phoenix Art Museum (b) One branch of McDonald's (c) Alo Cafe

Figure 9.1: Customer heat maps for three local businesses in Phoenix

9.1.3 Geographical Preference Inference

In this section, we will use examples to verify that the geographical influence is both business-dependent and customer-dependent.

We first use three case examples to show the geographical influence on different types of local businesses. We select three local businesses in Phoenix, i.e. the Phoenix Art Museum, a branch of McDonald's, and Alo Cafe. Figures 9.1a, 9.1b, 9.1c show the locations of the three businesses, represented by a blue mark each, together with the heat maps of their visitors. The location of a visitor is estimated by the average of all locations he/she has visited. There are two interesting observations. First, Phoenix Art Museum has more check-ins than McDonald's and Alo Cafe do. Second, the majority of the check-ins of McDonald's and Alo Cafe come from their nearby regions while the visitors of Phoenix Art Museum are scattered all over Phoenix. In addition, the number of museums in Phoenix is much fewer than the numbers of fast-food businesses and cafes. The rationale behind the observations is that people tend to get services from nearby businesses if the services are available since it takes less effort. However, for some businesses that are only available in a remote location, the customers may be more tolerant of traveling a long distance. Therefore, businesses such as fast-food and cafes get influenced more by the geographical convenience than businesses like museums. In CORALS, parameter w_b^g is used to model the geographical influence on a

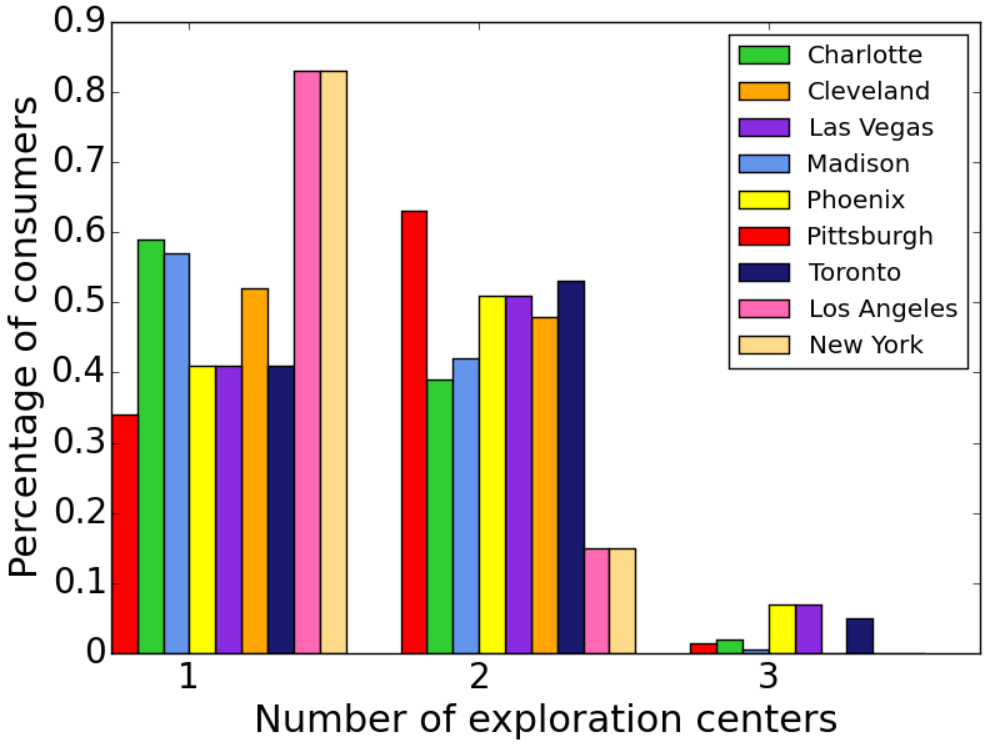
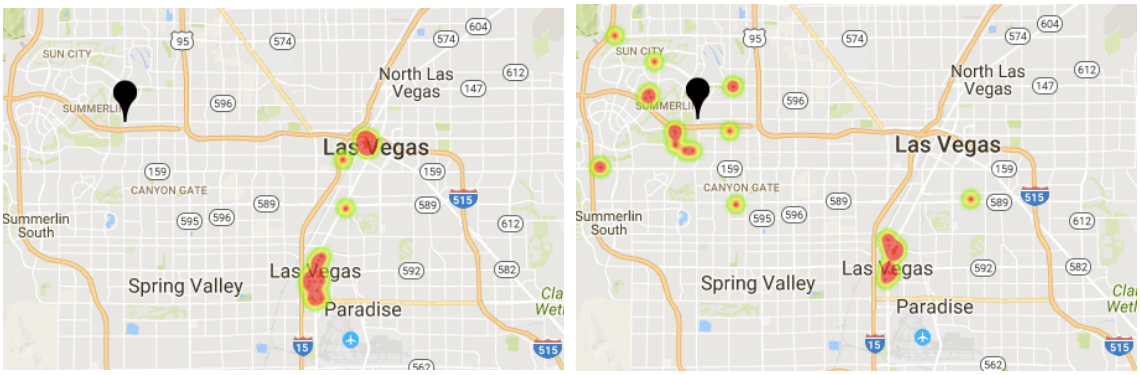


Figure 9.2: Exploration Center Distribution

business b. Higher values of w_b^g indicate greater influences on the geographical convenience. In Section 9.1.5, we show a detailed analysis of w_b^g on various types of businesses.

Then, we study the geographical influence on individual customers. We randomly sample two customers from Las Vegas and plot their check-ins in Figures 9.3a and 9.3b, respectively. We observe that the two customers have their own exploration preferences. User 1 tends to explore the main street in Las Vegas, while user 2 not only explores the main street but also checks in at the northwestern region of Las Vegas. Given a local business b, represented by the black marker, GMM tells $g_{b,u_1} < g_{b,u_2}$, which indicates that business b is more geographically convenient for user 2. The geographical convenience information, embedded in the GMM, helps CORALS better understand customers' decision-making processes from the perspective of the convenience of the local businesses.

Note that for each customer, we group his/her check-ins by affinity propagation to derive the number of components in the GMM. Figure 9.2 shows the customer percentage distri-



(a) User 1's explorations (b) User 2's explorations

Figure 9.3: Explorations of two customers in Las Vegas

butions over the number of exploration centers in different cities. We observe that most customers have only one or two exploration centers. The rationale behind it is that most customers explore around their workplaces or/and residences, which is consistent with the findings in the previous study [128].

9.1.4 Reputation Influence Analysis

In this section, we investigate how the *MAP* performance of CORALS changes with the number of reviews considered when constructing businesses' reputation vectors. First, we do not incorporate any reviews, notated as 0 reviews. Then, we use 1, 3, 5, 7, 9, and 11 most recent reviews to construct the reputation vectors of businesses, respectively. Figure 9.4 shows the performance of CORALS (measured by *MAP*) on the nine cities. In particular, the performance on the Yelp dataset is plotted in solid lines, while the performance on the Foursquare dataset is plotted in dashed lines. When we ignore review information in the model, the performance is relatively poor. As long as we incorporate the information of the most recent review, the performance improves. For example, the performance increases from 0.081 to 0.097 for Madison. However, when we incorporate more reviews to construct reputation vectors, the performance gain is marginal. This is mainly due to the fact that customers only read a few latest reviews to perceive the reputation of the local business.

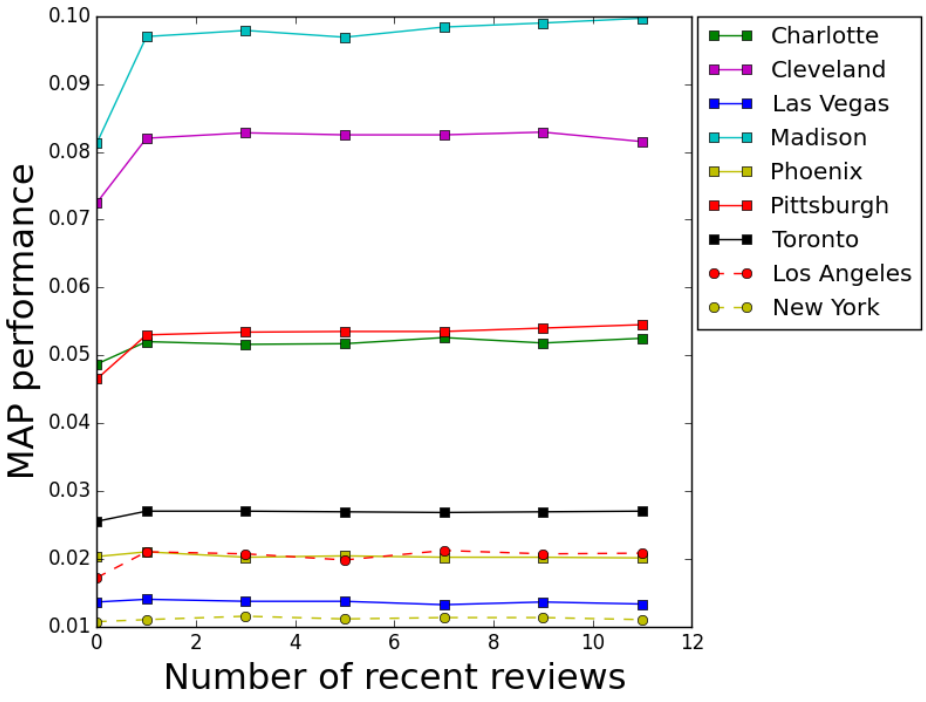


Figure 9.4: MAP performance over number of recent reviews

9.1.5 Analysis on Geographical Convenience and Reputation Reliance

In this section, we analyze to what extent the geographical convenience and online reviews affect customers' decisions in visiting various types of local businesses.

We look into five types of local businesses, i.e. fast-food, bar, cafe, salon, and museum in the two largest cities, i.e., Phoenix and Las Vegas, in terms of the number of customers and businesses. The type of the business is inferred from the name of the business. For each type of businesses, we look into their geographical influence weights w^g and reputation influence weights w^r, and calculate the type-wise median of the influence weights. Table 9.2 shows the analysis based on the businesses in Phoenix. There are 144 fast-food restaurants, 302 bars, 394 cafes, 144 salons, and 8 museums. The geographical influences of fast-food restaurants, bars, and cafes are all around 0.316. For salons, the low geographical influence weight, 0.306, indicates that customers are willing to travel a little bit farther for better haircare services. For museums, which are fewer in quantity, customers have to travel farther compared with other types of businesses. The geographical influence decreases to 0.248. Moreover, the

Table 9.2: Influential factors study for businesses in Phoenix

Phoenix	Fast-food	Bar	Cafe	Salon	Museum
Number of businesses	144	302	394	28	8
Geographical influence	0.316	0.313	0.321	0.306	0.248
Review influence	0.101	0.127	0.133	0.241	0.234

Table 9.3: Influential factors study for businesses in Las Vegas

Las Vegas	Fast-food	Bar	Cafe	Salon	Museum
Number of businesses	112	280	350	27	9
Geographical influence	0.302	0.29	0.298	0.297	0.232
Review influence	0.070	0.073	0.088	0.203	0.149

reputation also has distinct influences on different types of businesses. The reviews on fast-food restaurants, bars, and cafes have a relatively small influence on customers' decisions since customers care more about the convenience of these types of local businesses. For museums and salons, where customers care more about the experiences, reviews have a stronger influence. Table 9.3 shows the same analysis based on the businesses in Las Vegas, which is consistent with most discoveries in Phoenix. There is one interesting discovery about the geographical influence on bars in Las Vegas, which indicates that customers in Las Vegas are willing to take more effort in visiting faraway bars compared with fast-food restaurants, cafes, and even salons. A possible rationale behind it is that there are many attractive shows and events in Las Vegas bars.

9.2 Customer Recommendation in LBSNs with Few-shot Learning

In this section, we conduct extensive experiments on two real-world datasets to evaluate the performance of SEATLE.

9.2.1 Baselines

To compare our approach with others, the following 13 methods are adopted as baselines.

Recommendation methods without considering geographical influence:

- **WRMF**, weighted regularized matrix factorization [119] minimizes the square error loss by assigning both observed and fake check-ins with different weights.
- **MMMF**, maximum margin matrix factorization [120] minimizes the hinge loss based on matrix factorization.
- **BPRMF**, bayesian personalized ranking matrix factorization [121] optimizes pairwise bpr losses of observed check-ins and sampled fake check-ins.
- **CofiRank**, [122] optimizes the estimation of a ranking loss based on normalized discounted cumulative gain.
- **CLiMF**, [123] optimizes a different ranking-oriented mean reciprocal rank loss.

Conventional methods with geographical influence involved:

- **USG**, [13] is a collaborative filtering method. It utilizes distances between users and businesses as extra guidance to make recommendations.
- **GeoMF**, [125] explicitly learns user activity areas (distance-based) and business influences areas via matrix factorization.
- **Rank-GeoFM**, ranking-based geographical factorization [126] incorporates business neighborhood information via matrix factorization.
- **ASMF**, [16] mainly leverages social network information to improve recommendations.
- **ARMF**, [16] extends ASMF by further optimizing ranking losses.
- **CORALS**, [129] models geographical convenience and business reputation to improve recommendations.

Deep learning-based methods with geographical influence involved:

- **SAE-NAD**, self-attentive encoder and neighbor-aware decoder [28] applies auto-encoders to make recommendations.
- **PACE**, preference and context embedding [27], a deep neural architecture that jointly learns the embeddings of users and businesses by building a context graph.

Among these 13 baseline methods, WRMF is a point-wise matrix factorization method, while MMMF and BPRMF are pair-wise based. CofiRank, CLiMF focus on optimizing top ranked positions. These 5 methods do not utilize any ancillary information. USG, GeoMF, RankGeoFM, ASMF, ARMF, and CORALS utilize additional information, such as geographical distances, social networks, and online reviews to improve recommendation performance in LBSNs. SAE-NAD utilizes auto-encoders, with business neighborhood information considered, to make recommendations. PACE models geographical influence by a user-user and business-business context graph. All parameters in baselines are best tuned based on their guidelines.

To achieve best performances on different cities, the optimal parameters vary on different cities. Table 9.4 shows the main parameters and their default values to tune in the experiments.

Table 9.4: Main Parameters

Parameters	Value	Parameters	Value
Learning rate	0.001	Number of epochs N	30
Number of references k	4	Number of queries c	8
Margin γ	0.1	Tuple feature dimension	10
User feature dimension	10	Business feature dimension	10

9.2.2 Recommendation Performance

In this section, we evaluate the performances of SEATLE against the 13 baseline methods. Mean Average Precision (MAP) is adopted as the evaluation metric, which is also used

in [16, 129]. Given a ranked list rl of potential new customers, the average precision for a business b is:

$$ap_b = \frac{1}{\omega} \sum_{pos=1}^{|rl|} \text{precision}(pos) * \text{rel}(pos) \tag{9.3}$$

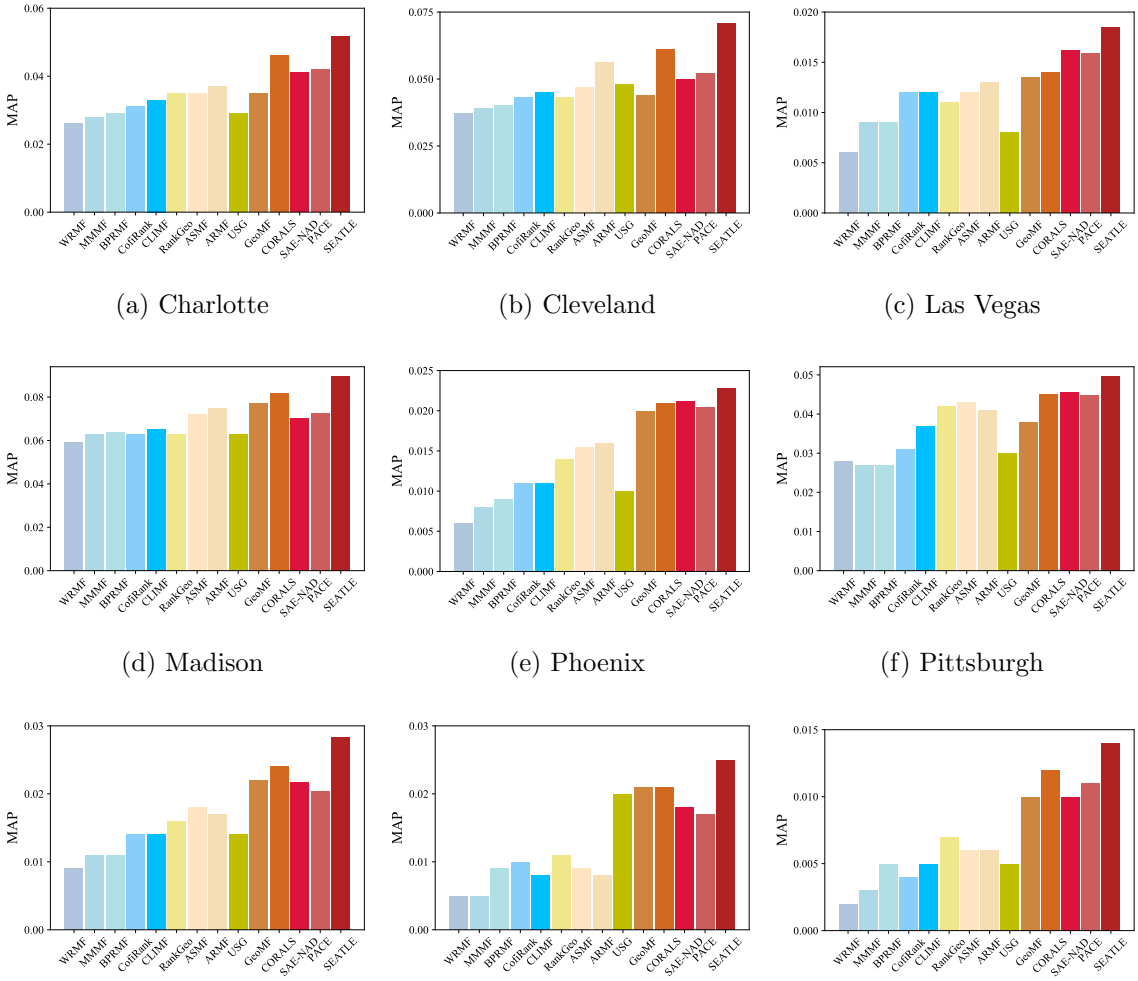
where ω is the number of new customers who visit a business b in the test set, pos denotes the position in the ranked list rl and $|rl|$ gives the total number of potential new customers in rl. Customers are ranked decreasingly based on how likely they will come in rl. precision(pos) is the precision of a cut-off rank list from 1 to pos, and rel(pos) is an indicator function that equals to 1 if the customer visits b in the test set, 0 otherwise. The mean average precision is the average of the average precision of all businesses.

$$\text{MAP} = \sum_{b=1}^{|B|} ap_b / |B| \tag{9.4}$$

Figure 9.5 shows the recommendation performances of different methods on the nine cities from the two datasets. Figures from 9.5a to 9.5g show the performances based on the seven cities in the Yelp dataset, while the last two Figures 9.5h and 9.5i show the performances based on the two cities in the Foursquare dataset.

Among methods which do not consider geographical influence, MMMF, BPRMF, Cofi-Rank, and CLiMF achieve better recommendation performances than WRMF in general. This demonstrates that point-wise methods, such as WRMF, which achieve low prediction errors, do not necessarily have high recommendation accuracy. In other words, directly optimizing the predicted check-ins may not provide the best recommendation lists to businesses.

After leveraging geographical influence, Rank-GeoMF, ASMF, ARMF, GeoMF, CORALS, SAE-NAD, and PACE outperform the five above methods, which do not incorporate any ancillary features. It verifies that modeling ancillary information can offer extra guidance and compensate for the sparsity issue in location-based recommendation tasks. USG, with geographical influence modeled, does not perform as well as expected in some cities, such as Charlotte, Las Vegas, etc. This is due to its oversimplified model design, which is a straightforward linear combination of user preference and geographical distance scores without proper optimizations. The performances of ASMF and ARMF on the foursquare dataset



(a) Charlotte (b) Cleveland (c) Las Vegas

(d) Madison (e) Phoenix (f) Pittsburgh

(g) Toronto (h) Los Angeles (i) New York

Figure 9.5: MAP performances of different methods over nine cities

are not as good as it usually achieves on the Yelp dataset. ASMF and ARMF mainly focus on leveraging social network information. This demonstrates that social network information may not always be reliable, while comprehensively utilizing geographical influence might be a better choice to improve performances in location-based recommendation tasks. In general, CORALS outperforms Rank-GeoMF, GeoMF, SAE-NAD, and PACE. This mainly results from the geographical convenience incorporated in CORALS rather than distance-based metrics employed in other models.

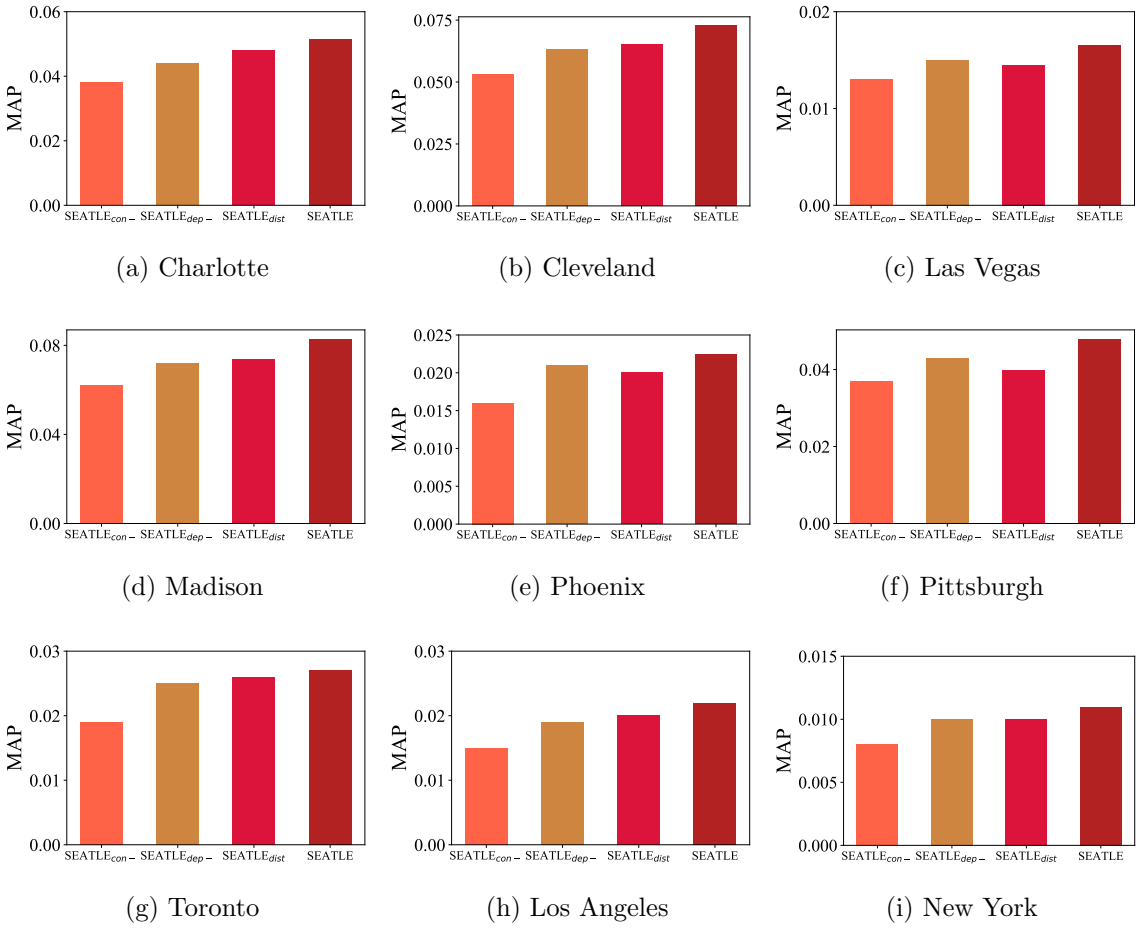
Among the deep learning-based methods, SEATLE achieves the best performance. The

reasons could be explained as follows. PACE models the geographical influence by a context graph, which does not explicitly model the user reachability to businesses. SAE-NAD captures the geographical dependency through a neighbor-aware auto-encoder, but it fails to incorporate geographical convenience. In general, SEATLE outperforms all baseline methods in the nine cities over the two datasets. SEATLE models both geographical convenience and dependency, which jointly and comprehensively express the power of geographical influence. Moreover, SEATLE adopts few-shot learning, designed for learning with limited data, as the framework to cope with data sparsity. These appropriate designs make SEATLE a good fit for new user recommendations in LBSNs.

9.2.3 Geographical Influence Analysis

In this section, we investigate the effectiveness of geographical convenience and dependency modelings. We develop SEATLE$_{con-}$ and SEATLE$_{dep-}$ by removing the convenience and dependency feature from SEATLE, respectively. To compare convenience-based and distance-based influence, we further develop SEATLE$_{dist}$ by replacing the convenience feature by a distance-based kernel metric, adopted from SAE-NAD. More precisely, the metric is given by $\exp(\gamma|\boldsymbol{l}_i - \boldsymbol{l}_j|)$, where $\boldsymbol{l}_i$ and $\boldsymbol{l}_j$ are the location coordinates of two locations, and γ is a hyper-parameter to control the correlation level of the two locations.

Figure 9.6 shows the MAP performances of SEATLE$_{con-}$, SEATLE$_{dep-}$, SEATLE$_{dist}$, and SEATLE. We observe that when the geographical convenience and dependency feature is removed from SEATLE, MAP performance drops correspondingly. The performance decrease more when eliminating geographical convenience as compared to eliminating geographical dependency. This observation applies to all nine cities. Therefore, we can safely conclude that incorporating geographical convenience and dependency helps improve the recommendation performance and the geographical convenience contributes more. We further compare SEATLE$_{dist}$ and SEATLE. SEATLE$_{dist}$ models distance-based geographical features, while SEATLE incorporates convenience-based geographical features. We notice that SEATLE outperforms SEATLE$_{dist}$ on all nine cities. This demonstrates the advan-



(a) Charlotte (b) Cleveland (c) Las Vegas

(d) Madison (e) Phoenix (f) Pittsburgh

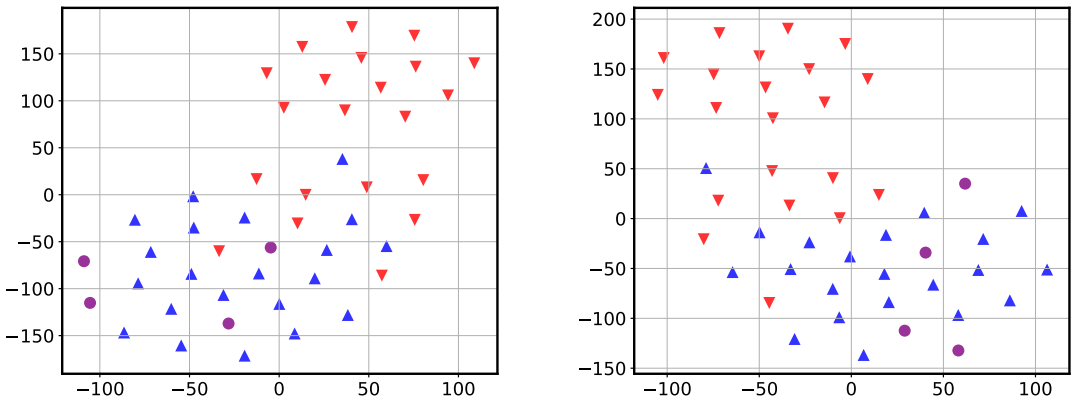
(g) Toronto (h) Los Angeles (i) New York

Figure 9.6: Geographical influence analysis over nine cities

tage of convenience-based geographical modeling since it gauge users' actual transportation efforts more accurately.

9.2.4 Few-shot Analysis

The goal of metric-learning-based few-shot learning is to distinguish positive queries from negative queries regarding some references with limited training instances. In this part, we investigate the effectiveness of few-shot learning in SEATLE by visualizing its learnt embeddings.



(a) Business 1

(b) Business 2

Figure 9.7: Embedding visualizations for references, positive and negative queries regarding two businesses

After training SEATLE, we randomly select two businesses. We project the reference, positive, and negative query embeddings into two dimensional space via t-SNE for visualizations. Figure 9.7 shows the embeddings regarding the two businesses. The references, positive, and negative queries are colored in purple, blue, and red, respectively. Figure 9.7a shows that four references lie on the left bottom area. Most positive queries are located in the same area near the references. A few hard-to-distinguish negative queries are located in the middle, while most negative queries are located in the right upper area, which are

relatively faraway from the references. We have similar embedding proximity observations for the second business in Figure 9.7b. These two figures demonstrate how SEATLE distinguishes observed (i.e. reference and positive) check-ins from fake check-ins by leveraging few-shot learning, which in this case makes a distinction based on only 4 reference check-ins.

9.3 Click Feedback-aware Query Recommendation

9.3.1 Evaluation Metrics

We employ Mean Reciprocal Rank (MRR) score [130] as the evaluation metric. Given a search query Q, let $Y(Q)$ be a ranked list of recommended queries determined by a suggestion method. We use $\text{rank}_1(Y(Q))$ to denote the rank of the first clicked query in $Y(Q)$. Formally,

$$\text{MRR} = \frac{1}{\text{rank}_1(Y(Q))}. \tag{9.5}$$

Essentially, the MRR score summarizes the ranks of the first clicked queries in the recommendation list $Y(Q)$ - A larger score indicates that the first clicked queries are ranked higher in the list.

9.3.2 Baselines

In our evaluations, we evaluate the MRR performance of *CFAN* against the following state-of-the-art methods.

- **[Query-based Variable Markov Model (QVMM)]** [35] makes query suggestions by learning the probability of query transitions over search sessions with the variable memory Markov model implemented by a suffix tree.
- **[Feature Based Suggestion (FBS)]** [131] counts on statistical features to make suggestions. Suggestion candidates are generated based on 1) term matching between search queries and candidates and 2) query co-occurrences in search sessions. Query

suggestions are made by ranking candidates based on a combination of statistical features.

- [**Reformulation-based Completion (RC)**] [36] is a non-deep learning based method exploiting query suggestions. 43 reformulation based features are proposed to capture user reformulation behaviors over search sessions with LambdaMART [132].

- [**Most Popular Suggestion (MPS)**] [5, 6] is a maximum likelihood method, which relies on "wisdom of the crowd". It ranks queries by the co-occurrence to the last query in the search sequence.

- [**Hybrid Suggestion (Hybrid)**] [133] considers both the context information and the popularity by ranking candidate queries based on a linear combination between the popularity and the similarity to recent search queries.

- [**Hierarchical Recurrent Encoder-Decoder (HRED)**] [5] is a deep learning based query suggestion method. HRED constructs a hierarchical encoder-decoder structure to model the sequential and hierarchical dependencies across terms and queries to make query suggestions.

- [**Reformulation inference network (RIN)**] [4] is the most state-of-the-art query suggestion method. Query reformulations between consecutive queries in search sessions are explicitly modeled in RIN to make suggestions.

9.3.3 Parameter Settings

We summarize the parameter settings of *CFAN* in our experiments in Table 9.5. We implemented *CFAN* based on TensorFlow [134] and chose Adam [135] as our optimizer to learn the model parameters. The hyperparameter settings were optimized with the grid search strategy [136]. In addition, we also optimized the parameter settings of all baselines using grid search strategy and report their best performance.

Table 9.5: Parameter Settings

Parameter	Value	Parameter	Value
Learning Rate	0.001	Embedding Size	50
Dropout Keep Ratio	0.8	Number of Epochs	10
Perturbation Bound ϵ	10		

9.3.4 Experimental Results

Table 9.6: The MRR performance of different methods in the test sets with different context lengths for the task of query suggestion.

Dataset	QVMM [35]	FBS [131]	RC [36]	MPS [5,6]	Hybrid [133]	HRED [5]	RIN [4]	CFAN
Overall Context	0.441	0.458	0.466	0.478	0.473	0.556	0.573	**0.650**
Short Context (1 query)	0.446	0.473	0.460	0.476	0.473	0.548	0.563	**0.639**
Medium Context (2 to 3 queries)	0.430	0.438	0.477	0.483	0.473	0.564	0.591	**0.661**
Long Context (4 and more queries)	0.431	0.440	0.469	0.474	0.475	0.574	0.593	**0.682**

Table 9.6 shows the MRR performance of different methods over various context lengths. Note that MRR ranges from zero to one and a higher MRR score indicates better ranking performance on query suggestions.

QVMM, a variable-memory Markov model, achieves the worst MRR performance among all the query suggestion methods. QVMM counts on query dependencies in search sessions to make query suggestions. The sparsity of search sessions leads to the poor ranking performance of QVMM. Feature based method (FBS) first selects candidate queries based on prefix matching and query dependencies. Then, candidate queries are ranked based on statistical popularity features. Compared with QVMM, FBS improves the overall MRR from 0.441 to 0.458. The reformulation-based completion method (RC) outperforms both QVMM and FBS. Its overall MRR score reaches 0.466. RC relies on generalized query dependency rules to make query suggestions. The improvements against QVMM and FBS demonstrate that generalized reformulation rules are more informative and powerful than simple query dependencies in search sessions. The popularity-based baseline method (MPS) slightly outperforms RC. The overall MRR score further increases from 0.466 to 0.478. It indicates that

the query popularity plays an important role in driving users to click the suggestions. This is mainly because when query suggestions are highly related to the search query, users tend to click popular and trending ones as follow-up search queries to explore. The hybrid suggestion method (Hybrid) performs slightly worse than MPS. The overall MRR score decreases from 0.478 to 0.473. This can be explained by the fact that the term-level similarities between suggestion candidates and search queries in Hybrid are not capable of capturing the their relationship in semantics, and thus fail to rank suggestion candidates properly when most suggestion candidates are similar to search queries in term levels.

Neural network based methods HRED, RIN, and *CFAN* outperform non-deep learning based methods by a large margin in general. HRED achieves MRR scores of 0.556, 0.548, 0.564, and 0.574 on overall, short, medium, and long contexts, respectively. RIN achieves MRR score of 0.573 on average. Among all the methods, *CFAN* achieves the best MRR performance. The MRR scores reach 0.650, 0.639, 0.661, and 0.682 on overall, short, medium, and long search contexts, respectively. In particular, *CFAN* achieves excellent ranking performance when the context information is rich. Rich context contains more user search queries and click feedback, which help reduce the query ambiguity and thus contribute to the ranking performance.

To discuss the performance of *CFAN* with different context lengths, we investigate the improvements of all neutral network based methods over QVMM. Figure 9.8 shows the improvements of HRED, RIN, and *CFAN* over QVMM with different context lengths. All these three methods apply hierarchical recurrent neural networks to encode search query sequences. The overall MRR improvement of HRED is about 26%. RIN adds an extra query reformulation layer to the encoder and further improves the MRR by 30% on average. Among all three neutral network based methods, *CFAN* achieves the most significant MRR improvements against QVMM. In general, *CFAN* improves MRR by 50% on overall scenarios. In particular, when the context information is rich, i.e. medium and long contexts, the improvements reach over 55%.

In the following paragraphs, we analyze the effectiveness of *CFAN* with and without

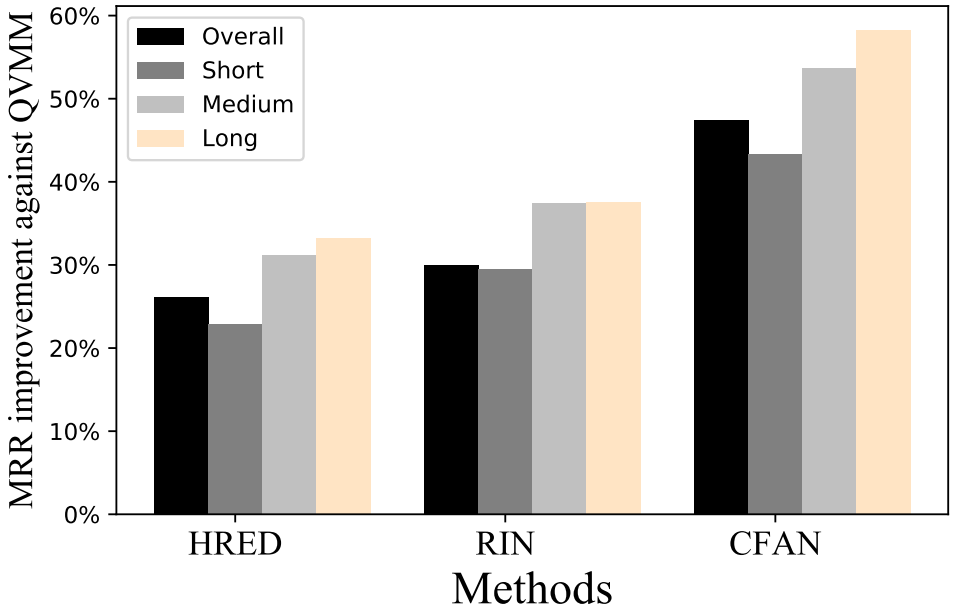


Figure 9.8: The MRR improvement of three methods over the QVMM baseline method with different context lengths.

incorporating user clicks and applying adversarial training sequentially.

Effectiveness of incorporating click feedback. We first investigated the effectiveness of incorporating user click feedback in *CFAN*. As a leave-one-out analysis, we train *CFAN* without user click information, denoted as $CFAN_{NF}$. To remove the click information from *CFAN*, training instances are constructed by concatenating only the embedding $\boldsymbol{s}$ of the search sequence S, the embedding $\boldsymbol{q}$ of the candidate query Q_{can}, and the similarity $\text{Sim}(S, Q)$ between them, where $\text{Sim}(S, Q) = \boldsymbol{s}^T \boldsymbol{W}_{\text{sim}} \boldsymbol{q}$. Figure 9.9 shows the MRR performance of $CFAN_{NF}$ and *CFAN*. When the context is short, there is only one query in the search sequence and there are no clicked suggestions as feedback. The MRR scores of $CFAN_{NF}$ and *CFAN* are roughly the same. When rich contexts are given, i.e., medium and long contexts, clicked suggestions become available. The MRR scores of *CFAN* improves significantly against $CFAN_{NF}$. The improvements on medium and long contexts also enhance the overall ranking performance of *CFAN* by increasing the MRR score from 0.63 to 0.65 compared with $CFAN_{NF}$. These improvements validate the necessity to incorporate user clicks when

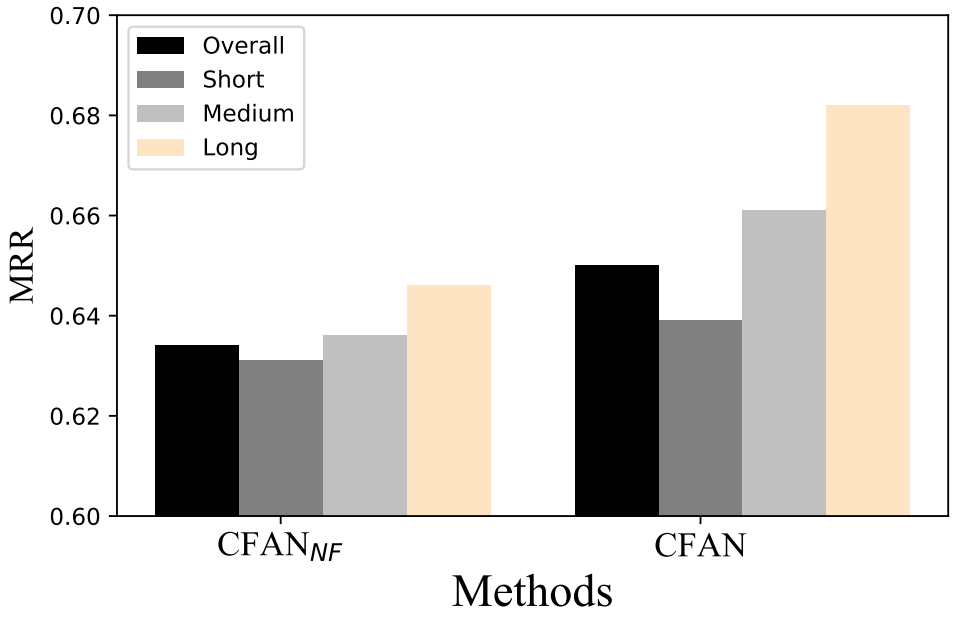


Figure 9.9: The MRR performance of *CFAN* with and without click feedback information.

modeling user search intent.

Adversarial training vs. normal training. In addition to investigating the effectiveness of incorporating user clicks, we also want to show the advantage of *CFAN* with adversarial training. We denote *CFAN* without adversarial training as $CFAN_{NA}$. $CFAN_{NA}$ only optimizes the ranking performance of the original training data with loss function defined by Equation 5.1. Figure 9.10 shows the MRR performance of $CFAN_{NA}$ and *CFAN*. We can see *CFAN* consistently outperforms $CFAN_{NA}$ in all suggestion scenarios, even when the context is short. It clearly owes to the adoption of adversarial training. Adversarial training generates unseen tough classification and ranking training instances and successfully shrink the performance gap between training and test by optimizing the constructed adversarial examples.

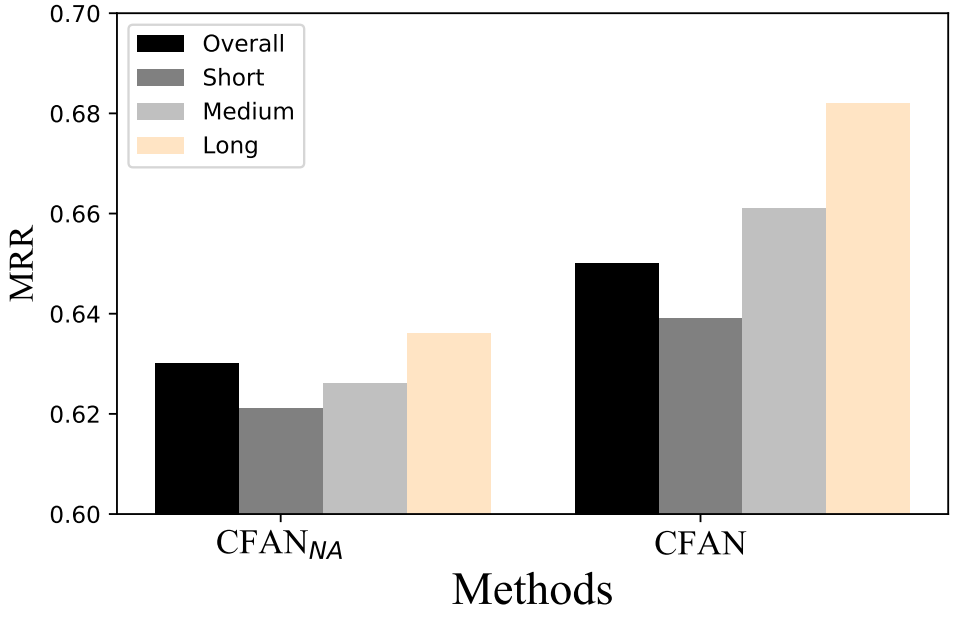
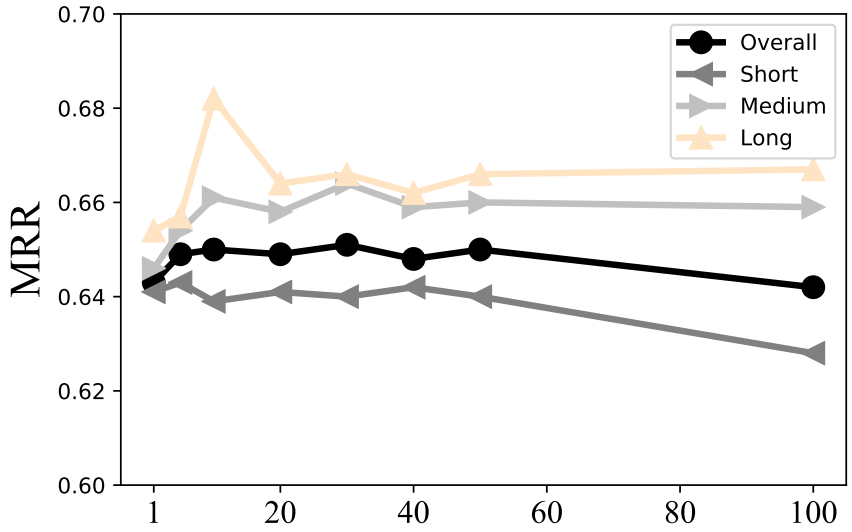


Figure 9.10: The MRR performance of *CFAN* with and without adversarial examples.

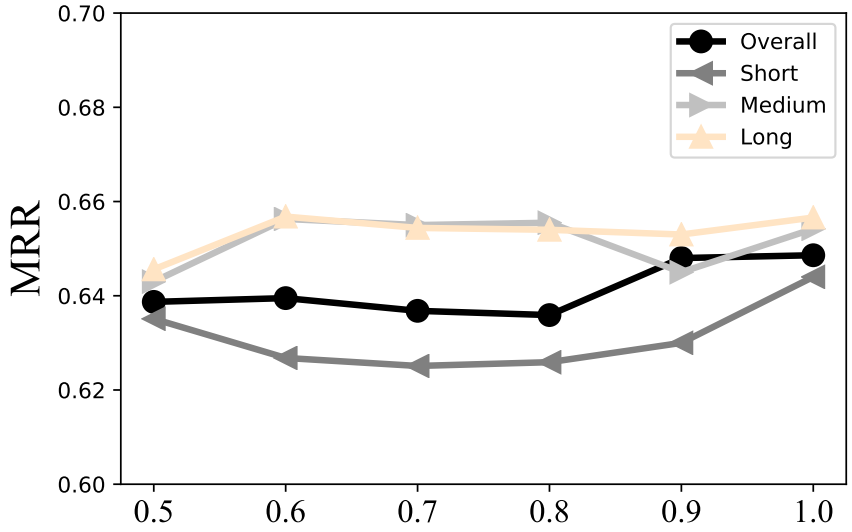
9.3.5 Parameter Sensitivity Studies

CFAN, as a neural network based method, involves the tuning of several general training parameters (i.e., learning rate, dropout ratio, embedding size, and number of training epochs etc.). The perturbation bound ϵ, special in the proposed model, comes as an extra tuning parameter. To investigate the sensitivity of *CFAN*, we examine how different choices of the parameters affect the performance of *CFAN*. Except for the parameter being tested, we set other parameters to the default values (see Table 9.5).

Due to space limit, we selectively show the results of parameters that are more relevant to our proposed model in Figure 9.11. Figures 9.11a and 9.11b show the MRR performance of *CFAN* as we change the perturbation bound ϵ and dropout keep ratio, respectively. By tracing out different values of ϵ, we can see that the MRR performances are consistently stable over a large range of ϵ. This eases our selection of the perturbation bound. Theoretically, the ranking performance decreases significantly as we increase ϵ large enough. As shown in Equation 5.18, choosing extreme large ϵ pollutes training instances and move into the regime



(a) Perturbation Bound



(b) Dropout Keep Ratio

Figure 9.11: The MRR performance of *CFAN* over different settings of parameters.

of rubbish inputs. We test it by setting ϵ to 1×10^4, the losses on both training and validation data fluctuate with no signs to increase. Dropout [137] is a common regularization method to avoid overfitting. *CFAN* is insensitive to the changes of it as the overall MMR performance stays around 0.64 with different settings of dropout keep ratio. It can be explained by the adoption of adversarial training. Adversarial training also functions as a regularizer. It manages the inevitable gap between the training error and the test error by optimizing the constructed adversarial examples. This makes *CFAN* insensitive to the change of dropout keep ratio. In addition, we also investigate the influences of different settings of embedding size and training epoch number. The MRR scores increase and peak at certain values of these two parameters as we increase them from small numbers. The MRR scores decrease as we further increase them because of the overfitting issue. Due to space limit, we do not show the corresponding figures.

9.4 Automatic Speaker Identification with Metric learning-based Few-shot Learning

In this section, we conduct extensive experiments on a real-world dataset to evaluate the performance of *AFEASI*.

9.4.1 Baselines

To evaluate the performance of *AFEASI*, the following eleven methods are adopted as baselines, including seven conventional neural network-based methods, one few-shot learning-based method, one waveform-based method, and two variants of *AFEASI*.

Conventional neural network-based methods:

- **1D-CNN**. Multiple layers of 1D-CNN are utilized to construct audio embeddings from MFCCs, where the convolution is conducted along the time dimension. Global average pooling [138] is employed for aggregation before feeding into an output layer, where

neutrons are equal to the total speakers for identification.

- **2D-CNN**. Different from 1D-CNN, 2D-CNN [139] is utilized to extract acoustic features from MFCCs.

- **LSTM** applies recurrent neural networks to investigate the acoustic frequency dependencies along all time steps. In the experiments, a bidirectional LSTM [91] is utilized to model such frequency dynamics along the time dimension and build audio embeddings.

- **Attentive-LSTM (A-LSTM)** differs from the LSTM method by introducing an attention layer [140] on top of the bidirectional LSTM to extract important acoustic signals at different time steps.

- **Attentive-CRNN (A-CRNN)** first utilizes a layer of 1D-CNN to extract local features at each time step and further builds an attentive LSTM model on top of such features to construct audio embeddings.

- **Self-attention (SA)** also seeks to extract audio embeddings by studying the frequency dynamics along the time dimension. More precisely, the self-attention technique [86] is utilized, where the same MFCC is considered as the input, query, and value matrices. Finally, the average of the fused vectors via self-attention operations serves as the audio embedding.

- **Attentive self-attention (A-SA)** first utilizes the self-attention technique [86] to fuse the acoustic vectors at different time steps. A weighted sum of the fused vectors over all time steps serves as the audio embedding.

Few-shot learning-based method:

- **Prototypical network (PN)** [45] adopts 2D-CNN as the building block to construct audio embeddings and applies prototypical loss [122] to learn from limited training data.

Raw waveform-based method:

- **Sincnet (SC)** [9] identifies speakers by directly training on the raw waveform of audios.

AFEASI variants:

- ***AFEASI***$_s$ differs from *AFEASI* in the choice of perturbation injections and only injects noises into audio instances in the support module.
- ***AFEASI***$_b$ injects noises into audio instances in both support and query modules.

Among the eleven baseline methods, the 1D-CNN, 2D-CNN, LSTM, A-LSTM, A-CRNN, SA, and A-SA differ in how to construct the audio embedding representation from the MFCCs. The CNN based methods employ convolutional operations to extract the local informative and discriminative features from MFCCs. The LSTM, A-LSTM, SA, and A-SA methods depend on investigating the dependencies of MFCC intensities at different time steps to construct audio embedding representations. The A-RCNN method utilizes both CNN and RNN to extract acoustic features and form audio embeddings. For these seven methods mentioned above, the constructed audio embeddings are further fed into the prediction blocks to yield speaker recognition. We include these seven methods as baselines to investigate which one of them is the most effective in extracting discriminative acoustic features from MFCCs in the context of ASR. PN utilizes 2D-CNN as the building block to construct audio embedding representations. It differs from the first seven baselines in how to conduct predictions. It utilizes metric learning to boost ASR performance. Sincnet differs from all baselines in the sense that it learns from the raw waveform of audios rather than from MFCCs. *AFEASI*$_s$ and *AFEASI*$_s$ are variants of *AFEASI*. They differ in where adversarial noises are injected. All parameters in these baselines are best tuned utilizing grid search.

9.4.2 Identification Performance

In this section, we evaluate the performances of *AFEASI* against different baseline methods. We adopt accuracy as the evaluation metric. Given a set of test audio instances, the accuracy

Table 9.7: Accuracy on test set over different audio embedding construction methods

Method	1s	3s	5s	7s	9s
1D-CNN	0.9021	0.9702	0.9853	0.9927	0.9931
2D-CNN	0.9038	0.9686	0.9780	0.9823	0.9879
LSTM	0.8650	0.9551	0.9607	0.9823	0.9888
A-LSTM	0.8848	0.9698	0.9819	0.9905	0.9922
A-CRNN	0.9198	0.9594	0.9720	0.9767	0.9810
SA	0.7537	0.8736	0.9107	0.9383	0.9405
A-SA	0.7886	0.9159	0.9435	0.9594	0.9642
SC	0.8147	0.8773	0.8806	0.8913	0.8991

acc is:

$$acc = \frac{\text{correctly identified test instances}}{\text{total test instances}}. \tag{9.6}$$

In this section, we investigate which technique is more effective on extracting informative acoustic biometric features from MFCCs. In particular, we compare 3 types of different methods, i.e., CNN, LSTM, and self-attention-based methods. Moreover, we also investigate the effectiveness of attention mechanisms on acoustic feature constructions. To further investigate how effective to directly identify speakers based on raw waveform of audios, we further include SC into the comparisons. To comprehensively compare these techniques on speaker identification, we vary the length of the audio instances from 1 second to 9 seconds with 2 seconds as the step size. Table 9.7 shows the corresponding performances on LibriSpeech. While the top seven rows show the performances of methods based on MFCCs, last row shows the performance of SC, which is waveform-oriented.

For MFCC-oriented methods, we have five observations. First, the longer the instance, the higher accuracy each method can achieve. It applies to all seven methods with different embedding construction strategies. It makes sense because the longer each audio instance, the richer acoustic information we have collected from each instance. Training, supported by rich acoustic information, contributes to high accuracy. The second observation is that

the SA method achieves the worst accuracy performance on all different duration settings. The SA method depends on feature fusions to learn inter-dependent feature representations. However, a piece of audio, especially a short one, could contain a notable portion of silence which do not contain any distinguishing information. Feature fusions with such uninformative and misleading features lead to defective accuracy performance. The third observation is that all methods, except SA, work well when instances are 3 seconds long or longer than that. It demonstrates that audios with at least 3 seconds might be informative enough to construct a speaker's acoustic biometric. Moreover, by comparing LSTM with A-LSTM and A-CRNN, we also notice that it benefits the accuracy performance by adding an attention layer, especially when the audio instance is relatively short. When each instance is only 1 second long, the accuracy of the LSTM method is only 0.8650. By distinguishing important information at different time steps, A-LSTM and A-CRNN improve the accuracy to 0.8848 and 0.9198, respectively. Analogously, we find similar performance improvement when comparing SA with A-SA. The last observation is that the accuracy performance of LSTM, A-LSTM, SA, and A-SA are more sensitive to short audios than CNN-based methods. For example, when each instance is only 1 second long, the accuracy of these methods are only 0.8650, 0.8848, 0.7537, and 0.7886, respectively. It can be explained by the silence in the audio instances. When the audio instance is very short, each instance contains limited informative acoustic features. Therefore, short audio instances are more vulnerable to noises such as silence. In such scenarios, the frequency dynamics over time captured by LSTM and self-attention-based methods are less reliable and robust than the local features captured by CNN-based methods. The raw waveform-based method, SC, does not work very well generally compared with MFCC-based methods. SC skips the construction of MFCC, which involves fast Fourier transform and other hard-to-learn procedures, to learn speaker identification. It is still a daunting task since raw waveform-based methods are deemed to require a huge amount of training data in order to achieve success.

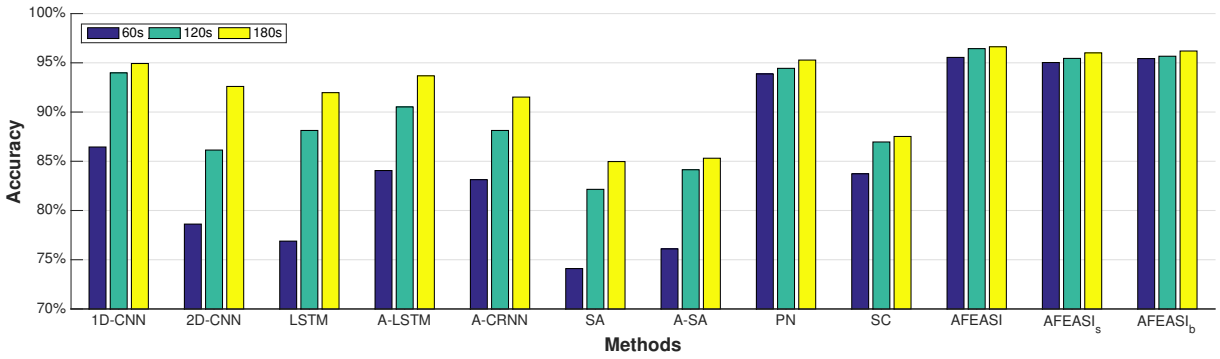


Figure 9.12: The accuracy of each method with different total training data per speaker on LibriSpeech.

9.4.3 Performance with Limited Training Data

In this section, we investigate the performance of all methods when facing a shortage of data for training. In order to make instant identification response, we fix the length of each audio instance to 3 seconds. We vary the total number of training instances per speaker from 20 to 60. If the total training instances per speaker is only 20 and each instance is 3 seconds long, there are only 60 seconds audios used for training for each speaker. When we relax the number of training instances per speaker to 40 and 60, 120 seconds and 180 seconds long cumulative audios will be used for training per speaker, respectively. Figure 9.12 shows the accuracy performance for all methods on different settings.

We observe that the fewer training instances we have for a speaker, the lower accuracy we achieve for all methods. For example, when we have 180 seconds long training instances for a speaker, the accuracy of 1D-CNN can reach as high as about 0.9493. However, when the training instances are reduced to 120 and 60 seconds per speaker, the accuracy is only 0.9398 and 0.8645, respectively. We observe a similar performance drop for 2D-CNN, LSTM, A-LSTM, SA, A-SA, and SC. These observations demonstrate that the strong discriminative power of deep learning models significantly depends on the availability of a sufficient amount of training data. When only limited instances are provided for training, the performance might be far from expectations. Without applying few-shot learning mechanisms, 1D-CNN

achieves the best accuracy performance on the three settings. This results from its simple network structure design, which is light-parameter dependent. The prototypical network, a metric-learning-based few-shot leaning method, achieves slightly higher accuracy. Its accuracy reaches about 0.93 as 60 seconds training instances are present for training per speaker. The high accuracy performance demonstrates the advantage of adopting few-shot learning, which fully utilizes the limited instances during training. Among all methods, *AFEASI* and its variants achieve the highest accuracy on all three settings, especially when the training instances are limited. Its accuracy is as high as about 0.95 for the setting of 60 seconds per speaker. This demonstrates the effectiveness of adopting attentive few-shot learning and adversarial learning when training from limited data.

9.4.4 Perturbation Injection Choice

In this section, we investigate and compare the effectiveness of different choices for injecting adversarial perturbations. *AFEASI* only injects perturbations into query instances. $AFEASI_s$ only injects perturbations into support instances, while $AFEASI_b$ injects dynamic perturbations into both query and support instances.

The last three groups of Figure 9.12 show the accuracy performance of *AFEASI* and its two variants on different settings. We observe that injecting adversarial perturbations into query instances only is effective enough to enhance identifications. For example, as training instances are as limited as 60 seconds per speaker. The accuracy of *AFEASI*, $AFEASI_b$, and $AFEASI_s$ are all as high as and close to 0.955. While injecting perturbations into only query instances allows us to quickly generate adversarial examples during training, as compared to generating noises in support instances or both query and support instances. Therefore, in this work we choose to inject perturbations into only query instances in consideration of computational efficiency.

Table 9.8: Accuracy on test set by injecting Gaussian noise

Methods	$AFEASI_{gaussian}$					*AFEASI*	*AFEASI_*
τ	1e-6	1e-5	1e-4	1e-3	1e-2	N/A	N/A
60s	0.9432	0.9498	0.9481	0.9376	0.9317	0.9555	0.9411
120s	0.9522	0.9556	0.9532	0.9491	0.9487	0.9644	0.9512
180s	0.9587	0.9630	0.9620	0.9553	0.9531	0.9663	0.9563

9.4.5 Effectiveness of Adversarial Training

In this section, we compare the effectiveness of data augmentation between conducting adversarial training and applying conventional audio augmentation methods.

To conduct conventional data augmentation, we inject random Gaussian noises to raw audios with different parameters τ that controls the intensity of injected noises. Formally, the augmented audio piece x_{au} can be represented as $x_{au} = x + \tau N_g$, where x and N_g are the original audio and the Gaussian noise, respectively. Finally, $AFEASI_{gaussian}$ denotes the conventional approach by replace the adversarial training with the participation of augmented data injected by Gaussian noises. In addition, we use *AFEASI_* to indicate the method simply removing adversarial training from *AFEASI* and evaluate the impact of adversarial training.

Table 9.8 shows the performance comparisons among *AFEASI_*, $AFEASI_{gaussian}$, and *AFEASI*. We observe that data augmentations by injecting Gaussian noises help address the data shortage issue. For example, the accuracy of *AFEASI_* is 0.9411 for the 60-seconds setting. When setting τ to $1e-5$ and $1e-4$, the accuracy of $AFEASI_{gaussian}$ improves to 0.9498 and 0.9841, respectively. However, the effectiveness of such data synthesis is sensitive to the setting of the weighting factor τ. For example, when τ is set to $1e-3$ and $1-e2$, heavier noises are injected into the raw audios. The injected noises obscure the raw informative signals and lead to worse accuracy performance. This could make it challenging to select a good τ in practice since it only helps with a narrow range of effective settings. In addition, we notice that *AFEASI* still achieves the highest accuracy performance over

all three settings. This demonstrates that intentional adversarial noises are more helpful in improving identification performance.

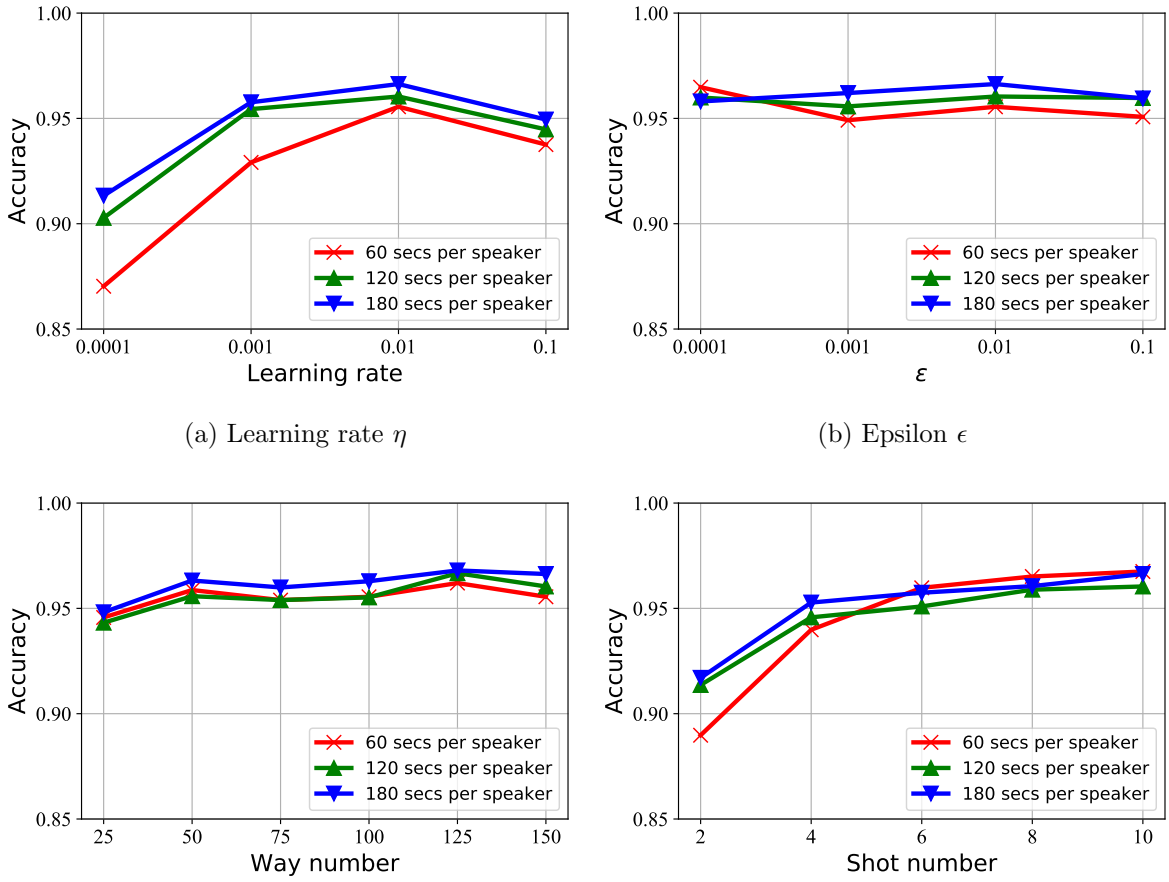
9.4.6 Sensitivity Study

Table 9.9: Main parameters of AFEASI in the experiments after fine-tuning.

Parameters	Value	Parameters	Value
Learning rate η	0.01	Number of epochs	20
Regularizer weight λ	1	Perturbation bound ϵ	0.01
Way number K	150	Shot Number N	10

In this section, we examine how different choices of parameters influence the performance of *AFEASI*. Except for the parameter being tested, we set other parameters at the default values (see in Table 9.9). Figure 9.13 shows the evaluation results as a function of one selected parameter when fixing others. Overall, we observe that *AFEASI* is not strictly sensitive to most parameters, which demonstrates the robustness of *AFEASI*.

Figure 9.13a shows the accuracy performances of *AFEASI* when we change the learning rate. It may get stuck to local optimal and lead to sub-optimal performance when the learning rate is either too small or too large. In this work, we set it as 0.01 with the consideration of the performance. Figure 9.13b shows the effect of varying ϵ, which controls the magnitude of the perturbations. *AFEASI* in general is not sensitive to the setting of ϵ and it achieves high accuracy performance with a wide range of ϵ from 0.0001 to 0.1. Figure 9.13c shows the performance of *AFEASI* when choosing different number of speakers in a training episode. We observe that *AFEASI* is not strictly sensitive to this parameter and it always achieves accuracy performance as high as around 0.95 for all settings with the parameter is larger than 25. Figure 9.13d shows the performance change via choosing different number of instances per speaker as references in the support module. We observe that the more instances we select to generate the speaker's acoustic biometric embedding as references, the higher accuracy we can achieve during the test in general. We also notice that



(a) Learning rate η

(b) Epsilon ϵ

(c) Way number K

(d) Shot number N

Figure 9.13: Parameter sensitivity studies on LibriSpeech

the increase of accuracy performance saturates as the number of shots increases more than 6. This is because: at the beginning, a larger value of the shot number N brings a stronger representation power to express speaker's acoustic characteristic, but the further increase of shot number might only provide limited and repeated information.

9.4.7 Household Deployment

One notable application of ASR is to enable personalized services at different households. In such scenarios, audio-enabled devices, such as Echo Dot and Google Home, only need to serve several peoples in a household. Therefore, we may not have to include all the speakers as

identification candidates. This could not only reduce the computation cost during inference and respond more quickly, but also significantly improve the identification accuracy. All these benefits depend on the flexibility of the identification model to accommodate only a portion of users as speaker candidates. All conventional deep learning methods mentioned in the baseline section fail to achieve this, since the output layer of these models is fixed with the number of neurons equal to the number of total speakers. *AFEASI* solves the issue by learning distance metric among different speaker candidates. The speaker with the smallest distance to the query instance in the hidden space among all candidates yields the prediction. In this way, *AFEASI* can enable fast and efficient speaker identification by considering only a small set of candidate speakers.

9.5 Automatic Speaker Identification with Gradient-based Few-shot Learning

We conduct experiments on LibriSpeech data set to evaluate the performance of *MDNML* against four popular algorithms.

9.5.1 Experimental Settings

The experiments are conducted on the LibriSpeech dataset, which is publicly available[3]. For the dataset, 75% of speakers are treated as existing users and the remaining 25% of speakers are treated as new users for the purpose of evaluation. We follow the previous work [45] to extract acoustic features from the raw audios. The first 20 MFCCs are extracted from speech after an energy-based voice activity detection. A 44 kHz sampling rate and a 25 ms hamming window with a 10 ms frame shift are used during the MFCC construction.

[3]LibriSpeech: `http://www.openslr.org/12`

9.5.2 Baselines

To evaluate the performance of *MDNML*, the following four methods are adopted as baselines.

- **MDN** [107] trains acoustic profiles for each new user from scratch without any knowledge gained from existing users.
- **PN** [45] utilizes the CNN-based prototypical network, a metric-learning-based few-shot technique, to conduct speaker identification.
- **PNL** [109] relies on Bi-LSTM-based prototypical network to perform speaker identification.
- **AFEASI** [106] applies adversarial training on prototypical network to achieve speaker identification.

9.5.3 Identification Performance

In this section, we evaluate the performances of *MDNML* against different baseline methods on the LibriSpeech dataset. We adopt household-level accuracy as the evaluation metric. The household-level accuracy first calculates the identification accuracy in each household and then averages the identification accuracy in each household by treating the importance of them equally.

Table 9.10: Accuracy with 2 seconds voice enrollment.

Utterance duration in test	1s	2s	3s	4s
MDN	80.0%	83.4%	86.2%	87.8%
PN	85.8%	86.0%	88.0%	89.4%
PNL	82.8%	86.4%	85.8%	85.0%
AFEASI	86.6%	86.6%	89.4%	90.2%
MDNML	88.6%	88.6%	90.2%	91.4%

To imitate the scenarios of serving new users, we set the duration of each enrollment utterance, which is used for profile adaptions for the new users, to small values, varying

Table 9.11: Accuracy with 4 seconds voice enrollment.

Utterance duration in test	1s	2s	3s	4s
MDN	85.8%	86.4%	88.2%	89.0%
PN	88.0%	87.6%	88.8%	89.2%
PNL	84.8%	86.0%	86.8%	87.2%
AFEASI	88.6%	89.0%	89.2%	89.8%
MDNML	89.6%	90.4%	92.2%	93.0%

from 2 to 4 seconds. Moreover, in order to offer instant identification response, we vary the duration of each test voice utterance from 1 second to 4 seconds. Tables 9.10 and 9.11 show the performance of different methods on the LibriSpeech dataset.

We have four observations from the results on the LibriSpeech dataset. First, the longer the duration of a voice utterance for identification in test, the higher accuracy each method can achieve. For example, when we have 2 seconds long voice utterance as the enrollment for profile adaption and 1 second long voice utterance to identify the speakers with testing data, MDN can reach only 80% accuracy; however, the accuracy increases to 83.4%, 86.2%, and 87.8% when the test voice utterance becomes to 2, 3, and 4 seconds, respectively. Note that this observation generally applies to all methods. The performance gain stems from the fact that the acoustic signals embedded in long voice utterances are more consistent and reliable. These consistent and reliable acoustic signals further yield confident speaker identifications, which are more accurate. Second, the longer the voice enrollment we have for a new user during the adaption process, the higher accuracy each method can achieve. It makes sense because the longer the voice enrollment for a user, the richer signals we can use to construct his/her acoustic profile by analyzing the enrolled utterance. Profile constructions, supported by rich acoustic information, contribute to the high accuracy. Third, we observe that PN, PNL, AFEASI, and *MDNML* generally outperform MDN. This shows the advantages of utilizing few-shot learning, which allows effective learning even with limited data. Fourth, *MDNML* consistently achieves the highest accuracy comparing with the four baselines in

all settings. It demonstrates the effectiveness of *MDNML*, which leverages knowledge learnt from existing users.

CHAPTER 10

Conclusion

In this thesis, we investigate how to mine patterns and knowledge from sparse and deficient data. More specifically, we dive into three concrete applications, i.e., location-based customer recommendation in social networks, query recommendation on search engines, and automatic speaker recognition.

For customer recommendation in location-based social networks, we propose two methods, i.e., *CORALS* and *SEATLE*. CORALS utilizes auxiliary information (i.e., geographical coordinates of businesses and online reviews) as extra guidance to make recommendations, while *SEATLE* focuses on incorporating not only geographical convenience (between customers and businesses) but also geographical dependency (among businesses) to make recommendations. Moreover, metric-learning-based few-shot learning is further adopted to improve recommendation performance from the limited training data. For query recommendation on search engines, we propose a context-aware method *CFAN*, which is capable of modeling both users' search query sequence and click sequence, to make query suggestions. Both users' formulated search queries and their clicks are incorporated to model their information need and collectively modeling these two factors allows us to understand users' search intent more accurately. In addition, adversarial training is introduced to improve the robustness and generalization of the recommendation model in the query suggestion model. For automatic speaker recognition, we investigate two types of few-shot learning strategies, i.e., metric-learning-based and gradient-based ones, and propose *AFEASI* and *MDNML*, respectively. *AFEASI* applies metric-learning-based few-shot learning to fully utilize the limited voice utterances and further relies on adversarial training, as a data augmentation technique, to achieve robust and well-generalized speaker identification. *MDNML* is proposed to achieve

automatic speaker identification with the goal of serving new users. It depends on mixture density networks, which are gradient-friendly and can model voice utterances with arbitrary lengths, to construct users' acoustic profiles. To leverage the identification knowledge learnt from existing users, model-agnostic meta-learning, a gradient-based meta-learning technique, is adopted, which can help us learn a set of well-initialized model parameters. Therefore, the acoustic profiles of new users could be constructed much faster based on the learnt model parameters.

REFERENCES

[1] R. Li, B. Kao, B. Bi, R. Cheng, and E. Lo, "DQR: a probabilistic approach to diversified query recommendation," in *21st ACM International Conference on Information and Knowledge Management, CIKM'12, Maui, HI, USA, October 29 - November 02, 2012*, pp. 16–25, 2012.

[2] R. Li, L. Li, X. Wu, Y. Zhou, and W. Wang, "Click feedback-aware query recommendation using adversarial examples," in *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pp. 2978–2984, 2019.

[3] J. Wen, J. Nie, and H. Zhang, "Clustering user queries of a search engine," in *Proceedings of the Tenth International World Wide Web Conference, WWW 10, Hong Kong, China, May 1-5, 2001*, pp. 162–168, 2001.

[4] J. Jiang and W. Wang, "RIN: reformulation inference network for context-aware query suggestion," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*, pp. 197–206, 2018.

[5] A. Sordoni, Y. Bengio, H. Vahabi, C. Lioma, J. G. Simonsen, and J. Nie, "A hierarchical recurrent encoder-decoder for generative context-aware query suggestion," in *Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, October 19 - 23, 2015*, pp. 553–562, 2015.

[6] M. Dehghani, S. Rothe, E. Alfonseca, and P. Fleury, "Learning to attend, copy, and generate for session-based query suggestion," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*, pp. 1747–1756, 2017.

[7] W. Xie, A. Nagrani, J. S. Chung, and A. Zisserman, "Utterance-level aggregation for speaker recognition in the wild," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019*, pp. 5791–5795, 2019.

[8] M. Hajibabaei and D. Dai, "Unified hypersphere embedding for speaker recognition," *CoRR*, vol. abs/1807.08312, 2018.

[9] M. Ravanelli and Y. Bengio, "Speaker recognition from raw waveform with sincnet," in *2018 IEEE Spoken Language Technology Workshop, SLT 2018, Athens, Greece, December 18-21, 2018*, pp. 1021–1028, 2018.

[10] C. Li, X. Ma, B. Jiang, X. Li, X. Zhang, X. Liu, Y. Cao, A. Kannan, and Z. Zhu, "Deep speaker: an end-to-end neural speaker embedding system," *CoRR*, vol. abs/1705.02304, 2017.

[11] J. S. Chung, A. Nagrani, and A. Zisserman, "Voxceleb2: Deep speaker recognition," *CoRR*, vol. abs/1806.05622, 2018.

[12] H. Gao, J. Tang, X. Hu, and H. Liu, "Exploring temporal effects for location recommendation on location-based social networks," in *RecSys '13, Hong Kong, China, October 12-16, 2013*, 2013.

[13] M. Ye, P. Yin, W. Lee, and D. L. Lee, "Exploiting geographical influence for collaborative point-of-interest recommendation," in *SIGIR '11, Beijing, China, July 25-29, 2011*, 2011.

[14] Q. Yuan, G. Cong, Z. Ma, A. Sun, and N. Magnenat-Thalmann, "Time-aware point-of-interest recommendation," in *SIGIR '13, Dublin, Ireland - July 28 - August 01, 2013*, 2013.

[15] B. Liu, Y. Fu, Z. Yao, and H. Xiong, "Learning geographical preferences for point-of-interest recommendation," in *KDD '13, Chicago, IL, USA, August 11-14, 2013*, 2013.

[16] H. Li, Y. Ge, R. Hong, and H. Zhu, "Point-of-interest recommendations: Learning potential check-ins from friends," in *SIGKDD '16, San Francisco, CA, USA, August 13-17, 2016*, 2016.

[17] H. Gao, J. Tang, X. Hu, and H. Liu, "Content-aware point of interest recommendation on location-based social networks," in *AAAI '15, January 25-30, 2015, Austin, Texas, USA.*, pp. 1721–1727, 2015.

[18] D. Lian, Y. Ge, F. Zhang, N. J. Yuan, X. Xie, T. Zhou, and Y. Rui, "Content-aware collaborative filtering for location recommendation based on human mobility data," in *ICDM '15, Atlantic City, NJ, USA, November 14-17, 2015*, pp. 261–270, 2015.

[19] H. Yin, B. Cui, Y. Sun, Z. Hu, and L. Chen, "LCARS: A spatial item recommender system," *ACM Trans. Inf. Syst.*, vol. 32, no. 3, 2014.

[20] B. Hu and M. Ester, "Social topic modeling for point-of-interest recommendation in location-based social networks," in *ICDM '14, Shenzhen, China, December 14-17, 2014*, pp. 845–850, 2014.

[21] J. Zhang and C. Chow, "Geosoca: Exploiting geographical, social and categorical correlations for point-of-interest recommendations," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015*, pp. 443–452, 2015.

[22] S. Wang, Y. Wang, J. Tang, K. Shu, S. Ranganath, and H. Liu, "What your images reveal: Exploiting visual contents for point-of-interest recommendation," in *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*, pp. 391–400, 2017.

[23] C. Cheng, H. Yang, I. King, and M. R. Lyu, "Fused matrix factorization with geographical and social influence in location-based social networks," in *AAAI '12, July 22-26, 2012, Toronto, Ontario, Canada.*, 2012.

[24] M. Lichman and P. Smyth, "Modeling human location data with mixtures of kernel densities," in *KDD '14, New York, USA - August 24 - 27, 2014*, 2014.

[25] J. Zhang and C. Chow, "igslr: personalized geo-social location recommendation: a kernel density estimation approach," in *SIGSPATIAL '13, Orlando, FL, USA, November 5-8, 2013*, 2013.

[26] Y. Liu, W. Wei, A. Sun, and C. Miao, "Exploiting geographical neighborhood characteristics for location recommendation," in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014*, pp. 739–748, 2014.

[27] C. Yang, L. Bai, C. Zhang, Q. Yuan, and J. Han, "Bridging collaborative filtering and semi-supervised learning: A neural approach for POI recommendation," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*, pp. 1245–1254, 2017.

[28] C. Ma, Y. Zhang, Q. Wang, and X. Liu, "Point-of-interest recommendation: Exploiting self-attentive autoencoders with neighbor-aware influence," in *CIKM*, pp. 697–706, ACM, 2018.

[29] F. Zhou, R. Yin, K. Zhang, G. Trajcevski, T. Zhong, and J. Wu, "Adversarial point-of-interest recommendation," in *WWW, San Francisco, CA, USA, May 13-17, 2019*, pp. 3462–34618, 2019.

[30] P. Boldi, F. Bonchi, C. Castillo, D. Donato, A. Gionis, and S. Vigna, "The query-flow graph: model and applications," in *Proceedings of the 17th ACM Conference on Information and Knowledge Management, CIKM 2008, Napa Valley, California, USA, October 26-30, 2008*, pp. 609–618, 2008.

[31] F. Bonchi, R. Perego, F. Silvestri, H. Vahabi, and R. Venturini, "Efficient query recommendations in the long tail via center-piece subgraphs," in *The 35th International ACM SIGIR conference on research and development in Information Retrieval, SIGIR '12, Portland, OR, USA, August 12-16, 2012*, pp. 345–354, 2012.

[32] R. A. Baeza-Yates and A. Tiberi, "Extracting semantic relations from query logs," in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Jose, California, USA, August 12-15, 2007*, pp. 76–85, 2007.

[33] Q. Mei, D. Zhou, and K. W. Church, "Query suggestion using hitting time," in *Proceedings of the 17th ACM Conference on Information and Knowledge Management, CIKM 2008, Napa Valley, California, USA, October 26-30, 2008*, pp. 469–478, 2008.

[34] H. Cao, D. Jiang, J. Pei, Q. He, Z. Liao, E. Chen, and H. Li, "Context-aware query suggestion by mining click-through and session data," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008*, pp. 875–883, 2008.

[35] Q. He, D. Jiang, Z. Liao, S. C. H. Hoi, K. Chang, E. Lim, and H. Li, "Web query recommendation via sequential query prediction," in *Proceedings of the 25th International Conference on Data Engineering, ICDE 2009, March 29 2009 - April 2 2009, Shanghai, China*, pp. 1443–1454, 2009.

[36] J. Jiang, Y. Ke, P. Chien, and P. Cheng, "Learning user reformulation behavior for query auto-completion," in *The 37th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '14, Gold Coast , QLD, Australia - July 06 - 11, 2014*, pp. 445–454, 2014.

[37] J. Jiang and P. Cheng, "Classifying user search intents for query auto-completion," in *Proceedings of the 2016 ACM on International Conference on the Theory of Information Retrieval, ICTIR 2016, Newark, DE, USA, September 12- 6, 2016*, pp. 49–58, 2016.

[38] B. Wu, C. Xiong, M. Sun, and Z. Liu, "Query suggestion with feedback memory network," in *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, pp. 1563–1571, 2018.

[39] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Trans. Audio, Speech & Language Processing*, vol. 19, no. 4, pp. 788–798, 2011.

[40] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, "Speaker verification using adapted gaussian mixture models," *Digital Signal Processing*, vol. 10, no. 1-3, pp. 19–41, 2000.

[41] F. Wang, W. Liu, H. Dai, H. Liu, and J. Cheng, "Additive margin softmax for face verification," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*, 2018.

[42] H. Bredin, "Tristounet: Triplet loss for speaker turn embedding," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2017, New Orleans, LA, USA, March 5-9, 2017*, pp. 5430–5434, 2017.

[43] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu, "Learning fine-grained image similarity with deep ranking," in *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, pp. 1386–1393, 2014.

[44] L. Wan, Q. Wang, A. Papir, and I. Lopez-Moreno, "Generalized end-to-end loss for speaker verification," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018, Calgary, AB, Canada, April 15-20, 2018*, pp. 4879–4883, 2018.

[45] P. Anand, A. K. Singh, S. Srivastava, and B. Lall, "Few shot speaker recognition using deep neural networks," *CoRR*, vol. abs/1904.08775, 2019.

[46] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *CoRR*, vol. abs/1312.6199, 2013.

[47] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *CoRR*, vol. abs/1412.6572, 2014.

[48] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *CoRR*, vol. abs/1607.02533, 2016.

[49] R. Jia and P. Liang, "Adversarial examples for evaluating reading comprehension systems," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pp. 2021–2031, 2017.

[50] J. Li, W. Monroe, and D. Jurafsky, "Understanding neural networks through representation erasure," *CoRR*, vol. abs/1612.08220, 2016.

[51] M. Alzantot, Y. Sharma, A. Elgohary, B. Ho, M. B. Srivastava, and K. Chang, "Generating natural language adversarial examples," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pp. 2890–2896, 2018.

[52] T. Miyato, S.-i. Maeda, M. Koyama, K. Nakae, and S. Ishii, "Distributional smoothing with virtual adversarial training," *arXiv preprint arXiv:1507.00677*, 2015.

[53] T. Miyato, A. M. Dai, and I. Goodfellow, "Adversarial training methods for semi-supervised text classification," *arXiv preprint arXiv:1605.07725*, 2016.

[54] B. Poole, J. Sohl-Dickstein, and S. Ganguli, "Analyzing noise in autoencoders and deep networks," *CoRR*, vol. abs/1406.1831, 2014.

[55] Z. Zhao, D. Dua, and S. Singh, "Generating natural adversarial examples," *CoRR*, vol. abs/1710.11342, 2017.

[56] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *ICML Deep Learning Workshop*, vol. 2, 2015.

[57] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 3630–3638, 2016.

[58] J. Snell, K. Swersky, and R. S. Zemel, "Prototypical networks for few-shot learning," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pp. 4080–4090, 2017.

[59] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. S. Torr, and T. M. Hospedales, "Learning to compare: Relation network for few-shot learning," in *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 1199–1208, 2018.

[60] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. P. Lillicrap, "Meta-learning with memory-augmented neural networks," in *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pp. 1842–1850, 2016.

[61] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," 2016.

[62] S. Fort, "Gaussian prototypical networks for few-shot learning on omniglot," *CoRR*, vol. abs/1708.02735, 2017.

[63] V. G. Satorras and J. B. Estrach, "Few-shot learning with graph neural networks," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[64] Y. Duan, M. Andrychowicz, B. C. Stadie, J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba, "One-shot imitation learning," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pp. 1087–1098, 2017.

[65] M. Yu, X. Guo, J. Yi, S. Chang, S. Potdar, Y. Cheng, G. Tesauro, H. Wang, and B. Zhou, "Diverse few-shot text classification with multiple metrics," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pp. 1206–1215, 2018.

[66] W. Xiong, M. Yu, S. Chang, X. Guo, and W. Y. Wang, "One-shot relational learning for knowledge graphs," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pp. 1980–1990, 2018.

[67] Yelp, "Yelp for Business Owners." `https://biz.yelp.com/support/what_is_yelp`, 2017.

[68] C. Smith, "By the numbers: 20 important Foursquare Stats." `http://expandedramblings.com/index.php/by-the-numbers-interesting-foursquare-user-stats/`, 2016.

[69] J. Constine, "Facebook relaunches Events app as Facebook Local, adds bars and food." `https://techcrunch.com/2017/11/10/facebook-local/`, 2017.

[70] M. Osman, "28 Powerful Facebook Stats Your Brand Can't Ignore in 2018." `https://sproutsocial.com/insights/facebook-stats-for-marketers/`, 2018.

[71] J. Antin, M. de Sá, and E. F. Churchill, "Local experts and online review sites," in *CSCW '12, Seattle, WA, USA, February 11-15, 2012 - Companion Volume*, 2012.

[72] BrightLocal, "Local Consumer Review Survey." `https://www.brightlocal.com/learn/local-consumer-review-survey/`, 2016.

[73] M. Xie, H. Yin, H. Wang, F. Xu, W. Chen, and S. Wang, "Learning graph-based POI embedding for location-based recommendation," in *CIKM '16, Indianapolis, IN, USA, October 24-28, 2016*, 2016.

[74] W. R. Tobler, "A computer movie simulating urban growth in the detroit region," *Economic geography*, vol. 46, no. sup1, 1970.

[75] J. Weston, S. Bengio, and N. Usunier, "WSABIE: scaling up to large vocabulary image annotation," in *IJCAI '11, Barcelona, Catalonia, Spain, July 16-22, 2011*, 2011.

[76] J. C. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

[77] D. A. Reynolds, "Gaussian mixture models," in *Encyclopedia of Biometrics*, 2009.

[78] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the royal statistical society. Series B (methodological)*, 1977.

[79] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *science*, vol. 315, no. 5814, 2007.

[80] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents," in *ICML '14, Beijing, China, 21-26 June 2014*, 2014.

[81] Z. Yao, Y. Fu, B. Liu, Y. Liu, and H. Xiong, "POI recommendation: A temporal matching between POI popularity and user regularity," in *ICDM '16, December 12-15, 2016, Barcelona, Spain*, 2016.

[82] Y. Liu, C. Liu, B. Liu, M. Qu, and H. Xiong, "Unified point-of-interest recommendation with temporal interval assessment," in *SIGKDD '16, San Francisco, CA, USA, August 13-17, 2016*, 2016.

[83] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *CoRR*, vol. abs/1609.02907, 2016.

[84] J. Vanschoren, *Meta-learning*, pp. 39–68. Germany: Springer, 2019.

[85] W. Xiong, M. Yu, S. Chang, X. Guo, and W. Y. Wang, "One-shot relational learning for knowledge graphs," *CoRR*, vol. abs/1808.09040, 2018.

[86] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pp. 6000–6010, 2017.

[87] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 770–778, 2016.

[88] R. Baraglia, C. Castillo, D. Donato, F. M. Nardini, R. Perego, and F. Silvestri, "Aging effects on query flow graphs for query suggestion," in *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, China, November 2-6, 2009*, pp. 1947–1950, 2009.

[89] B. J. Jansen, A. Spink, and T. Saracevic, "Real life, real users, and real needs: a study and analysis of user queries on the web," *Inf. Process. Manage.*, vol. 36, no. 2, pp. 207–227, 2000.

[90] M. Sanderson, "Ambiguous queries: test collections need more sense," in *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2008, Singapore, July 20-24, 2008*, pp. 499–506, 2008.

[91] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[92] A. Bordes, J. Weston, and N. Usunier, "Open question answering with weakly supervised embedding models," in *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2014, Nancy, France, September 15-19, 2014. Proceedings, Part I*, pp. 165–180, 2014.

[93] A. Severyn and A. Moschitti, "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, Santiago, Chile, August 9-13, 2015*, pp. 373–382, 2015.

[94] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[95] B. Kinsella, "Smart Speaker Owners Use Voice Assistants Nearly 3 Times Per Day." `https://voicebot.ai/2018/04/02/smart-speaker-owners-use-voice-assistants-nearly-3-times-per-day/`, 2018.

[96] R. Feldman, "Almost a quarter of Britons now own one or more smart home devices." `https://yougov.co.uk/topics/technology/articles-reports/2018/08/10/almost-quarter-britons-now-own-one-or-more-smart-h`, 2018.

[97] V. M. Patel, R. Chellappa, D. Chandra, and B. Barbello, "Continuous user authentication on mobile devices: Recent progress and remaining challenges," *IEEE Signal Process. Mag.*, vol. 33, no. 4, pp. 49–61, 2016.

[98] J. H. L. Hansen and T. Hasan, "Speaker recognition by machines and humans: A tutorial review," *IEEE Signal Process. Mag.*, vol. 32, no. 6, pp. 74–99, 2015.

[99] J. Kim and J. Park, "Multistage data selection-based unsupervised speaker adaptation for personalized speech emotion recognition," *Eng. Appl. of AI*, vol. 52, pp. 126–134, 2016.

[100] Z. Liu, Z. Wu, T. Li, J. Li, and C. Shen, "GMM and CNN hybrid method for short utterance speaker recognition," *IEEE Trans. Industrial Informatics*, vol. 14, no. 7, pp. 3244–3252, 2018.

[101] H. Yakura and J. Sakuma, "Robust audio adversarial example for a physical attack," *CoRR*, vol. abs/1810.11793, 2018.

[102] N. Carlini and D. A. Wagner, "Audio adversarial examples: Targeted attacks on speech-to-text," in *2018 IEEE Security and Privacy Workshops, SP Workshops 2018, San Francisco, CA, USA, May 24, 2018*, pp. 1–7, 2018.

[103] M. Xu, L. Duan, J. Cai, L. Chia, C. Xu, and Q. Tian, "Hmm-based audio keyword generation," in *Advances in Multimedia Information Processing - PCM 2004, 5th Pacific Rim Conference on Multimedia, Tokyo, Japan, November 30 - December 3, 2004, Proceedings, Part III*, pp. 566–574, 2004.

[104] S. Park, J. Park, S. Shin, and I. Moon, "Adversarial dropout for supervised and semi-supervised learning," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 3917–3924, 2018.

[105] X. He, Z. He, X. Du, and T. Chua, "Adversarial personalized ranking for recommendation," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, pp. 355–364, 2018.

[106] R. Li, J. Jiang, J. Liu, C. Hsieh, and W. Wang, "Automatic Speaker Recognition with Limited Data," in *Proceedings of WSDM, Houston, Texas, USA, February 3-7*, 2020.

[107] C. M. Bishop, "Mixture Density Networks," 1994.

[108] C. Finn, P. Abbeel, and S. Levine, "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks," in *Proceedings of ICML, Sydney, NSW, Australia, 6-11 August*, pp. 1126–1135, 2017.

[109] J. Wang, K. Wang, M. Law, F. Rudzicz, and M. Brudno, "Centroid-based Deep Metric Learning For Speaker Recognition," in *Proceedings of ICASSP, Brighton, United Kingdom, May 12-17*, pp. 3652–3656, 2019.

[110] Z. Liu, Z. Wu, T. Li, J. Li, and C. Shen, "GMM and CNN hybrid method for short utterance speaker recognition," *IEEE Transactions on Industrial informatics*, vol. 14, no. 7, pp. 3244–3252, 2018.

[111] A. Maurya, D. Kumar, and R. Agarwal, "Speaker recognition for Hindi speech signal using MFCC-GMM approach," *Procedia Computer Science*, vol. 125, pp. 880–887, 2018.

[112] O. Novotnỳ, O. Plchot, O. Glembek, L. Burget, and P. Matějka, "Discriminatively re-trained i-vector extractor for speaker recognition," in *Proceedings of ICASSP, Brighton, United Kingdom, May 12-17*, pp. 6031–6035, 2019.

[113] L. Juvela, B. Bollepalli, X. Wang, H. Kameoka, M. Airaksinen, J. Yamagishi, and P. Alku, "Speech Waveform Synthesis from MFCC Sequences with Generative Adversarial Networks," in *Proceedings of ICASSP, Calgary, AB, Canada, April 15-20, 2018*, pp. 5679–5683, 2018.

[114] R. Fu, J. Tao, Z. Wen, and Y. Zheng, "Phoneme dependent speaker embedding and model factorization for multi-speaker speech synthesis and adaptation," in *Proceedings of ICASSP, Brighton, United Kingdom, May 12-17*, pp. 6930–6934, 2019.

[115] L. Juvela, B. Bollepalli, X. Wang, H. Kameoka, M. Airaksinen, J. Yamagishi, and P. Alku, "Speech waveform synthesis from mfcc sequences with generative adversarial networks," in *Proceedings of ICASSP, Calgary, AB, Canada, April 15-20*, pp. 5679–5683, 2018.

[116] K. Vythelingum, Y. Estève, and O. Rosec, "Acoustic-dependent phonemic transcription for text-to-speech synthesis.," in *Interspeech*, pp. 2489–2493, 2018.

[117] B. J. Jansen, A. Spink, C. Blakely, and S. Koshman, "Defining a session on web search engines," *JASIST*, vol. 58, no. 6, pp. 862–871, 2007.

[118] M. Shokouhi, "Learning to personalize query auto-completion," in *The 36th International ACM SIGIR conference on research and development in Information Retrieval, SIGIR '13, Dublin, Ireland - July 28 - August 01, 2013*, pp. 103–112, 2013.

[119] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *(ICDM '08), December 15-19, 2008, Pisa, Italy*, 2008.

[120] M. Weimer, A. Karatzoglou, and A. J. Smola, "Improving maximum margin matrix factorization," *Machine Learning*, vol. 72, no. 3, 2015.

[121] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: bayesian personalized ranking from implicit feedback," in *UAI '09, Montreal, QC, Canada, June 18-21, 2009*, 2009.

[122] M. Weimer, A. Karatzoglou, Q. V. Le, and A. J. Smola, "COFI RANK - maximum margin matrix factorization for collaborative ranking," in *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, 2007.

[123] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, N. Oliver, and A. Hanjalic, "Climf: learning to maximize reciprocal rank with collaborative less-is-more filtering," in *RecSys '12, Dublin, Ireland, September 9-13, 2012*, 2012.

[124] J. Weston, H. Yee, and R. J. Weiss, "Learning to rank recommendations with the k-order statistic loss," in *RecSys '13, Hong Kong, China, October 12-16, 2013*, 2013.

[125] D. Lian, C. Zhao, X. Xie, G. Sun, E. Chen, and Y. Rui, "Geomf: joint geographical modeling and matrix factorization for point-of-interest recommendation," in *KDD '14, New York, USA - August 24 - 27, 2014*, 2014.

[126] X. Li, G. Cong, X. Li, T. N. Pham, and S. Krishnaswamy, "Rank-geofm: A ranking based geographical factorization method for point of interest recommendation," in *SIGIR '15, Santiago, Chile, August 9-13, 2015*, pp. 433–442, 2015.

[127] G. Hinton, N. Srivastava, and K. Swersky, "Overview of mini-batch gradient descent." `http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf`, 2012.

[128] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: user movement in location-based social networks," in *SIGKDD '11, San Diego, CA, USA, August 21-24, 2011*, 2011.

[129] R. Li, J. Jiang, C. Ju, and W. Wang, "Corals: Who are my potential new customers? tapping into the wisdom of customers' decisions," in *WSDM '19, Melbourne, Australia, February 11-15, 2019*, 2019.

[130] E. M. Voorhees, "The TREC-8 question answering track report," in *Proceedings of The Eighth Text REtrieval Conference, TREC 1999, Gaithersburg, Maryland, USA, November 17-19, 1999*, 1999.

[131] D. Yin, Y. Hu, J. Tang, T. D. Jr., M. Zhou, H. Ouyang, J. Chen, C. Kang, H. Deng, C. Nobata, J. Langlois, and Y. Chang, "Ranking relevance in yahoo search," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pp. 323–332, 2016.

[132] C. J. Burges, "From ranknet to lambdarank to lambdamart: An overview," *Learning*, vol. 11, no. 23-581, p. 81, 2010.

[133] Z. Bar-Yossef and N. Kraus, "Context-sensitive query auto-completion," in *Proceedings of the 20th International Conference on World Wide Web, WWW 2011, Hyderabad, India, March 28 - April 1, 2011*, pp. 107–116, 2011.

[134] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[135] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[136] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *KDD*, pp. 855–864, ACM, 2016.

[137] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[138] M. Lin, Q. Chen, and S. Yan, "Network in network," in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[139] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, pp. 1106–1114, 2012.

[140] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.