Empowering Search Engines: Exploring Transformer-Based Approaches for Query Auto-Completion

Alessandro Speggiorin

Master of Science
School of Informatics
University of Edinburgh
2023

Abstract

With the exponential increase of online content, search engines have assumed an essential role in order to help users search for relevant documents and multimedia resources. However, expressing the information need as a query is a challenging task due to the inherent ambiguity of languages and lack of context. Due to this fact, Query Auto-Completion (QAC) has become a prominent functionality offered by retrieval systems to help users formulate their search requirements in a well-defined and structured manner. To be more specific, given a query prefix, QAC systems present users with a ranked list of query completions that should align with their search intent. However, due to the long-tail nature of search queries, traditional neural and non-neural approaches often struggle to provide meaningful completions for previously unseen prefixes. Therefore, this project explores how the inherent nature of current State-of-the-Art pre-trained Transformer-based models can be leveraged to address the QAC task and provide meaningful completions for unseen prefixes. Our findings suggest that, despite the superior performance of non-neural methods, Transformer-based models can effectively tackle the QAC problem by generating completions for Out-of-Vocabulary tokens. However, the data format, the models' architecture, and the tokenisation strategy deeply affect the models' performance and behaviour.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Alessandro Speggiorin*)

Acknowledgements

I want to express my gratitude to my supervisor, Dr Walid Magdy, for his invaluable support and for guiding me throughout the duration of this research project. Moreover, I am thankful to Dr Amin Ahmad and Dr Suleman Kazi for their help and insightful feedback during our weekly meetings. I would also like to thank them for giving me access to additional resources required to conduct all the experiments presented in this report. I would like to also express my appreciation to Amr Keleg for helping me configure the server and solve technical difficulties. Lastly, a special acknowledgement goes to my parents and sisters for always being there when I needed them.

Table of Contents

Chapter 1

Introduction

1.1 Motivation & Project Scope

In the present day, search engines have assumed a critical role in facilitating users to effectively explore the exponentially growing online content by expressing their information need through free-text queries [12, 30]. However, the capability of search engines to retrieve relevant documents is limited by the semantic ambiguity and lack of context typical of free-text queries [8].

Consequently, Query Auto-Completion (QAC) has emerged as a prominent feature offered by search engines to guide users in formulating well-structured and error-free queries [8]. To be more specific, QAC systems aim to present users with a ranked list of candidate completions given a potentially partial input prefix [7]. For instance, given the input prefix *Order f*, viable completions could be *Order food online* and *Order flowers*. Hence, QAC systems aid users in formulating clear and concise queries, requiring minimal keystrokes, to find documents that align with their search requirements [79]. In light of this, diverse methods have been researched over time to address the QAC task. To be more specific, traditional QAC methods exploit hierarchical data structures in order to leverage statistical information derived from search logs to generate and rank query completions [7, 72]. However, while traditional QAC approaches often offer an effective solution to the problem, they tend to fail for previously unseen prefixes (Out of Vocabulary - OOV, tokens not present in the training data) [8, 78]. Therefore, to overcome these limitations, the utilisation of conventional deep neural networks gained traction in QAC research due to their inherent capability of generalising well to unseen data and to learn patterns implicitly without the need for manually engineered features [29]. Nonetheless, in recent years, Transformer-based models have gained increased popularity in fields such as natural language processing and computer vision due to their powerful capabilities and versatility [41, 76]. However, despite their superior performance over traditional neural networks, there has been limited research regarding their adoption for QAC. Therefore, this project aims to delve into the potential of exploring the intrinsic capabilities of Transformer-based models to address QAC by generating meaningful completions and tackling the OOV problem. Our findings suggest that while Transformer-based models can effectively provide semantically correct and structured query completions even for unseen prefixes, their performance is far from optimal. To be more specific, these models were found to perform poorly compared to more traditional and non-neural approaches that leverage statistical properties derived from data to generate and rank completions.

1.2 Research Objectives

This project focuses on exploring the potential of large pre-trained Transformer-based language models for the QAC task. To be more specific, and as previously mentioned, Transformer-based models were discovered to outperform traditional neural approaches in several tasks, from question-answering to machine translation [50, 75]. Despite their popularity in several domains, a very limited amount of research has been conducted on their adoption for QAC, possibly due to their computationally expensive and slow nature [36]. Therefore, the primary objective of this project is to investigate whether the pre-trained nature of Transformer-based language models can be leveraged to generate query completions **solely** from an input prefix and to effectively handle the OOV issue associated with the long-tail aspect of search log queries [71]. It is essential to mention that Transformer-based models have already demonstrated successful to QAC related tasks, including query suggestion [49], predicting the following query in session-based contexts [48], code auto-completion [2], and real-time smart email completion [9]. However, these tasks fall outside the specific research scope of this project.

In light of this, in this report, we aim to address the following research questions:

- *RQ1: Can the pre-trained nature of Transformer-based language models be exploited to address the QAC task and provide completions for unseen prefixes?*

- *RQ2: Does the data format affect the models' capabilities and learning behaviour?*

1.3 Report Outline

This report has been structured in a format that should guide the reader through the different key QAC and Transformer-related aspects. More precisely, this report aims to cover the essential concepts required to understand the subject matter. In light of this, the dissertation is structured as follows:

- **Chapter 2 - Background:** This Chapter introduces the topics and definitions required to contextualise the QAC task. This involves a brief summary of studies conducted in the QAC domain, an introduction to Transformer-based models and evaluation metrics required to compare QAC systems.
- **Chapter 3 - Design & Implementation:** This Chapter presents the core components underlying a QAC system. More specifically, this Chapter introduces the QAC pipeline developed as part of this project, including its architecture, requirements and the technologies utilised for its implementation.
- **Chapter 4 - Methodology:** This Chapter fully describes the experimental setup followed in order to promote this study's replicability. This involves information regarding the datasets, data preprocessing steps, the baselines, the models considered and their fine-tuning and decoding processes.
- **Chapter 5 - Evaluation:** This Chapter reports the evaluation results of the Transformer-based models considered for the QAC task with respect to the baselines. This concerns their overall performance and generalisation capabilities. Moreover, QAC systems are evaluated intrinsically by exploring generated query completions. Lastly, the key findings are discussed and tied back to the two research questions mentioned.
- **Chapter 6 - Conclusion:** The Conclusion provides a brief project summary by reinstating the project's goals and key findings. Furthermore, this Chapter outlines this project's limitations and suggests possible directions for future work.

Lastly, it is worth noting that some of the Sections in this report are based and extend upon the Informatics Project Proposal and MSc Progress Report coursework.[1]

[1] Note that Sections 1.1 and 1.2 in Chapter 1, Sections 2.1.1, 2.1.2, 2.1.3, 2.3 in Chapter 2 and Sections 4.1.1, 4.1.2 in Chapter 4 in this report, are based and extend upon the Informatics Project Proposal [67] and the MSc Progress Report Coursework [68].

Chapter 2

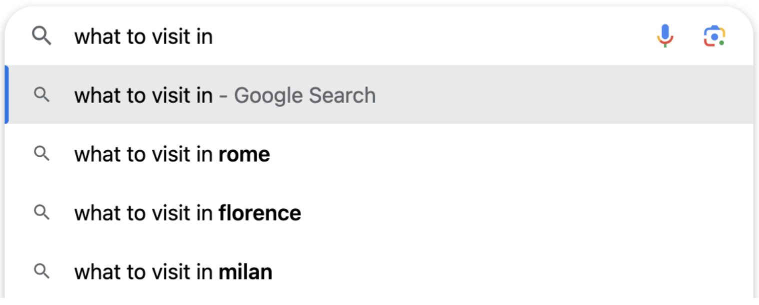
Background

2.1 Query Auto-Completion

2.1.1 Problem Definition

The primary function of Query Auto-completion is to provide users with a ranked list of possible query completions given an input prefix [72]. In other words, QAC systems aim to help users express their search requirements as a free-text query to identify the ideal query that most accurately matches their information need [30].

For instance, and as shown in Figure 2.1, given the query prefix *"what to visit in"*, plausible completions could be *"Rome"*, *"Florence"* or *"Milan"*.



what to visit in

what to visit in - Google Search

what to visit in **rome**

what to visit in **florence**

what to visit in **milan**

Figure 2.1: Query completion provided by Google.com using an example prefix.

In light of this, the QAC problem can be mathematically expressed as follows: for a partial input prefix p of arbitrary length, a QAC system returns a ranked set of query completions $\hat{R}(p)$, obtained from a pool of query candidates $C(p)$ by reducing a certain loss function L, measuring the disparity between the generated query candidates $R(p)$ and the ideal target query q [8]. The full QAC equation can be expressed as follows [8]:

$$\hat{R}(p) = \min_{R(p) \subset C(p)} L(q, R(p))$$

However, despite the successful adoption of reinforcement learning and hierarchical-based approaches to the QAC problem, in the context of this project, we focus primarily on two main classes of QAC methods: *Traditional* and *Learning-Based* QAC methodologies [45, 78, 79, 80].

2.1.2 Traditional QAC Approaches

Traditional QAC approaches can be categorised as those leveraging specific data structures for query completions generation [78]. To elaborate further, Traditional QAC methods construct data structures, such as prefix tries, and lookup-tables, directly from search logs data [25]. Consequently, extracting query candidates from an input prefix becomes a straightforward and efficient lookup operation executed directly on the custom-built data structure [25]. Moreover, candidate completions are often ordered based on statistical properties and metadata derived directly from the search logs [7].

Taking this into consideration, *Most Popular Completion (MPC)* is a widely adopted approach which often offers a strong baseline when comparing QAC systems [16]. More precisely, MPC relies on a prefix trie data structure to extract query completions that match the given prefix and subsequently rank them according to their frequency in the search logs [16]. Therefore, MPC works on the premise that popular and frequent queries are often viable prefix completions [5]. Similarly to MPC, other approaches adopt data structures for the query candidates generation process but rely on different features for the ranking component. Such features include temporal factors as well as user-specific statistics (i.e. location) to provide timely and personalised completions [28, 65]. Nevertheless, a significant drawback of Traditional QAC approaches lies in their inability to provide completions for previously unseen prefixes [78]. This limitation stems from their dependence on knowledge derived solely from the data available in the search logs [78]. Hence, to address the shortcomings, researchers have turned to the Learning-Based approaches outlined in the next Section.

2.1.3 Learning-Based QAC Approaches

Learning-Based QAC methodologies directly overcome the mentioned limitations by learning features for completions generation and ranking directly from data [29]. In this context, the adoption of deep learning models, in conjunction with more traditional approaches, has become predominant in the field to enhance the efficacy and accuracy of QAC systems [29]. For instance, in the work of Mitra and Craswell (2015), query candidates generated from synthetic prefix-suffix pairs are then ranked based on feature vectors learnt with Convolutional Latent Semantic models (CLSM) [47]. Furthermore, in other related studies, Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) networks are utilised as sub-word/character-language models (LM) to generate query candidates and better handle the OOV issue due to the long-tail natures of search logs queries [20, 29, 35, 54, 78]. However, due to the elevated computational cost associated with training word/character-based language models, a recent unnormalised LSTM-based language model was introduced to score query candidates by computing the log probability marginalisation term in an efficient manner [79]. Lastly, a recent non-neural method for QAC, known as QueryBlazer, outperformed neural alternatives, both in accuracy and efficiency, by exploiting a sub-word-level n-gram language model to pre-compute completions prior to runtime [31].

2.1.4 Evaluating a QAC system

When evaluating a QAC system, the metrics vary depending on the task under consideration (i.e. personalised or temporal query completions). As this project focuses on generating ranked completions based solely on a given input prefix, we consider three metrics that capture the overall QAC system's accuracy and behaviour: Mean Reciprocal Rank, Partial-Matching Mean Reciprocal Rank and Success Rate.

2.1.4.1 Mean Reciprocal Rank

Mean Reciprocal Rank ($MRR@k$) is the most widely adopted statistical metric for QAC, defined as the average reciprocal rank (RR) computed at a specific rank k (completions at ranks greater than k are disregarded) [8]. Hence, $MRR@k$ allows to evaluate a system performance based on the rank at which the ground truth completion appears in a ranked pool of candidates [8]. More formally, given a set of prefixes P, MRR can be expressed as follows [8]:

$$MRR = \frac{1}{|P|} \sum_{p \in P} \frac{1}{rank_p}$$

where $rank_p$ identifies the rank at which the prefix-based generated completion matching the ground truth query appears in the ranked set of candidates [8, 54].

2.1.4.2 Partial-Matching Mean Reciprocal Rank

Similarly to MRR, also Partial-Matching Mean Reciprocal Rank ($PMRR@k$) measures the quality of a ranked set of query completions based on the rank at which the ground truth query appears [54]. However, while MRR requires an exact match with the ground truth query, with PMRR, also partial matches between completions and true query are deemed as relevant (i.e. the generated query *order food*, would "partially match" the gold query *order food online*) [54]. In light of this, PMRR can be formalised as follows [54]:

$$PMRR = \frac{1}{|P|} \sum_{p \in P} \frac{1}{partial\ rank_p}$$

2.1.4.3 Success Rate

Success Rate ($SR@k$) can be defined as the average ratio of ground truth queries found within the top k query completion ranks [8]. However, as SR only measures whether the ground truth query appears in the ranked list, but not its rank, the value the metric can assume is always equal or greater than MRR [31].

2.2 An Overview of Transformer Models

In this Section, we aim to provide the background knowledge required in order to contextualise and define the role of Transformer models and their suitability for the QAC task.

2.2.1 From Sequence-To-Sequence to Transformer Models

Sequence-To-Sequence models can be defined as a broad category of models suitable for all those tasks requiring a mapping from a *source* sentence to *target* one (i.e. Machine Translation and Text Summarisation) [52]. For instance, as shown in Figure 2.2, a text translation task requires the conversion from one source language (i.e. English) to a target language (i.e. Italian) [70].

In this context, Sequence-To-Sequence models are often composed of two main building blocks: an Encoder and a Decoder [52, 70]. Encoder and Decoder, often implemented as a stack of Deep Neural Networks (i.e. Recurrent Neural Network (RNN) or LSTMs [64]), are responsible for "encoding" the source sequence as a

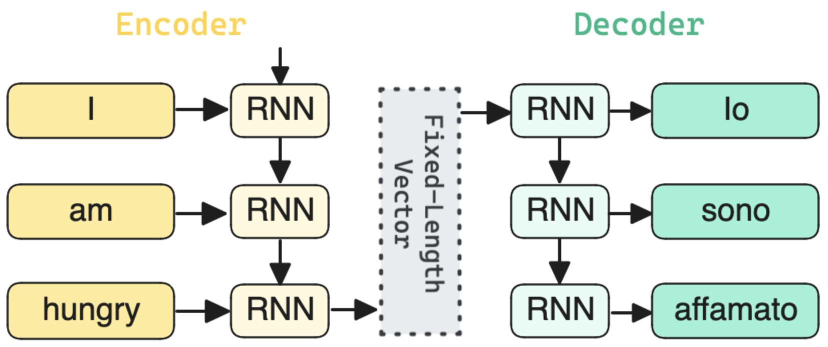


Figure 2.2: Example of a Sequence-To-Sequence Target to Source Translation from English to Italian. The Figure, which has been adapted from [52], schematises a Sequence-to-Sequence model's architecture from a high-level point of view.

condensed hidden representation of fixed-length and then utilise it to sequentially "decode" the next word in a sequence given its probability over the vocabulary [52].

Taking this into consideration, Transformer models, which belong to the broad family of Sequence-to-Sequence models, gained popularity in recent years by achieving State-of-the-Art (SOTA) performance in multiple domains spanning from Computer Vision to Natural Language Processing (NLP) [14, 33]. In this context, the Transformers' architecture stems and extends the Sequence-To-Sequence models' architecture previously mentioned. To be more specific, the original Transformer model introduced in the study conducted by Vaswani et al. (2017) consists of several identical Encoder/Decoder layers in conjunction with newly designed and implemented components such as *Positional Encodings* and *Attention* [76].

As shown in Figure 2.3, Positional Encodings, derived from *sine* and *cosine* functions of different frequencies, are added to the input and output embeddings in order to preserve and enforce the tokens' order within the provided sequences [76]. As a consequence, tokenised sentences such as *"It is sunny today"* and *"It today sunny is"* are treated differently by the model. Furthermore, one of the crucial elements characterising Transformer models is the concept of *Attention*. More precisely, and without focusing on the technicalities, Attention can be defined as a mechanism that allows Transformers to focus selectively on different parts of a sequence to learn their relative importance with respect to the rest of the sequence [76]. In other words, Attention helps models learn connections between tokens and their significance in context [76].

Furthermore, Transformer models leverage *Transfer Learning* to exploit pre-acquired knowledge and apply it to an unrelated problem and domain [59]. Specifically, Trans-

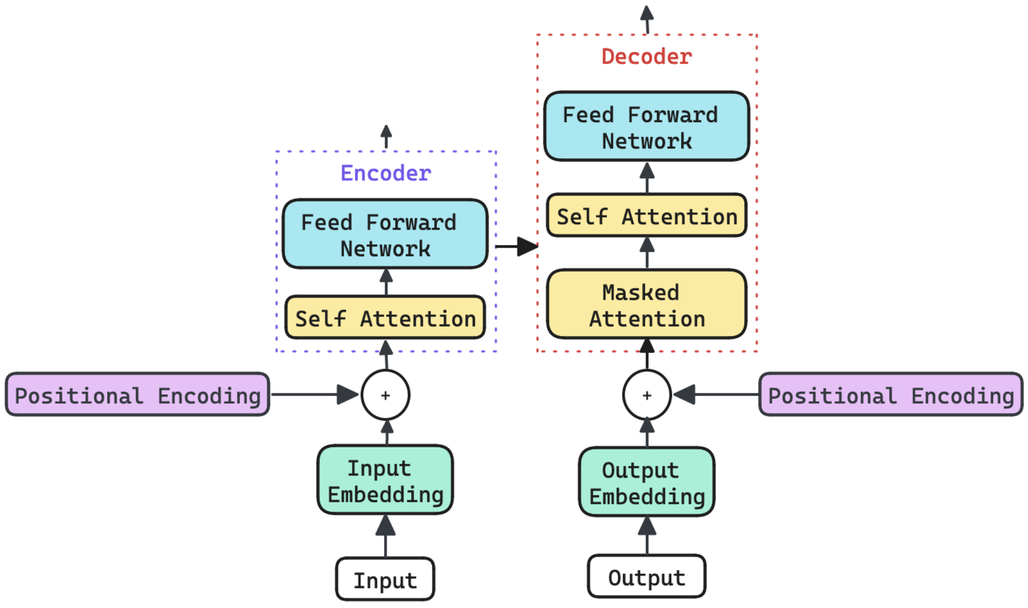


Figure 2.3: Schematic Representation of a Transformer model. The Figure, adapted form [76], simplifies the original Transformer architecture.

former models are often pre-trained on large corpora of diverse documents in order to acquire the knowledge to exploit later when fine-tuned on a downstream task [59]. The aim is to improve the models' versatility and ability to generalise effectively [59].

Consequently, it becomes evident that, within this project's scope, formalising the QAC task as a source-target mapping becomes an intuitive approach where a source prefix can be converted directly to a target query completion. Therefore, multiple Transformer-based models have been considered in order to leverage their inherent capabilities and pre-acquired knowledge for the QAC task.

2.2.2 T5

Text-to-Text Transfer Transformer, also known as T5, is a versatile large language model introduced by the authors of the paper *"Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer"* [59]. The authors proposed the T5 model, which adopts the same architecture shown in Figure 2.3 and acts as a unified framework capable of handling multiple tasks framed as text-to-text objectives [59]. Consequently, T5 is capable of tackling problems such as summarisation and machine translation by simply structuring the model's input according to the task under consideration (i.e. **summarize:** input / **translate English to Italian:** input) [59].

In light of this, the model's great flexibility comes from the fact that it was pre-trained on a diverse set of documents extracted from multiple sources and unified into a sole dataset known as C4 [59]. Hence, as a direct consequence, T5 can effectively apply task-specific knowledge to new domains (Transfer Learning) [59]. Lastly, it is worth noting that sentences provided as input to the model are tokenised as WordPiece tokens using the SentencePiece library [39, 59]. Therefore, the WordPiece algorithm allows processing tokens at a sub-word level, offering more flexibility over character and word level tokenisers [81].

2.2.3 ByT5

ByT5 is a Byte-to-Byte Transformer-based model released by Google Research in 2022 [82]. The model shares the same Encoder/Decoder architecture of T5 and mT5 (Massively Multilingual Pre-trained Text-to-Text Transformer) but introduces minor changes in order to handle Byte-to-Byte input sequences [82, 83]. For instance, with respect to T5 and mT5, ByT5 has a much smaller embedded hidden size of 256 (corresponding to its vocabulary size) compared to the much larger vocabularies for the other two models [82]. Furthermore, apart from having a different pre-training objective, ByT5 has a much denser Encoder (more layers in Encoder than Decoder), similarly to Encoder-only models such as BERT (Bidirectional Encoder Representations from Transformers) [15, 82].

Nonetheless, the main advantage offered by ByT5 is its ability to operate directly on raw sentences on a byte-level, eliminating the need for text-tokenisation and offering a robust token-free model with virtually no modifications to the original T5 architecture [82]. As a direct consequence, ByT5 outperformed models adopting sub-word tokenisation for noisy tasks where spelling, pronunciation, and OOV tokens were crucial components to consider during the training process [82]. Hence, the mentioned capabilities make ByT5 suitable for the task investigated in this project, where noisy and partial prefixes are often provided as input to QAC systems.

2.2.4 GPT-2

Similarly to T5 and ByT5, also the Generative Pretrained Transformer 2 model (GPT-2) has been designed to leverage Transfer Learning in order to address various NLP tasks. More precisely, GPT-2 was pre-trained in an unsupervised manner on several millions of documents and later fine-tuned on a wide variety of tasks in order to improve the model's robustness and generalisation capabilities [58]. Moreover, as for other

GPT models, GPT-2 is auto-regressive, as tokens are generated sequentially based on their likelihood conditioned upon the previous context (previously seen tokens) [18, 57, 58]. However, GPT-2 differs from T5 and ByT5 as it employs a Decoder-Only architecture [58]. Furthermore, GPT-2 uses Byte Pair Encoding (BPE) to represent input sequences [58]. In this context, BPE offers a tokenisation solution between character and word-level algorithms that operates at a byte-level [73]. As a direct consequence, BPE offers a robust solution to multiple languages, and OOV tokens [73]. Lastly, it is worth mentioning that GPT-2 proved to be effective and accurate in a multitude of text generation tasks, including text summarisation and question answering, making it suitable for the QAC task addressed in this report [58].

2.2.5 Decoding Algorithms

In order to generate an output sequence given an input as prompt, generative models such as T5 and GPT-2 leverage a softmax function in order to compute a "discrete probability distribution over the vocabulary" [37]. More formally, given a source sequence $X = \{x_1, x_2, ..., x_{|X|}\}$ and a target sequence $Y = \{y_1, y_2, ..., y_{|X|}\}$, the goal of a decoding algorithm is to find the sequence Y which maximises the conditional probability $P(Y|X)$ [89]. Hence, in order to achieve so, several decoding algorithms such as *greedy/beam search* and *top-k sampling* are often adopted in order to produce an output from the derived probability distribution [37].

2.2.5.1 Greedy Search

Computing $P(Y|X)$ is an intractable problem, as there are infinite possible sequences Y for which such conditional probability would have to be computed [10]. In this context, greedy search offers a simple solution to decode output sequences in a computationally efficient manner. More precisely, at each time step, greedy search selects the token with the highest probability conditioned on the given input sequence and output tokens generated so far [10]. Hence, while greedy search offers an efficient solution, outputs generated are rarely optimal [10]. More precisely, the algorithm operated shortsightedly as tokens are selected greedily by disregarding possible future high probability tokens [10].

2.2.5.2 Beam Search

Beam search offers a more robust solution between greedy search and exhaustive search [26]. To be more specific, beam search keeps track of the top k most likely hypotheses (token sequences generated so far) [10]. Furthermore, at each time step, the algorithm selects the next most likely tokens conditioned on the current hypothesis under consideration [10]. Hence, the algorithm repeats the process until the termination criterion is met, and the top output candidate (the one that maximises the overall conditional probability) is returned [89]. However, it is worth noting that also beam search leads to a globally sub-optimal solution which often deteriorates as k increases [21].

2.2.5.3 Top-k Sampling

Another popular approach often adopted in order to decode and generate output sentences is *top-k sampling* [17]. In top-k sampling, instead of selecting the successive token based on the probability distribution over the entire vocabulary, the next token is sampled only among the top-k most likely tokens in the "truncated" distribution [24]. However, while top-k sampling often leads to better generations than beam search, selecting an appropriate k value becomes challenging, with low k values leading to repeated outputs (just a few k tokens considered at each step) and high k values producing out-of-context and degraded generations [24]. In order to address some of the limitations mentioned, an additional parameter known as *temperature* can be introduced in order to adjust the probability distribution prior top-k sampling [24]. In this context, temperature provides control over the diversity of the generated output, with low values leading to more deterministic sequences and high values increasing the output randomness [6].

2.2.6 Evaluating Transformers Output

Evaluating Language Models is crucial to assessing models' performance and accuracy. In this context, models fine-tuned on a specific downstream task (i.e. question-answering) are often evaluated using task-specific metrics (i.e. Accuracy, F1-Score) [51]. However, when ground-truth target sentences are available, models' generations are evaluated with respect to the gold references by computing metrics such as Bilingual Evaluation Understudy (BLEU), Metric for Evaluation of Translation with Explicit Ordering (METEOR) and BertScore [3, 53, 88]. In the context of this project, we consider BLEU solely, despite its known limitations, due to its wide popularity in the research community [3].

2.2.6.1 Bilingual Evaluation Understudy

Bilingual Evaluation Understudy (BLEU) is a computationally effective and language-independent evaluation metric proposed initially to evaluate generations for machine translation tasks but later adopted in various NLP domains [53, 60]. More precisely, BLEU allows to measure the average phrase overlap between a model's generations and gold references in order to quantify their "translation closeness" [53]. Therefore, the BLEU score, in the range $[0,1]$ (where a value of 1 represents a perfect match between generation and gold reference), provides a measure of similarity that strongly corresponds to human evaluations [53].

2.3 QAC Systems & Transformers Limitations

As highlighted in this Chapter, Transformers are very powerful and versatile models that leverage mechanisms such as Attention and Transfer Learning in order to generate coherent and semantically correct sentences. However, training Transformers is computationally and resource intensive as Transformers often require many iterations over large quantities of data in order to achieve satisfactory performance [41]. In this context, the little research conducted on Transformers for QAC could be explained by the scarcity of high-quality datasets accessible to the general public due to the sensitive nature of search logs data [62]. Furthermore, the application of QAC systems requires low-latency solutions, which are unsuitable for the relatively slow inference capabilities of Transformer models [22]. Nonetheless, QAC systems could still leverage Transformers' generalisation capabilities to address the OOV issue by framing the QAC task as a sequence-to-sequence problem.

Chapter 3

Design & Implementation

This Chapter aims to introduce the core components underlying a QAC pipeline. More precisely, this Chapter starts by summarising the QAC task, its core units and the requirements a QAC system must satisfy. Moreover, it presents how the QAC pipeline has been designed and implemented for this project as separate modules. Lastly, the technologies utilised are briefly summarised to contextualise the computational costs and resources employed.

3.1 The QAC Task

3.1.1 Formalising the QAC Problem

As mentioned in Section 2.1, the goal of a QAC system is to produce a ranked list of candidate completions given an input prefix. Hence, a complete QAC pipeline requires multiple sequential steps in order to produce query completions.

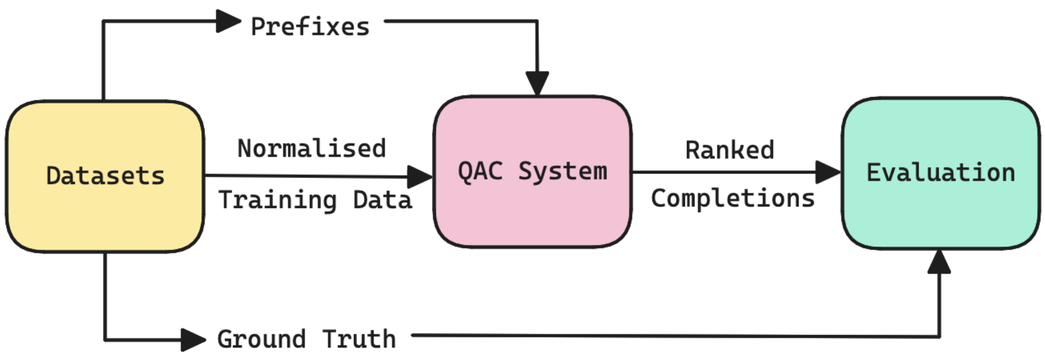


Figure 3.1: General Overiview of a QAC Pipeline.

To be more specific, and as shown in Figure 3.1, a QAC pipeline often comprises three main components: one or more datasets, a QAC system and an evaluation module. Firstly, queries in the search logs (dataset), are normalised in order to clean and remove noisy data points. Secondly, normalised queries, and often additional metadata, are fed to a QAC System as training data to learn patterns and infer dataset-specific statistical properties. Furthermore, prefixes extracted from the test data are provided as input to the QAC system in order to produce a ranked list of query completions. Lastly, ranked completions are evaluated with respect to the gold queries (ground truth) in order to compute evaluation metrics (i.e. MRR) and assess the system's performance.

3.1.2 QAC Pipeline Requirements

Considering the overall pipeline introduced in the previous Section and considering the project's goal, we can list the requirements a QAC pipeline must satisfy as follows:

- The pipeline should support multiple datasets. This involves formatting, splitting and normalising data in a standard and reproducible manner.
- The pipeline should expose functionalities to extract prefixes and ground truth directly from search logs queries (according to a predefined criterion).
- The QAC system should expose the completion candidates' generation and ranking processes. Also, input and output files should be standardised in order to easily extend the system.
- It should be possible to evaluate the QAC system directly from ranked completions and gold queries only.
- Several evaluation metrics should be available, and new ones quickly added as required.

3.2 Core Modules Design

In this Section, we present in more detail the core modules designed and implemented as part of this project. The goal is to provide the reader with a more detailed and project-specific overview of the main components and their inter-dependencies. In light of this, the QAC/experimental pipeline is structured as three separate core modules, which are outlined below and shown in Figure 3.2.

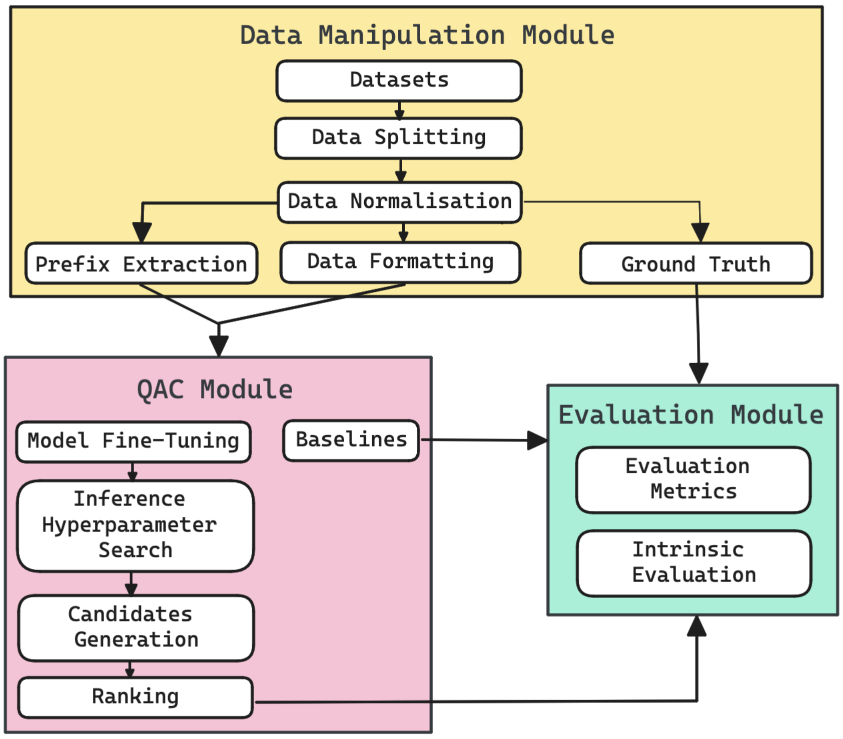


Figure 3.2: The Figure shows the detailed QAC pipeline and its core modules and units. The diagram also shows how data flows from one component to another and closely follows the methodology followed in this project and presented in Chapter 4. Also, note that inspiration has been taken from [66] for the Figure design and overall structure only.

- **Data Manipulation Module:** The data manipulation module offers all the functionalities required in order to convert and standardise the search logs datasets for training models and evaluating QAC systems. To be more specific, queries in the dataset under consideration are first split into train, validation and test partitions and successively normalised by adopting standard normalisation steps as described in related studies [31]. Furthermore, as part of this normalisation step, prefixes and gold queries are extracted from the test set for subsequent completions generation and evaluation. Lastly, normalised queries are then formatted as required in order to be compatible with the model/approach under consideration (i.e. baselines/Transformer-based QAC systems).

- **QAC Module:** The QAC module provides all the tools required to run a complete QAC system. More precisely, the module offers all the scripts for configuring and fine-tuning Transformer-based models, identifying the best decoding parameters and generating and ranking candidate completions. The module also incorporates the code required to replicate the baselines results considered in this project.

- **Evaluation Module:** Lastly, the evaluation module exposes all the metrics required to compare QAC systems (i.e. MRR, PMRR, SR). In this context, the evaluation module also provides functionalities to plot training and validation loss graphs, scripts to evaluate query completions intrinsically and functions to perform statistical hypothesis testing. Hence, this module aims to provide all the tools required to explore QAC systems' behaviour and understand where they fail and succeed.

Lastly, it is worth noting that the QAC pipeline has been designed in a modular and easily configurable manner. To be more specific, the three core modules have been implemented by unifying and standardising inputs, outputs and parameter configurations. Therefore, as a direct consequence, the current system implementation can be easily extended with additional datasets, models and evaluation metrics as long as input and output file formats are kept consistent.

3.3 Software & Hardware

The QAC pipeline described in the previous Section has been implemented using Python and PyTorch as the primary programming language and deep learning framework of choice [56, 69]. This decision's rationale lies in PyTorch's great flexibility, modularity and scalability, with support for sequential and parallel models' training and decoding [56]. In this context, all the experiments presented in this report have been conducted on an NVIDIA A100 SXM4 80GB and a Quadro RTX 6000 GPUs.

Chapter 4

Methodology

This Chapter presents the methodology and experimental setup followed in this project. This includes information about the datasets, data preprocessing steps, Transformer-based models fine-tuning, query completions generation and ranking. The goal is to introduce and define the main experimental pipeline in enough detail to facilitate the reproducibility of the experiments and results.

4.1 Datasets & Data Preprocessing

This Section presents the datasets used for the experiments, including steps such as data normalisation and formatting required to fine-tune Transformer-based models.

4.1.1 Datasets

4.1.1.1 AOL Search Logs

The America Online (AOL) search logs dataset is one of the most widely adopted datasets by the research community to explore tasks such as query auto-completion, session-based search and search personalisation [43, 55]. The dataset includes queries issued on the AOL platform from the 1st of March 2006 to the 31st of May 2006 [32]. The dataset consists of approximately 20 million queries (10 million unique) issued by 650,000 users during the mentioned period [55, 65]. Furthermore, each entry in the dataset comprises of an anonymised user id, the free-text query issued on the search engine, the query submission timestamp, the domain section of the URL clicked as a result of the search action and its corresponding rank (within the list of results retrieved) [55].

However, it is worth noting that despite the AOL dataset's popularity, there is much disagreement in the research community regarding its use and application [4]. Firstly, there is a privacy concern regarding using AOL data due to the limited and inadequate anonymisation process applied to user data [23]. Secondly, AOL logs have not been filtered or cleaned and therefore include sensitive categories such as porn [27]. Lastly, the AOL dataset is often deemed to be outdated due to the time period when queries were collected (i.e. 2006) [90]. Despite the mentioned limitations, AOL is still the primary dataset used for research in this field [72].

4.1.1.2 ORCAS Search Logs

The Open Resource for Click Analysis in Search (ORCAS) dataset is a click logs dataset released as part of the TREC Deep Learning Track in 2020 [11]. The dataset comprises 18 million queries (10 million unique queries) extracted from a subset of Bing's search logs gathered during a period of twenty-six months until January 2020 [11]. In this context, it is worth noting that, conversely to AOL data, ORCAS search logs have been aggregated and strictly filtered in order to avoid leaking users' personal information [11]. For instance, *k-anonymity filter* was applied to ensure that only queries issued by at least *k* users were present in the dataset to avoid personal information from being easily inferred [11].

In light of this, the metadata associated with ORCAS search logs includes query IDs, free-text queries, document IDs corresponding to the TREC Corpus and clicked URLs (without their ranking) [11].

4.1.1.3 Data Augmentation with Wikipedia Anchors

As a preliminary study, we explored the direction of augmenting both the AOL and ORCAS datasets by enriching them with anchor text extracted from Wikipedia [46]. To be more specific, approximately 114 million anchors (with duplicates) were extracted from 23 GBs of articles obtained from a publicly available Wikipedia dump [1]. The assumption underlying this choice was based on the fact that anchors text (text pointing to other Wikipedia pages) could be treated as queries and consequently enrich the datasets [13]. However, preliminary experiments on the augmented datasets revealed a degraded performance for all the baselines considered, possibly due to the difference in the nature of anchors text and search log queries. Due to this fact, this approach is only

[1]https://dumps.wikimedia.org/enwiki/latest/enwiki-latest-pages-articles.xml.bz2

mentioned in this Section and explored no further.

4.1.2 Data Normalisation & Splitting

As aforementioned, this project aims to investigate the role of Transformer-based models for QAC in a modular and reproducible manner. Due to this fact, we adopt the same data normalisation and splitting steps applied by several other studies in the field [31, 35]. Therefore, we applied the following normalisation steps to both AOL and ORCAS queries [31, 35]:

- Case folding.
- Removed non-ASCII characters, preserved only a subset of punctuation symbols and performed Unicode NFKC normalisation
- Replaced multiple spaces with single spaces and removed spaces at the beginning and end of queries.
- Removed multiple queries issued by the same user within a very short time frame.
- Filtered out queries with less than three characters.

Queries from both datasets are then split in a standard manner according to their timestamp, for AOL, and randomly for ORCAS [31, 35]. The splits with the corresponding numbers of queries are shown in the Table below:

	Train	**Valid**	**Test**
AOL	17,521,031	1,521,971	1,317,632
ORCAS	15,011,856	1,876,480	1,876,480

Table 4.1: AOL and ORCAS Dataset Splits.

Finally, it is essential to highlight that while both AOL and ORCAS datasets provide supplementary metadata, such as click-through data, we solely utilise raw queries for training and evaluating models in this project. To be more specific, as our primary objective is to generate completions directly from prefixes, we disregard personalised and session-based metadata.

4.1.3 Prefix Extraction

Before delving into specific data formats required when fine-tuning Transformer-based models for the QAC task, it is essential to define how prefixes have been extracted from AOL and ORCAS queries in order to fine-tune, validate and test models. More precisely, prefixes had to be extracted from the train/valid/test partitions (shown in Table 4.1) in order to construct the QAC-specific training set as well as to generate test prefixes required to run inference and evaluate QAC systems. In light of this, for each query q of length l_q, prefixes have been extracted by randomly selecting a prefix length value l_p, in the range $2 \leq l_p \leq l_q - 1$ as per similar studies in the field [31].

4.1.4 Data Formatting for Transformer-based Models

As mentioned in Section 2.2, one of the main advantages of Transformer-based models is their ability to leverage transfer learning in order to tackle cross-domain problems by formalising tasks as a sequence-to-sequence objective. In this context, the QAC task addressed in this report can also be framed as a sequence-to-sequence problem, where a source sequence (query prefix) is mapped to a target sequence (query completion). However, it is worth noting that as one of our aims is to explore whether different data formats affect how Transformers learn and decode outputs, we investigate two possible conditions for source-target mappings:

1. The first data format treats the sequence-to-sequence problem as a mapping from source (prefix) to target (full query completion) (i.e. *what to vis* $\rightarrow$ *what to visit in Paris*). From now on, we denote this mapping as *prefix* $\rightarrow$ *query*.
2. By contrast, the second data format considered frames the task as a mapping from query prefix to suffix (i.e. *what to vis* $\rightarrow$ *it in Paris*) and denoted as *prefix* $\rightarrow$ *suffix*.

Nonetheless, framing the QAC task as a sequence-to-sequence problem is insufficient, as different Transformer-based models require diverse input formats in order to learn and produce coherent and semantically correct outputs. Hence, in the next two Sections (4.1.4.1 & 4.1.4.2), we outline how queries have been formatted in order to be consumed by T5, ByT5 and GPT-2.

4.1.4.1 T5 & ByT5

The learning objective of both T5 and ByT5 is to generate a target sequence given an input sequence [59, 82]. Therefore, when fine-tuning the models, source-target pairs are provided as input to the models directly as textual data [59].

However, in the context of T5, and as a result of preliminary experiments, additional task-specific strings have been appended at the beginning of both source and target sentences. To be more specific, this templated addition aims to aid the model in learning the new QAC task and discriminate more effectively with respect to the prior learnt tasks (i.e. summarisation). In light of this, source and target sequences have been formatted as follows:

- **Source:** `prefix:prefix` (i.e. *prefix:how to build*)
- **Target:** `completion:full_query` (i.e. *completion:how to build a house*)

Nonetheless, it is worth noting that a slightly different format had to be used when fine-tuning on the second data format presented in Section 4.1.4 (*prefix* $\rightarrow$ *suffix*). More precisely, after several experimental runs, it was noted that models struggled to generate completions for prefixes terminating mid-word. To elaborate further, models could not identify whether the target was a continuation of the prefix or if the generation had to produce a completely new word. As a direct consequence, models were inconsistent in generating spaces at the beginning of the target sequence, making it impossible to extract the full completion. For instance, for the source prefix *Lond*, a generated target suffix might have included a space as the first character producing the incorrect completion *Lond on*.

Therefore, in order to address this issue, a new token (`<|space|>`) was introduced at the beginning of target sequences, but **only** under two conditions: when the source prefix terminated with whitespace or when the target sequence started with one. The underlying assumption was that, with the addition of this supplementary token, models would have learnt to generate the `space` token when required (i.e. suffix starts with a new word), making it possible to reconstruct completions during inference.

Hence, for a source terminating with whitespace, the formatting would have been the following:

- **Source:** `prefix:prefix` (i.e. *prefix:how to build*)
- **Target:** `completion:<|space|>suffix` (i.e. *completion:*`<|space|>`*a house*)

Lastly, data formatting for ByT5 follows the same structure outlined for T5. However, when constructing source and target sequences, no `prefix:` and `completion:` strings have been prepended as not required when fine-tuning ByT5 models for one unique task (as per documentation[2]).

4.1.4.2 GPT-2

Compared to T5 and ByT5, GPT-2 is auto-regressive in nature [58]. To be more specific, during the decoding/inference phase, the model behaves more like a traditional language model, as it processes sequences left to right and generates the next tokens based on the probability distribution over the vocabulary [18, 57, 58]. Therefore, when fine-tuning GPT-2 for the QAC task, it is impossible to provide training data in the form of two distinct source/target sequences (as done for the other two models) but needs to be fed to the model as a single string. Consequently, to address this issue, training data can be formatted by unifying source and target sequences as a single input capturing all the required task-specific information[3]. Hence, training data has been formatted by introducing three additional tokens, as shown in Figure 4.1.

Fine-Tuning Data Format:

```
<|startoftext|>prefix<|separator|>full_query/suffix<|endoftext|>
```

Inference Data Format:

```
<|startoftext|>prefix<|separator|>
```

Figure 4.1: GPT-2 Training/Inference data formats for *prefix* $\rightarrow$ *query/suffix* pairs.

In this context, the tokens `startoftext` and `endtoftext` are adopted to clearly define the sentence boundaries. By contrast, the `separator` token is instead introduced in order to clearly separate the source from the target, which is particularly crucial at inference time (as shown in Figure 4.1). Lastly, as for T5 and ByT5, an additional `space` token has been introduced when operating with the *prefix* $\rightarrow$ *suffix* data format under the same conditions discussed in Section 4.1.4.1 and as shown in Figure 4.2.

[2]https://huggingface.co/docs/transformers/model_doc/byt5

[3]The idea of unifying source/target sequences for GPT-2 training as a single string as well as the introduction of a new custom separator token was inspired by the following GitHub issue https://github.com/huggingface/transformers/issues/1464 and expanded upon a very similar data formatting methodology outlined in [66] for GPT-2 fine-tuning.

```
<|startoftext|>prefix<|separator|><|space|>suffix<|endoftext|>
```

Figure 4.2: GPT-2 Training data format for *query* $\rightarrow$ *suffix* pairs.

4.2 QAC Module

This Section introduces the core components behind the QAC systems considered in the project. Firstly, baselines and their configurations are briefly outlined. Secondly, details regarding how Transformer-based models have been fine-tuned are provided for reproducibility. Furthermore, inference settings are described for candidate generation from test prefixes. Lastly, the ranking algorithm adopted to rank the generated candidates is briefly summarised.

4.2.1 Baselines

In the context of this project, two main baselines have been considered: Most Popular Completion (MPC) [16] and QueryBlazer [31]. The primary motive behind this decision is that both baselines generate query completions directly and solely from raw queries and do not rely on additional metadata (i.e. click-through rate). Due to this fact, we can effectively compare the approaches evaluated in this project. Moreover, despite the two baselines being non-neural, they provide strong baselines that outperform neural approaches for the same task [31].

In light of this, the MPC baseline has been implemented by adapting code released as part of the study conducted by Kim & Gyuwan (2019), which offers all the functionalities required to build a prefix-trie from raw training queries as well as to extract and rank completions given test prefixes [35]. Similarly, to evaluate and replicate QueryBlazer results on both AOL and ORCAS datasets, we modified the code published alongside the research paper by Kang et al. (2021) [31]. In this context, the code provided allows to easily extract the vocabulary by utilising SentencePiece with BPE tokenisation, building a Finite State Transducer Encoder, and training an n-gram traditional language model on raw queries [31]. Lastly, both MPC and QueryBlazer have been evaluated by using the same parameter configurations presented in the study by Kang et al. (2021) [31]. To be more specific, two QueryBlazer configurations have been considered, with vocabulary sizes equal to 256 and 4096 (denoted QueryBlazer-256 and QueryBlazer-4096), n-gram size equal to 5 and count cutoff set as follows [00000].

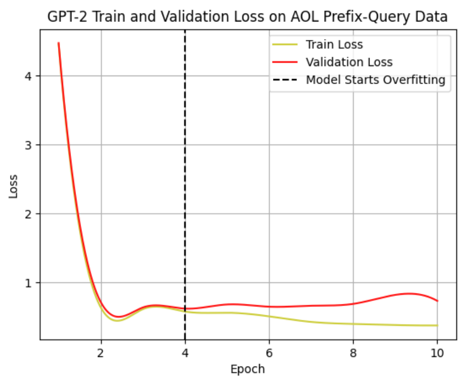
4.2.2 Transformer-based Models Fine-Tuning

The QAC module incorporates all the scripts required to fine-tune Transformer-based models in a structured and defined manner. This includes all the functionalities utilised to process the training data in the format described in Section 4.1.4 as well as utilities to fine-tune models and monitor the training and validation losses.

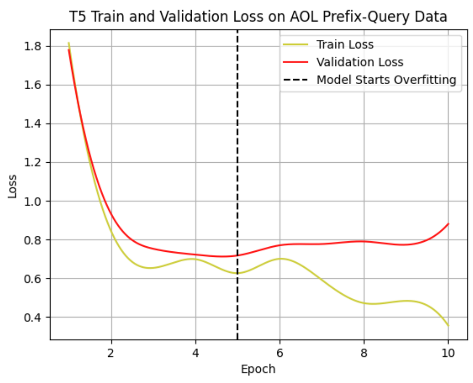
As will be described in Chapter 5, the three Transformer-based models considered have been fine-tuned on the two data formats previously presented (*prefix* $\rightarrow$ *query/suffix*). In this context, is it worth noting that, for both AOL and ORCAS datasets, the entire training set, including duplicated queries, has been utilised to fine-tune models. To elaborate further, while it might be argued that removing duplicates from the training data might improve language models' performance [40], in the context of this project, we operate under the assumption that preserving duplicates is more suitable for the QAC task. In other words, preserving query duplicates keeps the underlying distribution of queries intact, leading models to be slightly biased towards more popular queries, which might benefit the QAC system's overall performance.

In light of this, three Transformer-based models have been considered for the QAC task: **T5-Small** (60 million parameters), **ByT5-Small** (300 million parameters) and **GPT-2-Small** (117 million parameters).

In this context, as fine-tuning Transformer-based models is an expensive and time-consuming task, we quickly prototyped by fine-tuning models on a subset of one million documents randomly sampled from the AOL training set (on the *prefix* $\rightarrow$ *query* data format) and for ten epochs. The primary rationale behind this standard approach was to understand the models' behaviour and identify the minimal number of epochs required



(a) GPT-2-Small



(b) T5-Small

Figure 4.3: GPT-2 and T5 Train/Validation losses on the AOL *Prefix* $\rightarrow$ *Query* Format.

for the models to specialise in the QAC task without overfitting [1].

Consequently, and as shown in Figure 4.3, for both GPT-2 and T5, the training and validation losses rapidly decrease after the first two epochs and stabilise after the third one. By contrast, while GPT-2 clearly starts overfitting during the fourth epoch (4.3a - increase in validation loss and decrease in training loss), T5 overfits during the fifth epoch. However, as shown in Figure 4.3b, training and validation losses start diverging noticeably towards the end of the fourth epoch, indicating potential overfitting. Therefore, as a direct consequence of this preliminary analysis, and despite the models' size differences, we decided to fine-tune them by adopting the same training hyper-parameters to avoid introducing additional variability during the training process. Hence, models have been fine-tuned for three *epochs*, with *Adam with decoupled weight decay (AdamW) Optimiser* [42] and *learning rate* of 5×10^{-4}, *warmup steps* set to 5000, and *epsilon* value equal to 1×10^{-8}. Moreover, we used a *batch size* of 64, and input sentences were truncated to 32 tokens as the average number of tokens per input was found to be approximately 16 tokens. Lastly, it is essential to state that we are aware of the limitations of selecting hyper-parameters based solely on the models' performance on a subset of the data, as models' effectiveness could drastically vary due to their difference in size and when trained on the full dataset. Nonetheless, this decision was dictated due to time and computational constraints, and we leave hyper-parameters tuning as possible directions for future work.

4.2.3 Inference & Hyper-parameters Search

As mentioned in Section 2.2.5, different decoding algorithms produce diverse outputs, with some producing more coherent and deterministic generations and others decoding more random and variable sequences. In the context of QAC, and due to the nature of the task, we need to consider a decoding algorithm that allows to generate n diverse completions given an input prefix p. Therefore, as greedy search was found to produce mostly repetitive and less-diverse outputs, we focused solely on beam search and top-k sampling to generate completions [86]. However, similarly to fine-tuning Transformer-based models, also inference is a costly task [77]. Hence, conducting grid search, a standard approach adopted in multiple studies for optimal hyper-parameters identification, on the entirety of the validation set becomes infeasible [38, 44, 66, 84]. Consequently, in order to address the shortcomings, we explored the hyper-parameters space by conducting inference (beam search & top-k sampling) on a subset of 1000

		Temperature									
		0.2		**0.4**		**0.6**		**0.8**		**1.0**	
		Q	**S**	**Q**	**S**	**Q**	**S**	**Q**	**S**	**Q**	**S**
	20	0.547	0.547	0.544	**0.549**	0.541	0.536	0.530	0.531	0.520	0.515
	40	0.551	0.546	0.548	0.548	0.539	0.544	0.530	0.533	0.514	0.515
Top-k	**60**	0.548	0.548	0.544	0.546	0.543	0.541	0.530	0.527	0.511	0.517
	80	**0.552**	0.548	0.546	0.545	0.539	0.541	0.531	0.523	**0.508**	**0.511**
	100	0.548	0.547	0.546	0.545	0.542	0.536	0.529	0.526	0.512	0.515

Table 4.2: GPT-2-Small Top-k Sampling Hyper-parameters Search on a subset of 1000 prefixes from the AOL validation set. Results report the average BLEU score computed by generating one candidate completion per prefix. Scores are shown for models trained on the *Prefix* $\rightarrow$ *Query* (denoted as **Q**) and *Prefix* $\rightarrow$ *Suffix* (denoted as **S**) Data formats.

prefixes randomly extracted from the AOL validation set as such approach was found to be effective if performed on a representative subsets of data [34, 77]. More precisely, for each model, decoding algorithm, and parameter configuration, the average BLEU score has been computed over the set of sequences generated (one per prefix) with respect to the gold query. In this context, the parameters explored for beam search are *length penalty* and *number of beams* while for top-k sampling, we considered *top-k* and *temperature*. Taking this into consideration, beam search and top-k sampling results for GPT-2-Small are reported in Tables 4.2 and 4.3 respectively. Results for T5-Small and ByT5-Small are shown in the Appendix in Sections A.1 and A.2.

In light of this, as it is possible to see in Table 4.2, the *top-k* parameter does not impact much on the BLEU scores for both the data formats considered (lowest scores shown in red and highest scores shown in green). By contrast, *temperature* directly affects the BLEU results, with low temperature producing higher average scores and high temperature rapidly degrading the overall performance. This could be because high-temperature values often lead to more random outputs, which are more likely to diverge from the gold query [6]. However, the average BLEU scores obtained with top-k sampling are lower overall than those computed by beam search. More precisely, as shown in Table 4.3, a lower number of beams coupled with a low length penalty hurts the model's performance. By contrast, a higher number of beams in combination with a higher length penalty leads to slightly better generations. In this context, *length penalty* is a hyper-parameter that affects the generations' length with higher values promoting longer outputs [81]. Nonetheless, it is clear that results provided in Table 4.3 are only

		Length Penalty											
		0.0		**0.2**		**0.4**		**0.6**		**0.8**		**1.0**	
		Q	**S**	**Q**	**S**	**Q**	**S**	**Q**	**S**	**Q**	**S**	**Q**	**S**
# Beams	**10**	**0.550**	**0.549**	0.551	0.552	0.552	0.553	0.553	0.555	0.554	0.557	0.554	**0.558**
	15	**0.550**	**0.549**	0.552	0.551	0.553	0.553	0.553	0.555	0.555	0.557	0.555	**0.558**
	20	0.551	0.549	0.552	0.552	0.554	0.553	0.554	0.556	0.555	0.557	**0.556**	**0.558**
	30	0.551	0.549	0.552	0.551	0.554	0.553	0.554	0.555	0.555	0.557	0.555	**0.558**

Table 4.3: GPT-2-Small Beam Search Hyper-parameters Search on a subset of 1000 prefixes from the AOL validation set. Results report the average BLEU score computed by generating one candidate completion per prefix. Scores are shown for models trained on the *Prefix* $\rightarrow$ *Query* (denoted as **Q**) and *Prefix* $\rightarrow$ *Suffix* (denoted as **S**) Data formats.

marginally better as parameters change. Therefore, as this behaviour was consistent across models, we opted to run inference on the complete test set by selecting a number of beams value equal to 10 and a length penalty equal to 0.0. To be more specific, the marginally worse average BLEU scores obtained by selecting a lower number of beams give the advantage of a faster and more computationally efficient decoding phase [19]. Furthermore, a length penalty value of 0.0 allows to rank candidate generations according to their conditional probability (more details in the next Section), at the cost of promoting shorter generations.

4.2.4 Ranking Query Candidates

One of the core components of a QAC system is the ranking module. To be more precise, once candidate completions have been generated/extracted given a query prefix, they need to be ranked according to a defined criterion (i.e. most probable completions ranked higher). As previously mentioned in Section 2.1, several approaches have been leveraged to rank candidates, such as frequency-based ranking and unnormalised language models [16, 79]. However, as this project aims to explore the capabilities of Transformer-based models for generating QAC candidates, not much focus is invested in exploring ranking methodologies. Nonetheless, as beam search is adopted as decoding algorithm, its scoring function is leveraged in order to rank generation during inference. To be more specific, given an input sequence $\mathbf{x}$, the goal of beam search is to produce an output sequence with the highest conditional probability score for each token y_i conditioned on the input sequence and previous tokens [87]. More formally, the beam search scoring function for the generated sentences can be formalised as follows [87]:

$$\frac{1}{L^{\alpha}} \log P(y_1,...,y_L|\mathbf{x}) = \frac{1}{L^{\alpha}} \sum_{t'=1}^{L} \log P(y_{t'},...,y_{t'-1},\mathbf{x})$$

where L is the decoded sequence length, and α, is the length penalty parameter introduced in Section 4.2.3. It is worth noting that with the current scoring function, candidate scores cannot be directly compared as normalised over different sequence lengths. However, by setting α to be equal to 0.0, candidates are scored according to the following equation [87]:

$$\log P(y_1,...,y_L|\mathbf{x}) = \sum_{t'=1}^{L} \log P(y_{t'},...,y_{t'-1},\mathbf{x})$$

This allows to reason in terms of the log likelihood, and consequently compare and rank candidate generations based on their conditional probability [87].

4.3 Challenges & Limitations

In order to define the experimental pipeline described in this Chapter, several challenges had to be addressed. Firstly, fine-tuning Transformer-based models for the QAC task required exploring multiple data formats. More precisely, in our pilot experiments, the data format profoundly impacted the models' performance. Hence, identifying the suitable input format for each model proved to be crucial to tackle the QAC task. Moreover, training Transformer-based models on large datasets is computationally expensive, sometimes requiring days per epoch. Therefore, fine-tuning models with a subset of queries sampled from the training/validation data allowed to identify possible fine-tuning and inference hyper-parameter configurations rapidly. However, we are aware of the limitations of such an approach. To elaborate further, the parameters found only provide an indication of a possible sub-optimal configuration that might work with Transformer-based models. For instance, identifying the inference parameters by computing the average BLEU score on a random subset is a limiting strategy, as there is no guarantee for such a subset to be representative of the actual data distribution [77]. Furthermore, while setting the length penalty to 0.0 slightly hurts beam search results, it comes with the advantage of providing a candidates' ranking strategy. Consequently, the rationale behind our choices was dictated by the time and resource constraints imposed on this project. Therefore, further exploration with more data and parameter configurations are clearly required directions for future work.

Chapter 5

Evaluation

In this Chapter, we provide results from the QAC experiments conducted with Transformer-based models on the AOL and ORCAS datasets. More precisely, this Chapter aims to explore how Transformer-based models perform with respect to the baselines in terms of their performance and generalisation capabilities. Moreover, examples are provided to outline where different models fail or succeed in producing query completions. Lastly, results are discussed and tied back to the main research questions of this project.

5.1 Transformers for QAC Results

The Transformer-based models considered in this project (T5-Small, ByT5-Small and GPT-2-Small) have been fine-tuned on both the AOL and ORCAS datasets under the two data formats considered, *prefix* $\rightarrow$ *query* and *prefix* $\rightarrow$ *suffix*, and denoted as **PQ** and **PS** respectively from now on (for the exact parameter configuration adopted, refer back to Section 4.2.2). In this context, to evaluate models, ten completions per prefix (extracted from the datasets test sets) have been generated with beam search as decoding algorithms with a *number of beams* equal to 10 and *length penalty* set to 0.0. Consequently, all the evaluation metrics (MRR, PMRR and Success Rate) computed are reported at rank 10.

In light of this, results obtained on the AOL dataset are reported in Table 5.1 while results for the ORCAS dataset are shown in Table 5.2. In this context, and as shown in Table 5.1, it is clear that Transformer-based models effectively address the QAC task by generating query completions for both seen and unseen queries. More specifically, all the Transformer-based models outperform the MPC baseline for unseen queries and all the evaluation metrics. This fact suggests that such models can effectively generalise to

	MRR@10			**PMRR@10**			**SR@10**		
	Seen	**Unseen**	**All**	**Seen**	**Unseen**	**All**	**Seen**	**Unseen**	**All**
MPC	0.608	0.000	0.310	0.651	0.091	0.376	0.764	0.000	0.389
QueryBlazer-256	0.562	0.210	0.389	0.646	**0.417**	0.534	0.723	0.290	0.510
QueryBlazer-4096	**0.612**	**0.217**	**0.418**	**0.670**	0.410	**0.542**	**0.766**	**0.299**	**0.537**
T5-Small-PQ	0.425	0.169	0.299	0.483	0.307	0.397	0.544	0.229	0.389
T5-Small-PS	0.407	0.159	0.284	0.467	0.295	0.382	0.550	0.233	0.395
ByT5-Small-PQ	0.426	0.149	0.290	0.496	*0.338*	*0.418*	0.539	0.200	0.373
ByT5-Small-PS	0.441	0.166	0.306	0.502	0.307	0.406	0.546	0.225	0.389
GPT-2-Small-PQ	*0.463*	*0.176*	*0.322*	*0.513*	0.308	0.413	*0.587*	*0.243*	*0.418*
GPT-2-Small-PS	0.444	0.172	0.311	0.498	0.304	0.403	0.564	0.237	0.404

Table 5.1: QAC Results Reported on the AOL datasets. The best results overall are indicated in bold while the highest scores in italics represent the best results among Transformer-based models. Notice that out of 1,317,632 test queries, 670,810 were seen and 646,822 were unseen.

previously unseen data. Furthermore, it is evident that the data format adopted when training Transformer-based models significantly affects their performance. To elaborate further, both T5-Small-PQ and GPT-2-Small-PQ were found to outperform their respective PS variant. By contrast, ByT5-Small-PS generally performed better than ByT5-Small-PQ, suggesting that its token-free approach is beneficial when operating with partial suffixes as target sequences. Nonetheless, among all Transformer-based models, GPT-2-Small-PQ was found to be the best-performing model across metrics, capable of achieving satisfactory performance with both seen and unseen queries. However, it is worth noting that, although some Transformer-based models have an overall (**all** column in the Table) performance that is as least as good as MPC, both MPC and QueryBlazer-256/4096 offer very strong baselines with QueryBlazer-4096 achieving the best overall performance. More precisely, QueryBlazer-4096 consistently surpasses all the other QAC systems for both seen and unseen queries showing robustness and generalisation capabilities. In this context, it is worth noting that QueryBlazer-4096 MRR performance for seen queries is only marginally better than MPC. Therefore, we conducted a two-tailed paired t-test with significance level $\alpha = 0.05$ to evaluate whether the difference between the two distributions of scores is statistically significant [61]. Results of the two-tailed paired t-test conducted produced a $p < 0.05$ indicating that the difference between QueryBlazer-4096 and MPC for seen queries is statistically significant.

Taking this into consideration, the same findings outlined for AOL apply to the ORCAS dataset as shown in Table 5.2. However, conversely to findings for AOL, ByT5-Small-PQ was found to generally produce better generations than its PS variant. This could be justified because the noisy nature of the AOL dataset requires a model that performs on a more fine-grained level (i.e. suffix level) instead of generating the final query completion in its entirety.

	MRR@10			**PMRR@10**			**SR@10**		
	Seen	**Unseen**	**All**	**Seen**	**Unseen**	**All**	**Seen**	**Unseen**	**All**
MPC	0.402	0.000	0.244	0.549	0.165	0.398	**0.568**	0.000	0.345
QueryBlazer-256	0.335	0.248	0.301	0.579	0.501	0.548	0.462	0.340	0.414
QueryBlazer-4096	0.415	**0.257**	**0.353**	**0.637**	**0.517**	**0.590**	0.565	**0.362**	**0.485**
T5-Small-PQ	0.259	0.202	0.237	0.411	0.359	0.390	0.350	0.275	0.320
T5-Small-PS	0.234	0.184	0.214	0.392	0.343	0.372	0.338	0.270	0.311
ByT5-Small-PQ	**0.441**	0.166	*0.306*	*0.502*	0.307	0.406	*0.546*	0.225	*0.389*
ByT5-Small-PS	0.268	0.203	0.243	0.436	*0.373*	0.411	0.366	0.282	0.333
GPT-2-Small-PQ	0.311	*0.227*	0.278	0.442	*0.373*	*0.415*	0.430	*0.317*	0.386
GPT-2-Small-PS	0.284	0.213	0.256	0.410	0.350	0.386	0.397	0.298	0.360

Table 5.2: Results Reported on the ORCAS datasets. The best results overall are indicated in bold while the highest scores in italics represent the best results among Transformer-based models. Notice that out of 1,876,480 test queries, 1,139,849 were seen and 736,632 were unseen.

5.2 Generalisation Capabilities of QAC Systems

This Section aims to explore the generalisation capabilities of the QAC systems considered. More precisely, QAC systems (baselines and the best-performing Transformer-based models), fine-tuned on the AOL dataset, have been evaluated on the ORCAS test set (and vice versa). The goal is to explore how well diverse QAC systems perform on a partially unrelated test set in order to simulate real-life QAC applications, where prefixes provided as input might diverge from the data utilised during the training phase. In light of this, Table 5.3 presents the results for QAC systems trained on the AOL dataset and evaluated on the ORCAS test set. In this context, it can be noticed that MPC still offers a robust baseline for seen queries. However, its overall performance (**all** columns in the Table) is severely degraded as the split seen/unseen queries in the test set is heavily imbalanced and consequently leading many completions to have a score

	MRR@10			PMRR@10			SR@10		
	Seen	Unseen	All	Seen	Unseen	All	Seen	Unseen	All
MPC	0.418	0.000	0.053	0.507	0.113	0.163	0.579	0.000	0.073
QueryBlazer-4096	**0.450**	**0.210**	**0.241**	**0.576**	**0.445**	**0.462**	**0.600**	**0.288**	**0.327**
T5-Small-PQ	0.261	0.164	0.177	0.365	0.318	0.324	0.359	0.224	0.241
ByT5-Small-PS	0.282	0.155	0.171	0.396	0.324	0.333	0.383	0.213	0.235
GPT-2-Small-PQ	0.297	0.160	0.177	0.393	0.306	0.317	0.412	0.223	0.247

Table 5.3: Results for QAC Systems Trained on the AOL dataset but evaluated on the ORCAS dataset. Out of the 1,876,480 ORCAS test queries, 239,421 were seen and 1,637,060 unseen.

of 0.0. Moreover, Transformer-based models behave similarly as previously observed and described in Section 5.1. To be more precise, GPT-2-Small-PQ demonstrated superior performance compared to both T5-Small-PQ and ByT5-Small-PS. The enhanced generalisation capabilities of GPT-2-Small could be attributed to the higher quality dataset (WebText) used during its pre-training phase [58]. More precisely, WebText comprises diverse documents gathered from various sources without relying on the noisy Common Crawl, as was done to construct C4 and mC4 datasets (used to train T5 and ByT5 models) [59, 82].

Nonetheless, while ByT5-Small-PS was found to perform consistently better than T5-Small-PQ, its performance is generally worse when analysing its generalisation capabilities. Moreover, as for results shown in Tables 5.1 and 5.2, QueryBlazer-4096 achieves the highest performance overall, highlighting that its capabilities of computing segmentation candidates at a sub-word level as weighted finite state transducer helps the system to generalise well even for previously unseen queries [31].

	MRR@10			PMRR@10			SR@10		
	Seen	Unseen	All	Seen	Unseen	All	Seen	Unseen	All
MPC	0.464	0.000	0.161	0.515	0.067	0.223	0.654	0.000	0.227
QueryBlazer-4096	**0.547**	**0.202**	**0.322**	**0.622**	**0.371**	**0.459**	**0.702**	**0.271**	**0.421**
T5-Small-PQ	0.340	0.153	0.218	0.394	0.259	0.306	0.447	0.205	0.289
ByT5-Small-PS	0.354	0.155	0.224	0.409	0.266	0.316	0.444	0.213	0.293
GPT-2-Small-PQ	0.345	0.153	0.220	0.384	0.250	0.296	0.453	0.208	0.293

Table 5.4: Results for QAC Systems Trained on the ORCAS dataset but evaluated on the AOL dataset. Out of the 1,317,632 AOL test queries, 457,966 were seen and 859,666 unseen.

Similarly, as shown in Table 5.4, results obtained by evaluating QAC systems fine-tuned on the ORCAS dataset on the AOL test set share the same key findings mentioned. Therefore, in this case, QueryBlazer-4096 was also the best-performing model for both seen and unseen queries and for all the evaluation metrics considered. However, it is worth mentioning that, in this case, GPT-2-Small-PQ was not found to be the best among Transformer-based models. To be more specific, it is clear from Table 5.4 that ByT5-Small-PS generalises and performs better. These findings align with what was previously observed in Section 5.1 and suggest that ByT5-Small-PS provides a more robust and versatile solution when operating with noisy data (i.e. AOL dataset).

5.3 Prefix Length Impact on QAC

One of the main obstacles that QAC systems face is the inherent lack of context associated with the QAC task [8]. To be more specific, inferring a user intent solely from a very short input prefix is extremely challenging. For instance, for the short and ambiguous input prefix `www.` with gold query `www.pinerplantation.com`, GPT-2-Small-PQ would produce the following ranked completions `[www.google.com, www.yahoo.com, www.myspace.com, www.google]`. In this context, while such completions highlight that the model effectively captured information about the distribution of queries in the dataset, the lack of context impeded the model from generating the user-intended query. This common behaviour, shared by all the QAC systems presented in this report, is shown in Figure 5.1.

The Figure shows the percentage of successful completions at rank one based on the ratio between the length of the prefix issued to the QAC system and the length of the gold query (i.e. ground truth). Therefore, it is clear from Figure 5.1 that QAC systems struggle to handle short prefixes due to the lack of contextual information. Moreover, as the prefix length increases (with respect to the gold query length), so does the QAC systems' accuracy in producing correct completions. Therefore, while this fact highlights the limitations of the current QAC system, it also suggests possible directions for future work, where supplementary contextual information, such as click-through rate and user-based data, might be required to generate meaningful completion.

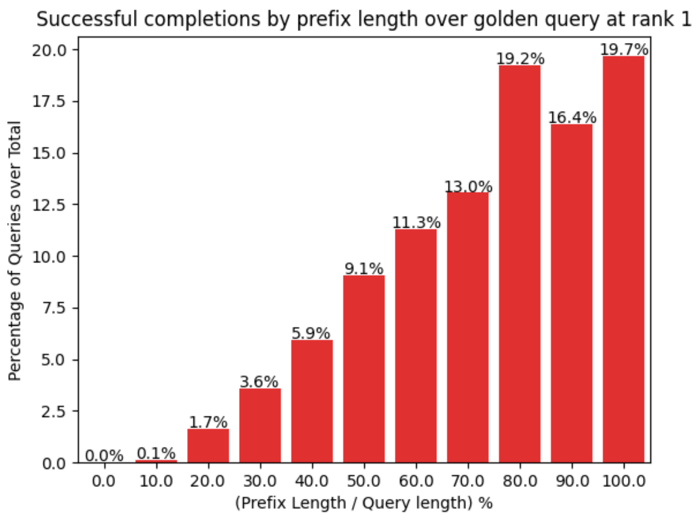


Figure 5.1: The plot shows GPT-2-Small-PQ successful completions at rank 1 based on the ratio of the prefix length over the gold query length.

5.4 Intrinsic Evaluation

This Section aims to intrinsically investigate how different QAC systems succeed or fail in providing successful completions. More precisely, the goal is to explore how QAC systems behave by analysing the completions they produce given an arbitrary input prefix. In light of this, the findings presented in Table 5.5 summarise the generated outputs produced by different QAC systems for the AOL dataset. In this context, the successful completions, where the gold query was accurately generated and present in the ranked output list, are highlighted in green. On the other hand, completions displayed in black indicate a complete failure in producing the gold query as part of the returned results. Moreover, for each gold query, it is reported whether the query had been seen when training the various QAC systems. Finally, it is essential to mention that only completions ranked within the top two positions are reported due to space limitations.

Taking this into consideration, and as shown in Table 5.5, it is clear that MPC fails to generate completions for unseen queries (indicated with a slash symbol). However, this is expected as MPC relies heavily on statistical information derived from the training data. Nevertheless, despite this drawback, MPC successfully generates outputs for seen and infrequent/specific queries (i.e. `prr 2-10-0 4483`) due to its ability to retain information about previously encountered queries. By contrast, QueryBlazer-4096 and the Transformer-based QAC systems successfully produce completions for unseen

Prefix	Golden Query	Seen	Rank	MPC	QueryBlazer-4096	T5-Small-PQ	ByT5-Small-PQ	GPT-2-Small-PQ
housego	housegop.state.il.us	True	1	housegop.state.il.us	housegoogle	housegoods	housegoods.com	housegoogle
			2	housego accountancy uk	housegop.org	housego.com	housegoods	housegoo
prr 2-1	prr 2-10-0 4483	True	1	prr 2-10-0 4483	prr 2-10-2	prr 2-1-1	prr 2-10	prr 2-11
			2	/	prr 2-10	prr 2-106	prr 2-12	prr 2-10
fligim.com	fligim.com	False	1	/	fligim.com	fligim.comhttp	fligim.comhttp	fligim.comhttp
			2	/	fligim.comhttp	fligim.com.	fligim.com.	fligim.com.
zip code mck	zip code mckinney texas	False	1	/	zip code mckinney texas	zip code mckinney tx	zip code mckinney tx	zip code mckinney tx
			2	/	zip code mckinney	zip code mckinney	zip code mckinney	zip code mckinley
www.n.d. doc	www.n.d. doctors.com	False	1	/	www.n.d. doctorates	www.n.d. doctors.com	www.n.d. doc.com	www.n.d. doc.com
			2	/	www.n.d. doctor	www.n.d. doctors	www.n.d. doctors	www.n.d. doctors
pink	pink.com	True	1	pinkworld	pinkworld	pink.com	pinkworld	pinkworld
			2	pink	pink	pinks	pink world	pink eye
davey	davey havoc	True	1	davey havok	davey havok	daveys	davey havoc	davey jones
			2	davey jones	davey jones	davey's	davey havok	davey havok
mozila 20f	mozila 20firefox	False	1	/	mozila 20fashion	mozila 20foods	mozila 20firefox	mozila 20fishing
			2	/	mozila 20florida	mozila 20food	mozila 20fox	mozila 20funds
intel versus a	intel versus amd	False	1	/	intel versus andros	intel versus aol	intel versus aol	intel versus amd
			2	/	intel versus art	intel versus ad	intel versus audio	intel versus athlon
report	report aol	True	1	report	report	reports	reporter	report aol
			2	report spam	report spam	report.com	reports	report cards

Table 5.5: The Table shows query completions generated by different QAC systems trained on the AOL dataset and given an input prefix. Completions highlighted in green identify QAC systems that successfully produce the reference completion (gold query) as the first candidate. By contrast completions in black represent systems that do not return the gold query as part of the output ranked list.

queries. However, while QueryBlazer-4096 generates a correct output for the prefix `fligim.com` (corresponding exactly to the gold query), all the Transformer-based models only partially match the gold query. More specifically, as a direct observation of our empirical studies, it was noticed that Transformer-based models tend to always generate a longer output given an arbitrary input prefix. For instance, for the prefix `fligim.com`, T5-Small-PQ adds a dot at the end of the completion (`fligim.com.`), causing only a partial match with the gold query. In this context, it is also clear from the examples shown in Table 5.5 that T5-Small-PQ has the tendency of adding `.com` directly to the input prefix. This could be a direct consequence of the nature of the AOL dataset, which comprises many URLs and websites.

Considering now ByT5-Small-PQ, it can be noted that the model effectively produces completions for misspelt queries. To be more specific, both the gold queries `davey havoc` and `mozila 20firefox` present spelling mistakes which are effectively parsed and processed by the model, possibly due to its token-free nature. Hence, this fact highlights that ByT5-Small, by operating on a byte-level, can produce semantically correct completions even for ill-formed prefixes. However, it is noteworthy that generations produced still include spelling errors and do not fix the issue. Lastly, considering the prefix `intel versus a`, it is clear that GPT-2-Small-PQ successfully leverages transfer learning to produce semantically correct and conceptually related generations (`intel versus amd`) for previously unseen queries.

5.5 Discussion of Research Outcomes

This Section aims to summarise the main key findings and tie them back to the research questions initially introduced in Chapter 1. To be more specific, the goal of this project was to investigate how Transformer-based models could be utilised for the QAC task and explore how different architectures and data formats would affect the models' behaviour. The underlying assumption was based on the inherent capabilities of pre-trained Transformer-based models to learn semantic patterns and correlation from one task and apply them to a different one [59]. In light of this, and considering the *RQ1 (Can the pre-trained nature of Transformer-based language models be exploited to address the QAC task and provide completions for unseen prefixes?)* it is clear that Transformer models can effectively learn how to tackle the QAC task and address the OOV issue. More precisely, Transformer-based models were found to often outperform the MPC baseline for unseen prefixes and perform better overall. However, the capability of MPC to leverage statistical properties derived from data (i.e. query frequency) often proved to be effective for queries seen during the training phase. Furthermore, considering the performance among Transformer-based QAC systems, GPT-2-Small was found to be the most effective model for the task. To be more precise, the model was capable of generating meaningful completions by handling partial words. This behaviour could be justified with GPT-2's diverse learning objective and the tokenisation algorithm (BPE) adopted [58]. Moreover, GPT-2-Small proved to be effective in handling OOV prefixes and producing semantically coherent generations. This fact could be a direct consequence of the auto-regressive nature of GPT-2, which makes the model focus only on the prefix (rather than the entire sentence as for T5/ByT5) in order to produce complete generations [85]. Therefore, due to this unidirectional aspect, GPT-2 utilises only the prefix as context, which closely aligns with the QAC task.

Similarly to GPT-2-Small, also ByT5-Small was capable of leveraging its token-free nature to produce completions for partial inputs and consequently outperform the other Transformer-based approaches on the ORCAS dataset (shown in Table 5.2) [82].

By contrast, T5-Small struggled to achieve satisfactory performance, suggesting that WordPiece tokenisation might not be suitable for the task [59].

Considering now the *RQ2 (Does the data format affect the models' capabilities and learning behaviour?)*, it is evident that different data formats significantly impact models' performance. To be more specific, formatting data as a mapping from *prefix* to *query* often led to better query completions and, consequently, evaluation results. However, ByT5-Small was found to better handle noisy data formatted as *prefix* $\rightarrow$ *suffix* probably due to its token-free nature [82]. Nonetheless, it is essential to mention that the non-neural QueryBlazer-4096 was found to be the best-performing model for both AOL and ORCAS datasets. This fact highlights that QAC is a challenging task requiring QAC systems to be capable of extracting and combining dataset-specific knowledge to derive completions and generalise well to unseen data.

However, it is worth noting that despite the superior performance of QueryBlazer-4096, the methodology presented in this report has limitations that need to be considered when analysing results. To be more specific, due to time and resource constraints, sub-optimal configurations had to be used in order to explore the role of Transformer-based models for QAC. Therefore, additional experiments should be carried out to identify optimal fine-tuning, decoding and ranking configurations specific to each model.

Chapter 6

Conclusions

6.1 Summary & Takeaways

Query Auto-Completion, has become a crucial feature of search engines which aids users in formalising their search intent as a well-structured and free-text query [30]. However, because of the limited contextual information provided as input by users, inferring the user intent solely from a short prefix is a very challenging task [8]. Furthermore, due to the long-tail nature of search log queries and the Out-of-Vocabulary issue, QAC systems need to be robust and generalise well for previously unseen queries [8, 71]. To address the shortcomings, several neural and non-neural approaches have been researched over the years to learn and derive patterns directly from search logs data.

In light of this, this project aimed to explore whether the inherent capabilities of pre-trained Transformer-based models could be leveraged for the QAC task and address the OOV issue. In this context, three Transformer-based models, GPT-2, T5, and ByT5, have been fine-tuned on two separate search logs datasets, AOL and ORCAS, with the goal of generating query completions solely from an input prefix (i.e. no additional metadata has been used to fine-tune models). Moreover, two diverse data formats have been utilised to explore the impact of data on the models' learning objectives. To be more precise, models have been fine-tuned with the goal of generating full query completions or suffixes given an input prefix.

Our findings suggest that Transformer-based models can effectively learn to generate meaningful completions solely from an input prefix. More precisely, models can process incomplete prefixes and decode completions even for previously unseen queries and consequently address the OOV issue. Moreover, the data format used when fine-tuning models significantly affects their performance, with generating full queries being a more effective learning objective than sole suffixes. In this context, and among models, GPT-2 was found to be the most robust and versatile, showing good overall performance and effective generalisation capabilities. Nonetheless, while Transformer-based models often surpassed the Most Popular Completion baseline in terms of general performance, non-neural approaches, such as QueryBlazer, proved to be the most effective and robust among the QAC systems considered. This suggests that, because of the inner mechanism underlying QueryBlazer, extracting patterns at a sub-word level and specific to a dataset is a more suitable approach for QAC compared to leveraging transfer learning for unseen queries. However, it is worth noting that while this study highlights that Transformer-based models can be employed to address the QAC task, we are aware of several limitations in our experimental setup that might affect the models' performance. Therefore, we outline these limitations in the next Section by pointing them as possible directions for future work.

6.2 Future Work

As aforementioned, the methodology presented in this report has multiple limitations dictated by time and resource constraints. Therefore, multiple additional research directions could be explored to improve and expand upon the current QAC systems. Firstly, for the three Transformer-based models considered (GPT-2, T5 and ByT5), both the fine-tuning and inference phases have been executed by adopting the same hyper-parameter configurations. Hence, while such configurations have been identified by observing the models' behaviour on a subset of the data available, additional experiments could be conducted to identify optimal model-specific parameters based on the entire training set. Secondly, as ranking query generations based on beam search likelihood is sub-optimal, QAC systems might benefit from a more sophisticated ranking function. This might involve training a neural language model directly on the search log queries and then ranking generations according to their sentence probability computed by the language model. Moreover, fine-tuning models on randomly generated prefixes might introduce bias if the distribution of prefixes length over queries length is imbalanced. Consequently, data augmentation and re-balancing techniques could be adopted to enrich the datasets and help models to better capture syntactic and semantic correlations in text. Lastly, training larger models such as BLOOM [63] and LLaMA [74], as well as smaller Transformer-based models, might help capture dataset-specific patterns and provide latency-efficient solutions required for real-world applications of QAC systems.

Bibliography

[1] Saahil Afaq and Smitha Rao. Significance of epochs on training a neural network. *Int. J. Sci. Technol. Res*, 9(06):485–488, 2020.

[2] Gareth Ari Aye, Seohyun Kim, and Hongyu Li. Learning autocompletion from real-world datasets. In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pages 131–139. IEEE, 2021.

[3] Satanjeev Banerjee and Alon Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

[4] Ziv Bar-Yossef and Maxim Gurevich. Mining search engine query logs via suggestion sampling. *Proc. VLDB Endow.*, 1(1):54–65, aug 2008.

[5] Ziv Bar-Yossef and Naama Kraus. Context-sensitive query auto-completion. In *Proceedings of the 20th international conference on World wide web*, pages 107–116, 2011.

[6] Massimo Caccia, Lucas Caccia, William Fedus, Hugo Larochelle, Joelle Pineau, and Laurent Charlin. Language gans falling short. *arXiv preprint arXiv:1811.02549*, 2018.

[7] Fei Cai, Maarten De Rijke, et al. A survey of query auto completion in information retrieval. *Foundations and Trends® in Information Retrieval*, 10(4):273–363, 2016.

[8] Yi Chang and Hongbo Deng. *Query understanding for search engines*. Springer, 2020.

[9] Mia Xu Chen, Benjamin N Lee, Gagan Bansal, Yuan Cao, Shuyuan Zhang, Justin Lu, Jackie Tsay, Yinan Wang, Andrew M Dai, Zhifeng Chen, et al. Gmail smart compose: Real-time assisted writing. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2287–2295, 2019.

[10] Yun Chen, Victor OK Li, Kyunghyun Cho, and Samuel R Bowman. A stable and effective learning strategy for trainable greedy decoding. *arXiv preprint arXiv:1804.07915*, 2018.

[11] Nick Craswell, Daniel Campos, Bhaskar Mitra, Emine Yilmaz, and Bodo Billerbeck. Orcas: 20 million clicked query-document pairs for analyzing search. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 2983–2989, 2020.

[12] W Bruce Croft, Donald Metzler, and Trevor Strohman. *Search engines: Information retrieval in practice*, volume 520. Addison-Wesley Reading, 2010.

[13] Van Dang and Bruce W. Croft. Query reformulation using anchor text. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, page 41–50, New York, NY, USA, 2010. Association for Computing Machinery.

[14] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[16] Giovanni Di Santo, Richard McCreadie, Craig Macdonald, and Iadh Ounis. Comparing approaches for query autocompletion. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, page 775–778, New York, NY, USA, 2015. Association for Computing Machinery.

[17] Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. *arXiv preprint arXiv:1805.04833*, 2018.

[18] Luciano Floridi and Massimo Chiriatti. Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30:681–694, 2020.

[19] Markus Freitag and Yaser Al-Onaizan. Beam search strategies for neural machine translation. *arXiv preprint arXiv:1702.01806*, 2017.

[20] Alex Graves and Alex Graves. Long short-term memory. *Supervised sequence labelling with recurrent neural networks*, pages 37–45, 2012.

[21] James Hargreaves, Andreas Vlachos, and Guy Emerson. Incremental beam manipulation for natural language generation. *arXiv preprint arXiv:2102.02574*, 2021.

[22] Ali Hassani, Steven Walton, Nikhil Shah, Abulikemu Abuduweili, Jiachen Li, and Humphrey Shi. Escaping the big data paradigm with compact transformers. *arXiv preprint arXiv:2104.05704*, 2021.

[23] Daniel Hillyard and Mark Gauen. Issues around the protection or revelation of personal information. *Knowledge, Technology & Policy*, 20:121–124, 2007.

[24] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019.

[25] Bo-June (Paul) Hsu and Giuseppe Ottaviano. Space-efficient data structures for top-k completion. In *Proceedings of the 22nd International Conference on World Wide Web*, WWW '13, page 583–594, New York, NY, USA, 2013. Association for Computing Machinery.

[26] Liang Huang, Kai Zhao, and Mingbo Ma. When to finish? optimal beam search for neural text generation (modulo beam size). *arXiv preprint arXiv:1809.00069*, 2018.

[27] Bernard J. Jansen and Danielle Booth. Classifying web queries by topic and user intent. In *CHI '10 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '10, page 4285–4290, New York, NY, USA, 2010. Association for Computing Machinery.

[28] Danyang Jiang, Honghui Chen, and Fei Cai. Exploiting query's temporal patterns for query autocompletion. *Mathematical Problems in Engineering*, 2017:1–8, 2017.

[29] Danyang Jiang, Wanyu Chen, Fei Cai, and Honghui Chen. Neural attentive personalization model for query auto-completion. In *2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, pages 725–730. IEEE, 2018.

[30] Joemon M Jose, Emine Yilmaz, João Magalhães, Pablo Castells, Nicola Ferro, Mário J Silva, and Flávio Martins. *Advances in Information Retrieval: 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14–17, 2020, Proceedings, Part I*, volume 12035. Springer Nature, 2020.

[31] Young Mo Kang, Wenhao Liu, and Yingbo Zhou. Queryblazer: efficient query autocompletion framework. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pages 1020–1028, 2021.

[32] Rafiullah Khan, Arshad Ahmad, Alhuseen Omar Alsayed, Muhammad Binsawad, Muhammad Arshad Islam, and Mohib Ullah. Qupid attack: machine learning-based privacy quantification mechanism for pir protocols in health-related web search. *Scientific Programming*, 2020, 2020.

[33] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. *ACM Comput. Surv.*, 54(10s), sep 2022.

[34] Krishnateja Killamsetty, Guttu Sai Abhishek, Aakriti Lnu, Ganesh Ramakrishnan, Alexandre Evfimievski, Lucian Popa, and Rishabh Iyer. Automata: Gradient based data subset selection for compute-efficient hyper-parameter tuning. *Advances in Neural Information Processing Systems*, 35:28721–28733, 2022.

[35] Gyuwan Kim. Subword language model for query auto-completion. *arXiv preprint arXiv:1909.00599*, 2019.

[36] Sehoon Kim, Coleman Hooper, Thanakul Wattanawong, Minwoo Kang, Ruohan Yan, Hasan Genc, Grace Dinh, Qijing Huang, Kurt Keutzer, Michael W Mahoney, et al. Full stack optimization of transformer inference: a survey. *arXiv preprint arXiv:2302.14017*, 2023.

[37] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. *arXiv preprint arXiv:2301.10226*, 2023.

[38] Kalpesh Krishna, Yapei Chang, John Wieting, and Mohit Iyyer. Rankgen: Improving text generation with large ranking models. *arXiv preprint arXiv:2205.09726*, 2022.

[39] Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*, 2018.

[40] Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. Deduplicating training data makes language models better. *arXiv preprint arXiv:2107.06499*, 2021.

[41] Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. A survey of transformers. *AI Open*, 2022.

[42] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[43] Sean MacAvaney, Craig Macdonald, and Iadh Ounis. Reproducing personalised session search over the aol query log. In *European Conference on Information Retrieval*, pages 627–640. Springer, 2022.

[44] Luca Massarelli, Fabio Petroni, Aleksandra Piktus, Myle Ott, Tim Rocktäschel, Vassilis Plachouras, Fabrizio Silvestri, and Sebastian Riedel. How decoding strategies affect the verifiability of generated text. *arXiv preprint arXiv:1911.03587*, 2019.

[45] Qiaozhu Mei, Dengyong Zhou, and Kenneth Church. Query suggestion using hitting time. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 469–478, 2008.

[46] David Milne and Ian H. Witten. Learning to link with wikipedia. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, page 509–518, New York, NY, USA, 2008. Association for Computing Machinery.

[47] Bhaskar Mitra and Nick Craswell. Query auto-completion for rare prefixes. In *Proceedings of the 24th ACM international on conference on information and knowledge management*, pages 1755–1758, 2015.

[48] Agnès Mustar, Sylvain Lamprier, and Benjamin Piwowarski. Using bert and bart for query suggestion. In *Joint Conference of the Information Retrieval Communities in Europe*, volume 2621. CEUR-WS. org, 2020.

[49] Agnès Mustar, Sylvain Lamprier, and Benjamin Piwowarski. On the study of transformers for query suggestion. *ACM Transactions on Information Systems (TOIS)*, 40(1):1–27, 2021.

[50] Khalid Nassiri and Moulay Akhloufi. Transformer models used for text-based question answering systems. *Applied Intelligence*, pages 1–34, 2022.

[51] Khalid Nassiri and Moulay Akhloufi. Transformer models used for text-based question answering systems. *Applied Intelligence*, 53(9):10602–10635, 2023.

[52] Graham Neubig. Neural machine translation and sequence-to-sequence models: A tutorial. *arXiv preprint arXiv:1703.01619*, 2017.

[53] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[54] Dae Hoon Park and Rikio Chiba. A neural language model for query auto-completion. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1189–1192, 2017.

[55] Greg Pass, Abdur Chowdhury, and Cayley Torgeson. A picture of search. In *Proceedings of the 1st international conference on Scalable information systems*, pages 1–es, 2006.

[56] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[57] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[58] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[59] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

[60] Ehud Reiter. A structured review of the validity of bleu. *Computational Linguistics*, 44(3):393–401, 2018.

[61] Amanda Ross and Victor L Willson. Paired samples t-test. In *Basic and advanced statistical tests*, pages 17–19. Brill, 2017.

[62] Tetsuya Sakai, Douglas W Oard, and Noriko Kando. *Evaluating Information Retrieval and Access Tasks: NTCIR's Legacy of Research Impact*. Springer Nature, 2021.

[63] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*, 2022.

[64] Alex Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404:132306, 2020.

[65] Milad Shokouhi. Learning to personalize query auto-completion. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 103–112, 2013.

[66] Alessandro Speggiorin. Document expansion: A comparison of neural generative transformer models, 2020. Honours Individual Project Dissertation submitted at the University of Glasgow during the academic year 2019/2020 and authored by myself (Alessandro Speggiorin - S1659400).

[67] Alessandro Speggiorin. Enhancing search engine efficiency with transformer-based language models for query autocompletion, 2023. Informatics Project Proposal Coursework submitted at the University of Edinburgh during the academic year 2022/2023 and authored by myself (Alessandro Speggiorin - S1659400).

[68] Alessandro Speggiorin. Transformers model for query autocompletion, 2023. MSc Progress Report Coursework submitted at the University of Edinburgh during

the academic year 2022/2023 and authored by myself (Alessandro Speggiorin - S1659400).

[69] KR Srinath. Python–the fastest growing programming language. *International Research Journal of Engineering and Technology*, 4(12):354–357, 2017.

[70] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014.

[71] Idan Szpektor, Aristides Gionis, and Yoelle Maarek. Improving recommendation for long-tail queries via templates. In *Proceedings of the 20th international conference on World wide web*, pages 47–56, 2011.

[72] Saedeh Tahery and Saeed Farzi. Customized query auto-completion and suggestion—a review. *Information Systems*, 87:101415, 2020.

[73] Cagri Toraman, Eyup Halit Yilmaz, Furkan Şahinuç, and Oguzhan Ozcelik. Impact of tokenization on language models: An analysis for turkish. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 22(4):1–21, 2023.

[74] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[75] Elan Van Biljon, Arnu Pretorius, and Julia Kreutzer. On optimal transformer depth for low-resource language translation. *arXiv preprint arXiv:2004.04418*, 2020.

[76] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[77] Savan Visalpara, Krishnateja Killamsetty, and Rishabh Iyer. A data subset selection framework for efficient hyper-parameter tuning and automatic machine learning. In *ICML Workshops*, 2021.

[78] Po-Wei Wang, Huan Zhang, Vijai Mohan, Inderjit S Dhillon, and J Zico Kolter. Realtime query completion via deep language models. *eCOM@ SIGIR*, 2319, 2018.

[79] Sida Wang, Weiwei Guo, Huiji Gao, and Bo Long. Efficient neural query auto completion. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 2797–2804, 2020.

[80] Yingfei Wang, Hua Ouyang, Hongbo Deng, and Yi Chang. Learning online trends for interactive query auto-completion. *IEEE Transactions on Knowledge and Data Engineering*, 29(11):2442–2454, 2017.

[81] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[82] Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. Byt5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306, 2022.

[83] Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online, June 2021. Association for Computational Linguistics.

[84] Li Yang and Abdallah Shami. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415:295–316, 2020.

[85] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019.

[86] Sina Zarrieß, Henrik Voigt, and Simeon Schüz. Decoding methods in neural language generation: a survey. *Information*, 12(9):355, 2021.

[87] Aston Zhang, Zachary C Lipton, Mu Li, and Alexander J Smola. Dive into deep learning. *arXiv preprint arXiv:2106.11342*, 2021.

[88] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019.

[89] Yidan Zhang, Yu Wan, Dayiheng Liu, Baosong Yang, and Zhenan He. Rmbr: A regularized minimum bayes risk reranking framework for machine translation. *arXiv preprint arXiv:2203.00201*, 2022.

[90] Kazimierz Zielinski, Radoslaw Nielek, Adam Wierzbicki, and Adam Jatowt. Computing controversy: Formal model and algorithms for detecting controversy on wikipedia and in search queries. *Information Processing & Management*, 54(1):14–36, 2018.

Appendix A

Additional Results

A.1 T5-Small Hyper-Parameters Search

		Temperature									
		0.2		0.4		0.6		0.8		1.0	
		Q	**S**	**Q**	**S**	**Q**	**S**	**Q**	**S**	**Q**	**S**
	20	0.545	0.540	0.540	0.539	0.538	0.527	0.532	0.527	0.510	0.512
	40	0.543	0.542	0.546	0.541	0.538	0.534	0.529	0.521	**0.508**	0.514
Top-k	**60**	0.545	0.540	**0.547**	0.536	0.531	0.531	0.525	0.524	0.512	0.511
	80	0.541	**0.544**	0.540	0.540	0.536	0.536	0.530	0.522	0.516	**0.503**
	100	0.545	0.541	0.545	0.538	0.537	0.533	0.527	0.528	0.515	0.509

Table A.1: T5-Small Top-k Sampling Search Hyper-parameters Search on a subset of 1000 prefixes from the AOL validation set. Results report the average BLEU score computed by generating one candidate completion per prefix. Scores are shown for models trained on the *Prefix* $\rightarrow$ *Query* (denoted as **Q**) and *Prefix* $\rightarrow$ *Suffix* (denoted as **S**) Data formats.

		Length Penalty											
		0.0		0.2		0.4		0.6		0.8		1.0	
		Q	S	Q	S	Q	S	Q	S	Q	S	Q	S
# Beams	**10**	0.547	**0.541**	0.547	**0.541**	0.549	0.543	0.549	0.544	**0.550**	0.543	**0.550**	0.543
	15	**0.544**	**0.541**	0.546	0.544	0.548	0.542	0.548	0.544	0.549	0.544	**0.550**	**0.545**
	20	0.545	0.542	0.547	**0.545**	0.549	0.543	0.549	0.544	**0.550**	0.544	**0.550**	**0.545**
	30	0.545	0.542	0.547	0.544	0.548	0.543	0.549	0.544	0.549	0.544	0.549	0.544

Table A.2: T5-Small Beam Search Hyper-parameters Search on a subset of 1000 prefixes from the AOL validation set. Results report the average BLEU score computed by generating one candidate completion per prefix. Scores are also shown for models trained on the *Prefix* $\rightarrow$ *Query* (denoted as **Q**) and *Prefix* $\rightarrow$ *Suffix* (denoted as **S**) Data formats.

A.2 ByT5-Small Hyper-Parameters Search

		Temperature									
		0.2		0.4		0.6		0.8		1.0	
		Q	S	Q	S	Q	S	Q	S	Q	S
	20	0.546	**0.550**	0.541	0.540	0.530	0.532	0.519	0.525	0.511	**0.504**
	40	0.545	0.547	0.538	0.542	0.530	0.538	0.526	0.523	0.513	0.510
Top-k	**60**	0.545	0.548	0.538	0.539	0.528	0.533	0.525	0.525	0.505	0.512
	80	**0.548**	0.547	0.540	0.541	0.530	0.534	0.521	0.522	**0.509**	0.508
	100	0.546	**0.550**	0.539	0.542	0.527	0.536	0.528	0.521	0.519	0.513

Table A.3: ByT5-Small Top-k Sampling Hyper-parameters Search on a subset of 1000 prefixes from the AOL validation set. Results report the average BLEU score computed by generating one candidate completion per prefix. Scores are shown for models trained on the *Prefix* $\rightarrow$ *Query* (denoted as **Q**) and *Prefix* $\rightarrow$ *Suffix* (denoted as **S**) Data formats.

		Length Penalty											
		0.0		**0.2**		**0.4**		**0.6**		**0.8**		**1.0**	
		Q	**S**	**Q**	**S**	**Q**	**S**	**Q**	**S**	**Q**	**S**	**Q**	**S**
# Beams	**10**	**0.548**	0.556	0.552	0.562	0.554	0.563	0.554	0.562	0.554	0.562	0.554	0.563
	15	0.549	0.555	**0.553**	0.561	0.555	**0.564**	0.555	0.563	0.556	0.562	0.556	**0.564**
	20	0.549	0.553	0.554	0.561	0.556	0.563	0.557	0.562	0.557	0.561	0.558	0.562
	30	0.550	0.554	0.555	0.562	0.558	**0.564**	0.558	0.563	0.558	0.563	**0.559**	**0.564**

Table A.4: ByT5-Small Beam Search Hyper-parameters Search on a subset of 1000 prefixes from the AOL validation set. Results report the average BLEU score computed by generating one candidate completion per prefix. Scores are shown for models trained on the *Prefix* $\rightarrow$ *Query* (denoted as **Q**) and *Prefix* $\rightarrow$ *Suffix* (denoted as **S**) Data formats.