Supporting Search Engines with Knowledge and Context

NIKOS VOSKARIDES

Supporting Search Engines with Knowledge and Context

Nikos Voskarides

Supporting Search Engines with Knowledge and Context

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. ir. K.I.J. Maex
ten overstaan van een door het College voor Promoties ingestelde
commissie, in het openbaar te verdedigen in
de Agnietenkapel
op vrijdag 5 februari 2021, te 16:00 uur

door

Nikos Voskarides

geboren te Lefkosia

Promotiecommissie

Promotor:	prof. dr. Maarten de Rijke	Universiteit van Amsterdam
Copromotor:	dr. Edgar J. Meij	Bloomberg
Overige leden:	prof. dr. Paul Groth	Universiteit van Amsterdam
	prof. dr. Evangelos Kanoulas	Universiteit van Amsterdam
	dr. Maarten Marx	Universiteit van Amsterdam
	prof. dr. Simone Teufel	University of Cambridge
	dr. Suzan Verberne	Universiteit Leiden

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

The research was supported by the Netherlands Organisation for Scientific Research (NWO) under project number CI-14-25.

Cover by Evros Voskarides
Printed by Ipskamp Printing, Amsterdam

ISBN: 978-94-6421-207-5

Acknowledgements

Maarten, your supervision has been transformative. Thank you for caring and for pushing me to become better.

Edgar, you have been a source of energy. Thank you for believing in me.

ILPS is a fantastic research group, mainly because of Maarten's vision and consistency.

> Mano, thank you for sharing your enthusiasm and for making sure I'd find my way.
>
> Petra, thank you for going above and beyond for the group.
>
> Ridho, for being my knowledge graph buddy.
>
> Katya, for having an alternative viewpoint.
>
> Marzieh, for reminding me of my roots.
>
> Anne, Daan, for encouraging me.
>
> Vangeli, for being unconventional.
>
> Harrie, for being a good listener.
>
> Ana, for speaking up.
>
> Ke, for sharing your passion.
>
> Rolf, for being punctual.
>
> Dan, for introducing me to the Guqin.
>
> Christophe, for setting the bar high.
>
> Sabrina, for seeing my work through a different lens.
>
> Bob, Chang, David, Hamid, Hosein, Ilya, Isaac, Maartje, Marlies, Mostafa, Pengjie, Tom, Wouter, Zhaochun, Ziming, and all the other ILPSers I worked with over the years, thank you for the fun memories.

I spent most of my PhD studies in Amsterdam.

> Giorgo, your help has been unmeasurable; thank you for sharing the burdens and multiplying the joys.
>
> Mario, thank you for not compromising.
>
> You both have been there for me in happy and rough times.
>
> Achillea, Eleni, Fee, Gianni, Ioanna, Kyriaco, Pari, Sofia, Stathi, you brought me joy every time I was around you. Thank you.

During my PhD studies I did two internships, both of which were incredibly rewarding experiences.

> In the summer of 2017, I interned at Bloomberg in London. Thank you Edgar, Ridho, Abhinav, Anju, Malvina, Miles, Diego for being great hosts. Nicoletta, Thiago, thank you for looking out for me. Iakove, Dora, Leo, Marilena, for all the fun times.
>
> In the summer of 2018, I interned at Amazon in Barcelona. Roi, Hugo, Lluis, Vassili, Marc, Jordi, thank you for making my stay memorable.

Thank you Maarten, Paul, Simone, Suzan and Vangeli for agreeing to serve in my committee. Giorgo, Katya, thank you for being my paranymphs. Harrie, thank you for translating my thesis summary to Dutch.

Next, my family.

> Papa, Mamma, your unconditional love and the example you have set have taught me the importance of knowledge and context. Thank you for enduring my absence all these years.
>
> This thesis is dedicated to you.
>
> Christofore, Evro, thank you for the (silent) support.
>
> Pappou, for the inspiration.

Iris, thank you. You are my only home.

Nikos Voskarides
Lefkosia, January 2021

Contents

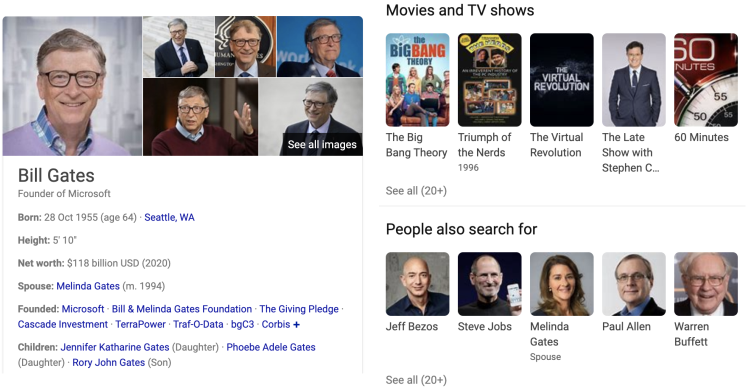
1 Introduction

Search engines leverage large repositories of knowledge to improve information access [115, 128, 147]. These repositories may store unstructured knowledge such as textual documents or social media posts, or structured knowledge such as attributes of and relationships between real-world objects and topics.

In order to effectively leverage knowledge, search engines should account for context, i.e., additional information about the user and the query [7, 18, 165, 194]. In this thesis, we study how to support search engines in leveraging knowledge while accounting for different types of context: (1) context that the search engine proactively provides to enrich search results (e.g., information on tourist attractions when searching for a city), (2) context that stems from the interactions between the user and the search engine in a conversational search session, or (3) context provided by the user to specify a broad query.

Search engine result pages (SERPs) present information that is meant to be relevant to the user's query [13, 78, 168]. Apart from the traditional "ten blue links", modern SERPs are increasingly being enriched with additional context that often comes from structured knowledge sources to enhance the user experience [47]. Knowledge graphs (KGs), which store world knowledge in the form of facts, are the most prominent structured knowledge source for search engines [25, 106, 147]. This is natural since the majority of queries issued to search engines contain entities stored in KGs [70]. KGs are used to support different components of modern SERPs, such as direct answers to user queries and KG panels [208], which present facts about the entity identified in the query and other, related entities to support exploratory search (see Figure 1.1) [21, 25, 75, 120]. A challenge that arises when presenting such structured knowledge in a SERP is that it is stored in a formal form, not directly suitable for presentation to the user. Tackling this challenge is the focus of the first research theme of this thesis, where we study how to make structured knowledge more accessible to the search engine user.

Users interact with the search results presented to them in multiple ways and they provide signals that may be used by search engines to improve the user experience, for instance by continuously learning better ranking functions [38, 79, 80, 84, 88]. Recent advances in natural language processing and deep learning have enabled the wide-spread use of interactive systems in real-world applications [64], which, in turn, has fueled a resurgence of research in conversational search [6, 45, 46, 155, 210]. In conversational search, the user interacts with the search engine during relatively short



Bill Gates
Founder of Microsoft

Born: 28 Oct 1955 (age 64) · Seattle, WA

Height: 5' 10"

Net worth: $118 billion USD (2020)

Spouse: Melinda Gates (m. 1994)

Founded: Microsoft · Bill & Melinda Gates Foundation · The Giving Pledge · Cascade Investment · TerraPower · Traf-O-Data · bgC3 · Corbis +

Children: Jennifer Katharine Gates (Daughter) · Phoebe Adele Gates (Daughter) · Rory John Gates (Son)

Movies and TV shows

- The Big Bang Theory
- Triumph of the Nerds 1996
- The Virtual Revolution
- The Late Show with Stephen C...
- 60 Minutes

People also search for

- Jeff Bezos
- Steve Jobs
- Melinda Gates Spouse
- Paul Allen
- Warren Buffett

Figure 1.1: Part of a SERP KG panel in response to the query “Bill Gates” (split in two parts).

sessions to gather knowledge over large unstructured knowledge repositories [16, 43]. A prominent challenge in conversational search is that the search engine has to keep track of the evolving context during the conversation so as to enable more natural interactions. Addressing this challenge is the focus of the second research theme of this thesis, where we study how to identify relevant context from the conversation history in order to improve interactive knowledge gathering.

Search engines facilitate knowledge gathering for different types of users. A large portion of research in information retrieval has focused on how to answer information needs of users in web search [27, 117, 124, 211]. In contrast to web search, in professional search, users express their information needs in a different way and aim to access and explore domain-specific knowledge [91, 157, 183]. Writers are a type of professional users who heavily rely on search engines [74, 173]. For instance, writers in the scientific domain use search engines to find relevant references to include in their articles [19, 76]. Another prominent example of professional search engine users are writers in the news domain [44, 51]. Such writers create narratives around specific events and use search engines to support them in this process [39, 93, 161]. In the third research theme of this thesis, we study how to support writers explore unstructured knowledge about past events given an incomplete narrative that specifies a main event and a context.

1.1 Research Outline and Questions

This thesis focuses on three research themes aimed at supporting search engines with knowledge and context: (1) making structured knowledge more accessible to the user by describing and contextualizing KG facts (Chapters 2, 3 and 4), (2) improving interactive knowledge gathering by identifying relevant context in conversational search (Chapter 5), and (3) supporting knowledge exploration for narrative creation by retrieving event-focused news articles in context (Chapter 6).

Below, we describe the main research questions for each chapter. In each chapter we describe more fine-grained subquestions that we ask to answer each main research question.

1.1.1 Making structured knowledge more accessible to the user

SERPs often include structured knowledge for queries that mention real-world entities in the form of KG facts. Facts are stored in KGs in a formal form (e.g., ⟨Bill Gates, founderOf, Microsoft⟩). When presenting a KG fact to the user, however, it is more natural to use human-readable descriptions that verbalize and contextualize the fact [66]. For instance, a possible description of the KG fact ⟨Bill Gates, founderOf, Microsoft⟩ is: *Bill Gates is an American business magnate and the principal founder of Microsoft Corporation.* In our first study (Chapter 2), we cast the problem of finding such descriptions as a retrieval task:

RQ1 Given a KG fact and a text corpus, can we retrieve textual descriptions of the fact from the text corpus?

We propose a method that first extracts and enriches candidate sentences that may be referring to the entities of the fact from a text corpus, and then ranks those sentences. Our results show that we can reliably retrieve sentences that accurately describe a given fact, under the condition that a relevant sentence exists in the underlying text corpus.

However, it is likely that this condition does not hold in cases where a given fact is not explicitly described in the text corpus at hand. This limits the applicability of our proposed method in real-world scenarios. In order to address this limitation, in our second study (Chapter 3), we consider a text generation task:

RQ2 Given a KG fact, can we automatically generate a textual description of the fact in the absence of an existing description?

We propose to first create sentence templates for each relationship in the KG using existing fact descriptions. Then, given a KG fact that expresses a specific relationship, we select a relevant template and fill it using additional information from the KG (other facts), if needed. We find that our method can generate contextually rich descriptions and is robust against KG incompleteness.

KG fact descriptions often contain mentions of other, related facts that provide additional context and thus increase the user's understanding of the fact as a whole (e.g., *Bill Gates founded Microsoft with Paul Allen*). Given the large size of KGs, many facts could potentially be relevant to the fact of interest, thus we need to automate the task of finding those other facts. This is the focus of our next study (Chapter 4):

RQ3 Can we contextualize a KG query fact by retrieving other, related KG facts?

We propose a method that first enumerates other candidate facts in the neighborhood of the query fact and then ranks those facts with respect to their relevance to the query fact. We propose the *neural fact contextualization method* (NFCM), a neural ranking model that combines automatically learned and hand-crafted features. In addition, we propose to use a distant supervision method to automatically gather training data for NFCM. We find that NFCM outperforms several baseline methods and that distant supervision is effective for this task.

1.1.2 Improving interactive knowledge gathering

The ultimate goal of conversational AI is interactive knowledge gathering [64]. Search engines can play a crucial role towards achieving that goal. An interactive search engine should support conversational search, where a user aims to interactively find information stored in large unstructured knowledge repositories [45].

In our next study (Chapter 5), we focus on multi-turn passage retrieval as an instance of conversational search [46]. Here, the query at the current turn may be underspecified. Thus, we need to identify relevant context from the conversation history to arrive at a better expression of the query. We answer the following research question:

RQ4 Can we use query resolution to identify relevant context and thereby improve retrieval in conversational search?

Here, *query resolution* refers to the task of adding missing context from the conversation history to the current turn query, if needed. We propose to model query resolution as a term classification task. We design *query resolution by term classification* (QuReTeC), a neural query resolution model based on bidirectional transformers. Since obtaining human-curated training data specifically for query resolution may be cumbersome, we propose a distant supervision method that automatically generates supervision data for QuReTeC using query-passage relevance pairs. We find that when integrating QuReTeC in a multi-stage ranking architecture we can significantly outperform baseline models. In addition, we find that the distant supervision method we propose can substantially reduce the amount of human-curated training data required to train QuReTeC.

1.1.3 Supporting knowledge exploration for narrative creation

Writers such as journalists often use search engines to find relevant material to include in event-oriented narratives [51, 83, 140]. Such material can provide background knowledge on the event itself or connections to other events that can help writers generate new angles on the narrative and thus better engage the reader [39, 93]. Previous work has focused on exploring knowledge for narrative creation from different sources, such as social media [44, 52, 213], or from sources with a more narrow scope, such as political speeches [113].

In our next study (Chapter 6), we focus on supporting knowledge exploration from a corpus of event-centric news articles for narrative creation. More specifically, we study a real-world scenario where the writer has already generated an incomplete narrative that specifies a main event and a context, and aims to retrieve relevant news articles that discuss other events from the past. We answer the following research question:

RQ5 Can we support knowledge exploration for event-centric narrative creation by performing news article retrieval in context?

We formally define this task and propose a retrieval dataset construction procedure that relies on existing news articles to simulate incomplete narratives and relevant articles. We conduct experiments on two datasets derived from this procedure and find that state-of-the-art lexical and semantic rankers are not sufficient for this task. We find that combining those rankers with one that ranks articles by reverse chronological order outperforms those rankers alone. We also perform an in-depth quantitative and qualitative analysis of the results along different dimensions to acquire insights into the characteristics of this task.

1.2 Main Contributions

In this section, we summarize the main contributions of this thesis.

Theoretical contributions

1. We formalize the task of retrieving knowledge graph fact descriptions stored in a text corpus (Chapter 2).

2. We formalize the task of generating knowledge graph fact descriptions (Chapter 3).

3. We formalize the task of knowledge graph fact contextualization (Chapter 4).

4. We formulate the task of query resolution for conversational search as term classification (Chapter 5).

5. We formalize the task of news article retrieval in context for event-centric narrative creation (Chapter 6).

Algorithmic contributions

6. A learning to rank method that combines a rich set of features for retrieving knowledge graph fact descriptions (Chapter 2).

7. A method for generating knowledge graph fact descriptions by template construction and filling (Chapter 3).

8. Neural fact contextualization method (NFCM), a method for contextualizing knowledge graph facts, and a distant supervision method for gathering training data automatically (Chapter 4).

9. Query resolution by term classification (QuReTeC), a method for query resolution for multi-turn passage ranking, and a distant supervision method for gathering training data automatically (Chapter 5).

10. A retrieval dataset construction procedure for the task of news article retrieval in context for event-centric narrative creation (Chapter 6).

Empirical contributions

11. Retrieving knowledge graph fact descriptions (Chapter 2)
    (a) Empirical comparison of our proposed learning to rank model and other sentence retrieval methods.
    (b) Empirical comparison of relationship-dependent models against an independent model.
    (c) Analysis of how different feature types contribute to the performance of our model and an error analysis of common errors made by our model.
12. Generating knowledge graph fact descriptions (Chapter 3)
    (a) Empirical comparison of different methods by automatic and manual evaluation.
    (b) Analysis of specific cases where our method succeeds or fails.
13. Contextualizing knowledge graph facts (Chapter 4)
    (a) Empirical comparison of NFCM and heuristic baselines.
    (b) We show that learning ranking functions using distant supervision is beneficial.
    (c) Analysis of the effect of handcrafted and automatically learned features on retrieval effectiveness.
14. Query resolution for conversational search (Chapter 5)
    (a) Empirical comparison of QuReTeC and multiple baselines of different nature.
    (b) We show that distant supervision can substantially reduce the amount of gold standard training data needed to train QuReTeC.
    (c) Qualitative analysis of specific cases where our method succeeds or fails.
15. News article retrieval in context (Chapter 6)
    (a) Empirical comparison of state-of-the-art lexical rankers on this task.
    (b) We show that a combination of lexical and semantic rankers with one that ranks articles by reverse chronological order outperforms those rankers alone.
    (c) An in-depth quantitative and qualitative analysis of the performance of the rankers under comparison among different dimensions.

Resources

16. A manually annotated dataset for knowledge graph fact description retrieval.
17. An automatically extracted dataset for knowledge graph fact description generation.
18. A manually annotated dataset for knowledge graph fact contextualization.
19. An open source implementation of QuReTeC.
20. An automatically extracted dataset for news article retrieval in context.

1.3 Thesis Overview

The thesis is organized in three parts.

In the first part we study how to make KG facts more accessible to users in search applications. Specifically, given a specific KG fact, we study how to retrieve textual descriptions of the fact (Chapter 2), how to generate a textual description of the fact in the absence of an existing description (Chapter 3), and how to retrieve other KG facts to contextualize the fact (Chapter 4).

In the second part we study how to improve interactive knowledge gathering by performing query resolution for multi-turn passage retrieval (Chapter 5).

In the third part we study how to support narrative creation by performing news article retrieval in context (Chapter 6).

In Chapter 7 we conclude the thesis and discuss directions for future work.

1.4 Origins

Below we list which publication is the origin of each chapter.

Chapter 2 is based on the conference paper: N. Voskarides, E. Meij, M. Tsagkias, M. de Rijke, and W. Weerkamp. Learning to explain entity relationships in knowledge graphs. In *ACL-IJCNLP*. ACL, 2015 [185].

NV designed the method and ran the experiments. EM helped with algorithmic design. All authors conributed to the text, NV did most of the writing.

Chapter 3 is based on the conference paper: N. Voskarides, E. Meij, and M. de Rijke. Generating descriptions of entity relationships. In *ECIR*. Springer, 2017 [186].

NV designed the method and ran the experiments. All authors contributed to the text, NV did most of the writing.

Chapter 4 is based on the conference paper: N. Voskarides, E. Meij, R. Reinanda, A. Khaitan, M. Osborne, G. Stefanoni, K. Prabhanjan, and M. de Rijke. Weakly-supervised contextualization of knowledge graph facts. In *SIGIR*. ACM, 2018 [187].

NV designed the method and ran the experiments. EM, RR contributed to the experimental design. AK helped with the infrastructure. All authors contributed to the text, NV did most of the writing.

Chapter 5 is based on the conference paper: N. Voskarides, D. Li, P. Ren, E. Kanoulas, and M. de Rijke. Query resolution for conversational search with limited supervision. In *SIGIR*. ACM, 2020 [189].

NV designed the method and ran the experiments. DL contributed to the experimental design and ran baseline models. All authors contributed to the text, NV did most of the writing.

Chapter 6 is based on the conference paper: N. Voskarides, E. Meij, S. Sauer, and M. de Rijke. News article retrieval in context for event-centric narrative creation. In *Under submission*, 2020 [190].

NV designed the method and ran the experiments. All authors contributed to the text, NV did most of the writing.

The thesis also indirectly benefited from insights gained from the following publications:

- N. Voskarides, D. Odijk, M. Tsagkias, W. Weerkamp, and M. de Rijke. Query-dependent contextualization of streaming data. In *ECIR*. Springer, 2014 [184].
- N. Voskarides, D. Li, A. Panteli, and P. Ren. ILPS at TREC 2019 Conversational Assistant Track. TREC, NIST, 2019 [188].
- G. Sidiropoulos, N. Voskarides, and E. Kanoulas. Knowledge graph simple question answering for unseen domains. In *AKBC*, 2020 [166].
- F. Sarvi, N. Voskarides, L. Mooiman, S. Schelter, and M. de Rijke. A comparison of supervised learning to match methods for product search. In *eCOM 2020: The 2020 SIGIR Workshop on eCommerce*. ACM, 2020 [160].
- A. M. Krasakis, M. Aliannejadi, N. Voskarides, and E. Kanoulas. Analysing the effect of clarifying questions on document ranking in conversational search. In *ICTIR*. ACM, 2020 [95].

Part I

Making Structured Knowledge more Accessible to the User

2 Retrieving Knowledge Graph Fact Descriptions

In the first part of this thesis, we study how to make structured knowledge more accessible to the user. In this chapter, we aim to answer **RQ1**: Given a KG fact and a text corpus, can we retrieve textual descriptions of the fact from the text corpus?

Knowledge graph (KG) facts express entity relationships in a formal form. In the scope of this chapter we use the term “explaining entity relationships” as an alias for “retrieving KG fact descriptions”.

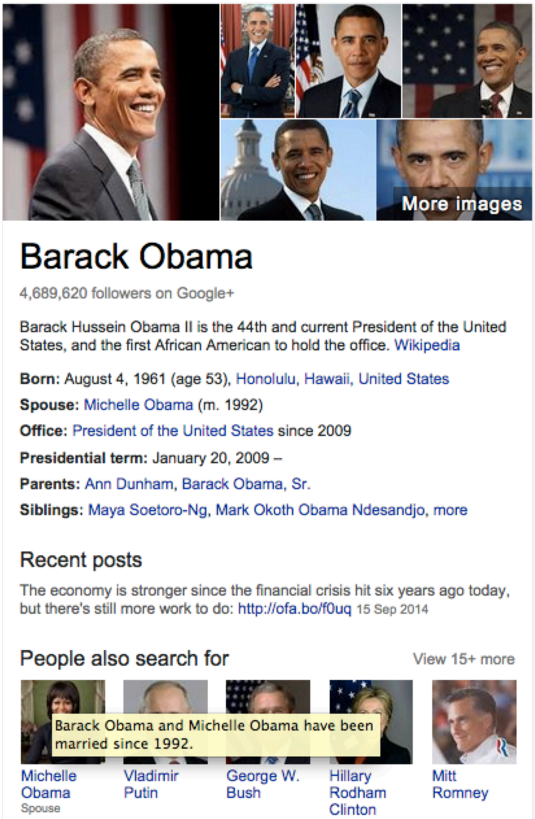
2.1 Introduction

Knowledge graphs are a powerful tool for supporting a large spectrum of search applications including ranking, recommendation, exploratory search, and web search [56]. A knowledge graph aggregates information around entities across multiple content sources and links these entities together, while at the same time providing entity-specific properties (such as age or employer) and types (such as actor or movie).

Although there is a growing interest in automatically constructing knowledge graphs, e.g., from unstructured web data [42, 60, 193], the problem of providing evidence on why two entities are related in a knowledge graph remains largely unaddressed. Extracting and presenting evidence for linking two entities, however, is an important aspect of knowledge graphs, as it can enforce trust between the user and a search engine, which in turn can improve long-term user engagement, e.g., in the context of related entity recommendation [21]. Although knowledge graphs exist that provide this functionality to a certain degree (e.g., when hovering over Google’s suggested entities, see Figure 2.1), to the best of our knowledge there is no previously published research on methods for entity relationship explanation.

In this chapter we propose a method for explaining the relationship between two entities, which we evaluate on a newly constructed annotated dataset that we make publicly available. In particular, we consider the task of explaining relationships between pairs of Wikipedia entities. We aim to infer a human-readable description for an entity pair given a relationship between the two entities. Since Wikipedia does not explicitly

This chapter was published as [185].



Barack Obama

4,689,620 followers on Google+

Barack Hussein Obama II is the 44th and current President of the United States, and the first African American to hold the office. Wikipedia

Born: August 4, 1961 (age 53), Honolulu, Hawaii, United States

Spouse: Michelle Obama (m. 1992)

Office: President of the United States since 2009

Presidential term: January 20, 2009 –

Parents: Ann Dunham, Barack Obama, Sr.

Siblings: Maya Soetoro-Ng, Mark Okoth Obama Ndesandjo, more

Recent posts

The economy is stronger since the financial crisis hit six years ago today, but there's still more work to do: http://ofa.bo/f0uq 15 Sep 2014

People also search for

Barack Obama and Michelle Obama have been married since 1992.

Figure 2.1: Part of Google's search result page for the query "barack obama". When hovering over the related entity "Michelle Obama", an explanation of the relationship between her and "Barack Obama" is shown.

define relationships between entities we use a knowledge graph to obtain these relations. We cast our task as a sentence ranking problem: we automatically extract sentences from a corpus and rank them according to how well they describe a given relationship between a pair of entities. For ranking purposes, we extract a rich set of features and use learning to rank to effectively combine them. Our feature set includes both traditional information retrieval and natural language processing features that we augment with entity-dependent features. These features leverage information from the structure of the knowledge graph. On top of this, we use features that capture the presence in a sentence of the relationship of interest. For our evaluation we focus on "people" entities and we use a large, manually annotated dataset of sentences.

We break down **RQ1** to three research sub-questions. First, we ask what the effectiveness of state-of-the-art sentence retrieval models is for explaining a relationship between two entities (**RQ1.1**). Second, we consider whether we can improve over sentence retrieval models by casting the task in a learning to rank framework (**RQ1.2**). Third, we examine whether we can further improve performance by using relationship-dependent models instead of a relationship-independent one (**RQ1.3**). We complement these research questions with an error and feature analysis.

Our main contributions are a robust and effective method for explaining entity relationships, detailed insights into the performance of our method and features, and a manually annotated dataset.

2.2 Related Work

We combine ideas from sentence retrieval, learning to rank, and question answering to address the task of explaining relationships between entities.

Previous work that is closest to the task we address in this chapter is that of Blanco and Zaragoza [20] and Fang et al. [61]. First, Blanco and Zaragoza [20] focus on finding and ranking sentences that explain the relationship between an entity and a query. Our work is different in that we want to explain the relationship between two entities, rather than a query. Fang et al. [61] explore the generation of a ranked list of knowledge base relationships for an entity pair. Instead, we try to select sentences that describe a particular relationship, assuming that this is given.

Our approach builds on sentence retrieval, where one retrieves sentences rather than documents that answer an information need. Document retrieval models such as tf-idf, BM25, and language modeling [11] have been extended to tackle sentence retrieval. Three of the most successful sentence retrieval methods are TFISF [8], which is a variant of the vector space model with tf-idf weighting, language modeling with local context [62, 126], and a recursive version of TFISF that accounts for local context [54]. TFISF is very competitive compared to document retrieval models tuned specifically for sentence retrieval (e.g., BM25 and language modeling [110]) and we therefore include it as a baseline.

Sentences that are suitable for explaining relationships can have attributes that are important for ranking but cannot be captured by term-based retrieval models. One way to combine a wide range of ranking features is learning to rank (LTR). Recent years have witnessed a rapid increase in the work on learning to rank, and it has proven to be a very successful method for combining large numbers of ranking features, for web search, but also other information retrieval applications [3, 28, 171]. We use learning to rank and represent each sentence with a set of features that aim to capture different dimensions of the sentence.

Question answering (QA) is the task of providing direct and concise answers to questions formed in natural language [77]. QA can be regarded as a similar task to ours, assuming that the combination of entity pair and relationship form the "question" and that the "answer" is the sentence describing the relationship of interest. Even though we do not follow the QA paradigm in this chapter, some of the features we use are inspired by QA systems. In addition, we employ learning to rank to combine the devised features, which has recently been successfully applied for QA [3, 171].

2.3 Problem Statement

We address the problem of explaining relationships between pairs of entities in a knowledge graph. We operationalize the problem as a problem of ranking sentences from documents in a corpus that is related to the knowledge graph. More specifically, given two entities e_i and e_j that form an entity pair $\langle e_i, e_j \rangle$, and a relation r between them, the task is to extract a set of candidate sentences $S_{ij} = \{s_{ij_1}, \ldots, s_{ij_k}\}$ that refer to $\langle e_i, e_j \rangle$ and to impose a ranking on the sentences in S_{ij}. The relation r has the general form $\langle \mathit{type}(e_i), \mathit{terms}(r), \mathit{type}(e_j) \rangle$, where $\mathit{type}(e)$ is the type of the entity e (e.g., `Person` or `Actor`) and $\mathit{terms}(r)$ are the terms of the relation (e.g., `CoCastsWith` or `IsSpouseOf`).

We are left with two specific tasks: (1) extracting candidate sentences S_{ij}, and (2) ranking S_{ij}, where the goal is to have sentences that provide a perfect explanation of the relationship at the top position of the ranking. The next section describes our methods for both tasks.

2.4 Explaining Entity Relationships

We follow a two-step approach for automatically explaining relationships between entity pairs. First, in Section 2.4.1, we extract and enrich sentences that refer to an entity pair $\langle e_i, e_j \rangle$ from a corpus in order to construct a set of candidate sentences. Second, in Section 2.4.2, we extract a rich set of features describing the entities' relationship r and use supervised machine learning in order to rank the sentences in S_{ij} according to how well they describe the relationship r.

2.4.1 Extracting candidate sentences

To create a set of candidate sentences for a given entity pair and relationship, we require a corpus of documents that is pertinent to the entities at hand. Although any kind of document collection can be used, we focus on Wikipedia in this chapter, as it provides good coverage for the majority of entities in our knowledge graph.

First, we extract surface forms for the given entities: the title of the entity's Wikipedia article (e.g., "Barack Obama"), the titles of all redirect pages linking to that article (e.g., "Obama"), and all anchor text associated with hyperlinks to the article within Wikipedia (e.g., "president obama"). We then split all Wikipedia articles into sentences and consider a sentence as a candidate if (i) the sentence is part of either entities' Wikipedia article and contains a surface form of, or a link to, the other entity; or (ii) the sentence contains surface forms of, or links to, both entities in the entity pair.

Next, we apply two sentence enrichment steps for (i) making sentences self-contained and readable outside the context of the source document and (ii) linking the sentences to entities. For (i), we replace pronouns in candidate sentences with the title of the entity. We apply a simple heuristic for the people entities, inspired by [197]:[1] we count the frequency of the terms "he" and "she" in the article for determining

[1] We experimented with the Stanford co-reference resolution system [101] and Apache OpenNLP and found that they were not able to consistently achieve the level of effectiveness that we require.

the gender of the entity, and we replace the first appearance of “he” or “she” in each sentence with the entity’s title. We skip this step if any surface form of the entity occurs in the sentence.

For (ii), we apply entity linking to provide links from the sentence to additional entities [121]. This need arises from the fact that not every sentence in an article contains explicit links to the entities it mentions, as Wikipedia guidelines only allow one link to another article in the article’s text.[2] The algorithm takes a sentence as input and iterates over n-grams that are not yet linked to an entity. If an n-gram matches a surface form of an entity, we establish a link between the n-gram and the entity. We restrict our search space to entities that are linked from within the source article of the sentence and from within articles to which the source article links. This way, our entity linking method achieves high precision as almost no disambiguation is necessary.

As an example, consider the sentence “He gave critically acclaimed performances in the crime thriller Seven…” on the Wikipedia page for Brad Pitt. After applying our enrichment steps, we obtain “`Brad_Pitt` gave critically acclaimed performances in the crime thriller `Seven`…”, making the sentence human readable and link to the entities `Brad_Pitt` and `Seven_(1995_film)`.

2.4.2 Ranking sentences

After extracting candidate sentences, we rank them by how well they describe the relationship of interest r between entities e_i and e_j. There are many signals beyond simple term statistics that can indicate relevance. Automatically constructing a ranking model using supervised machine learning techniques is therefore an obvious choice. For ranking we use learning to rank (LTR) and represent each sentence with a rich set of features. Tables 2.1 and 2.2 list the features we use. Below we provide a brief description of the more complex ones.

Text features This feature type regards the importance of the sentence s at the term level. We compute the density of s (feature 4) as:

$$density(s) = \frac{1}{K \cdot (K+1)} \sum_{j=1}^{n} \frac{idf(t_j) \cdot idf(t_{j+1})}{distance(t_j, t_{j+1})^2}, \tag{2.1}$$

where K is the number of keyword terms in s and $distance(t_j, t_{j+1})$ is the number of non-keyword terms between keyword terms t_j and t_{j+1}. We treat stop words and numbers in s as non-keywords and the remaining terms as keywords. Features 5–8 capture the distribution of part-of-speech tags in the sentence.

Entity features These features partly build on [118, 177] and describe the entities and are dependent on the knowledge graph. Whether e_i or e_j is the first appearing entity in a sentence might be an indicator of importance (feature 13). The spread of e_i and e_j in the sentence (feature 14) might be an indicator of their centrality in the sentence [20]. Features 15–22 capture the distribution of part-of-speech tags in the

[2] `http://en.Wikipedia.org/wiki/Wikipedia:Manual_of_Style/Linking`

Table 2.1: Text and entity features used for sentence ranking.

#	Name	Gloss
Text features		
1	Sentence length	Length of s in words
2	Sum of idf	Sum of IDF of terms of s in Wikipedia
3	Average idf	Average IDF of terms of s in Wikipedia
4	Sentence density	Lexical density of s, see Equation 2.1 [100]
5–8	POS fractions	Fraction of verbs, nouns, adjectives, others in s [122]
Entity features		
9	#entities	Total number of entities in s
10	Link to e_i	Whether s contains a link to the entity e_i
11	Link to e_j	Whether s contains a link to the entity e_j
12	Links to e_i and e_j	Whether s contains links to both entities e_i and e_j
13	Entity first	Is e_i or e_j the first entity in the sentence?
14	Spread of e_i, e_j	Distance between the last match of e_i and e_j in s [20]
15–22	POS fractions left/right	Fraction of verbs, nouns, adjectives, others to the left/right window of e_i and e_j in s [122]
23–25	#entities left/right/between	Number of entities to the left/right or between entities e_i and e_j in s
26	common links e_i, e_j	Whether s contains any common link of e_i and e_j
27	#common links	The number of common links of e_i and e_j in s
28	Score common links e_i, e_j	Sum of the scores of the common links of e_i and e_j in s
29–30	#common links prev/next	The number of common links of e_i and e_j in previous/next sentence of s

sentence in a window of four words around e_i or e_j in s [122], complemented by the number of entities between, to the left of, and to the right of the entity pair (features 23–25).

We assume that two articles that have many common articles that point to them are strongly related [196]. We hypothesize that, if a sentence contains common inlinks from e_i and e_j, the sentence might contain important information about their relationship. Hence, we add whether the sentence contains a common link (feature 26) and the number of common links (feature 27) as features. We score a common link l between e_i and e_j using:

$$score(l, e_i, e_j) = sim(l, e_i) \cdot sim(l, e_j), \tag{2.2}$$

where $sim(\cdot, \cdot)$ is defined as the similarity between two Wikipedia articles, computed

Table 2.2: Relationship and source features used for sentence ranking.

#	Name	Gloss
Relationship features		
31	Match $terms(r)$?	Whether s contains any term in $terms(r)$
32	Match $wordnet(r)$?	Whether s contains any phrase in $wordnet(r)$
33	Match $word2vec(r)$?	Whether s contains any phrase in $word2vec(r)$
34–36	or's	Boolean OR of feature 31 and one or both of features 32 and 33
37–38	or(31, 32, 33) prev/next	Boolean OR of features 31, 32, 33 for the previous/next sentence of s
39	Average $word2vec(r)$	Average cosine similarity of phrases in $word2vec(r)$ that are matched in s
40	Maximum $word2vec(r)$	Maximum cosine similarity of phrases in $word2vec(r)$ that are matched in s
41	Sum $word2vec(r)$	Sum of cosine similarity of phrases in $word2vec(r)$ that are matched in s
42	Score LC	Lucene score of s with $titles(e_i, e_j)$, $terms(r)$, $wordnet(r)$, $word2vec(r)$ as query
43	Score R-TFISF	R-TFISF score of s with queries constructed as above
Source features		
44	Sentence position	Position of s in document from which it originates
45	From e_i or e_j?	Does s originate from the Wikipedia article of e_i or e_j?
46	#(e_i or e_j)	Number of occurrences of e_i or e_j in document from which s originates, inspired by document smoothing for sentence retrieval [125]

using a variant of Normalized Google Distance [196]. Feature 28 then measures the sum of the scores of the common links.

Previous research shows that using surrounding sentences is beneficial for sentence retrieval [54]. We therefore consider the number of common links in the previous and next sentence (features 29–30).

Relationship features Feature 31 indicates whether any of the relationship-specific terms occurs in the sentence. Only matching the terms in the relationship may have low coverage since terms such as “spouse” may have many synonyms and/or highly related terms, e.g., “husband” or “married”. Therefore, we use WordNet to find synonym phrases of r (feature 32); we refer to this method as $wordnet(r)$.

Alternatively, we use word embeddings to find such similar phrases [119]. Such embeddings take a text corpus as input and learn vector representations of words and phrases consisting of real numbers. Given the set V_r consisting of the vector representations of all the relationship terms and the set V which consists of the vector representations of all the candidate phrases in the data, we calculate the distance between a candidate phrase represented by a vector $\mathbf{v}_i \in V$ and the vectors in V_r as:

$$distance(\mathbf{v}_i, V) = \cos\left(\mathbf{v}_i, \sum_{\mathbf{v}_j \in V_r} \mathbf{v}_j\right), \tag{2.3}$$

where $\sum_{\mathbf{v}_j \in V_r} \mathbf{v}_j$ is the element-wise sum of the vectors in V_r and the distance between two vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ is measured using cosine similarity. The candidate phrases in V are then ranked using Equation 2.3 and the top-m phrases are selected, resulting in features 33, 39, 40, and 41; we refer to the ranked set of phrases that are selected using this procedure as $word2vec(r)$.

In addition, we employ state-of-the-art retrieval functions and include the scores for queries that are constructed using the entities e_i and e_j, the relation r, $wordnet(r)$, and $word2vec(r)$. We use the titles of the entity articles $titles(e)$ to represent the entities in the query and two ranking functions, Recursive TFISF (R-TFISF) and LC,[3] (features 42–43). TFISF is a sentence retrieval model that determines the level of relevance of a sentence s given a query q as:

$$R(s,q) = \sum_{t \in q} \log(tf_{t,q} + 1) \cdot \log(tf_{t,s} + 1) \cdot \log\left(\frac{n+1}{0.5 + sf_t}\right), \tag{2.4}$$

where $tf_{t,q}$ and $tf_{t,s}$ are the number of occurrences of term t in the query q and the sentence s respectively, sf_t is the number of sentences in which t appears, and n is the number of sentences in the collection. R-TFISF is an improved extension of the TFISF method [54], which incorporates context from neighboring sentences in the ranking function:

$$R_c(s,q) = (1-\mu)R(s,q) + \mu[R_c(s_{prev}(s), q) + R_c(s_{next}(s), q)],$$

where μ is a free parameter and $s_{prev}(s)$ and $s_{next}(s)$ indicate functions to retrieve the previous and next sentence, respectively. We use a maximum of three recursive calls.

Source features Here, we refer to features that are dependent on the source document of the sentences. We have three such features.

2.5 Experimental Setup

In this section we describe the dataset, manual annotations, learning to rank algorithm, and evaluation metrics that we use to answer our research questions.

[3] In preliminary experiments R-TFISF and LC were the best performing among a pool of sentence retrieval methods.

2.5.1 Dataset

We draw entities and their relationships from a proprietary knowledge graph that is created from Wikipedia, Freebase, IMDB, and other sources, and that is used by the Yahoo web search engine. We focus on "people" entities and relationships between them.[4] For our experiments we need to select a manageable set of entities, which we obtain as follows. We consider a year of query logs from a large commercial search engine, count the number of times a user clicks on a Wikipedia article of an entity in the results page and perform stratified sampling of entities according to this distribution. As we are bounded by limited resources for our manual assessments, we sample 1476 entity pairs that together with nine unique relationship types form our experimental dataset.

We use an English Wikipedia dump dated July 8, 2013, containing approximately 4M articles, of which 50638 belong to "people" entities that are also in our knowledge graph. We extract sentences using the approach described in Section 2.4.1, resulting in 36823 candidate sentences for our entities. On average we have 24.94 sentences per entity pair (maximum 423 and minimum 0). Because of the large variance, it is not feasible to obtain exhaustive annotations for all sentences. We rank the sentences using R-TFISF and keep the top-10 sentences per entity pair for annotation. This results in a total of 5689 sentences.

Five human annotators provided relevance judgments, manually judging sentences based on how well they describe the relationship for an entity pair, for which we use a five-level graded relevance scale (perfect, excellent, good, fair, bad).[5] Of all relevance grades 8.1% is perfect, 15.69% excellent, 19.98% good, 8.05% fair, and 48.15% bad. Out of 1476 entity pairs, 1093 have at least one sentence annotated as fair. As is common in information retrieval evaluation, we discard entity pairs that have only "bad" sentences. We examine the difficulty of the task for human annotators by measuring inter-annotator agreement on a subset of 105 sentences that are judged by 3 annotators. Fleiss' kappa is $k = 0.449$, which is considered to be moderate agreement.

2.5.2 Machine learning

For ranking sentences we use a Random Forest (RF) classifier [26].[6] We set the number of iterations to 300 and the sampling rate to 0.3. Experiments with varying these two parameters did not show any significant differences. We also tried several feature normalization methods, none of them being able to significantly outperform the runs without feature normalization.

We obtain POS tags using the Stanford part-of-speech tagger and filter out a standard list of 33 English stopwords. For the word embeddings we use *word2vec* and train our model on all text in Wikipedia using negative sampling and the continuous bag of words architecture. We set the size of the phrase vectors to 500 and $m = 30$.

[4] Note that, except for the co-reference resolution step described in Section 2.4.1, our method does not depend on this restriction.

[5] `https://github.com/nickvosk/acl2015-dataset-learning-to-explain-entity-relationships`

[6] In preliminary experiments, we contrasted RF with gradient boosted regression trees and LambdaMART and found that RF consistently outperformed other methods.

Table 2.3: Results for five baseline variants. See text for their description and significant differences.

Baseline	NDCG@1	NDCG@10	ERR@1	ERR@10
B1	0.7508	0.8961	0.3577	0.4531
B2	0.7511	0.8958	0.3584	0.4530
B3	0.7595	0.8997	0.3696	0.4600
B4	0.7767	0.9070	0.3774	0.4672
B5	**0.7801**	**0.9093**	**0.3787**	**0.4682**

2.5.3 Evaluation metrics

We employ two main evaluation metrics in our experiments, NDCG [85] and ERR [33]. The former measures the total accumulated gain from the top of the ranking that is discounted at lower ranks and is normalized by the ideal cumulative gain. The latter models user behavior and measures the expected reciprocal rank at which a user will stop her search. We consider these ranking-based graded evaluation metrics at two cut-off points: position 1, corresponding to showing a single sentence to a user, and 10, which accounts for users who might look at more results. We report on NDCG@1, NDCG@10, ERR@1, ERR@10, and Exc@1, which indicates whether we have an "excellent" or "perfect" sentence at the top of the ranking. Likewise, Per@1 indicates whether we have a "perfect" sentence at the top of the ranking (not all entity pairs have an excellent or a perfect sentence).

We perform 5-fold cross validation and test for statistical significance using a paired two-tailed t-test. We depict a significant difference in performance for $p < 0.01$ with $^\blacktriangle$ (gain) and $^\blacktriangledown$ (loss) and for $p < 0.05$ with $^\vartriangle$ (gain) and $^\triangledown$ (loss). Boldface indicates the best score for a metric.

2.6 Results and Analysis

We compare the performance of typical document retrieval models and state-of-the-art sentence retrieval models in order to answer **RQ1.1**. We consider five sentence retrieval models: Lucene ranking (LC), language modeling with Dirichlet smoothing (LM), BM25, TFISF, and Recursive TF-ISF (R-TFISF). We follow related work and set $\mu = 0.1$ for R-TFISF, $k = 1$ and $b = 0$ for BM25 and $\mu = 250$ for LM [62].

In our experiments, a query q is constructed using various combinations of surface forms of the two entities e_i and e_j and the relationship r. Each entity in the entity pair can be represented by its title, the titles of any redirect pages pointing to the entity's article, the n-grams used as anchors in Wikipedia to link to the article of the entity, or the union of them all. The relationship r can be represented by the terms in the relationship, synonyms in $wordnet(r)$, or by phrases in $word2vec(r)$.

First, we fix the way we represent r. Baseline B1 does not include any representation of r in the query. B2 includes the relationship terms of r, while B3 includes the relationship terms of r and the synonyms in $wordnet(r)$. B4 includes the terms of r and the phrases in $word2vec(r)$, and B5 includes the relationship terms of r, the synonyms in $wordnet(r)$ and the phrases in $word2vec(r)$. Combining these variations with the entity representations, we find that all combinations that use the titles as representation and R-TFISF as the retrieval function outperform all other combinations.This can be explained by the fact that titles are least ambiguous, thus reducing the possibility of accidentally referring to other entities. BM25 and LC perform almost as well as R-TFISF, with only insignificant differences in performance.

Table 2.3 shows the best performing combination of each baseline, i.e., varying the representation of r and using titles and R-TFISF. B4 and B5 are the best performing baselines, suggesting that $word2vec(r)$ and $wordnet(r)$ are beneficial. B5 significantly outperforms all baselines except B4.

We also experiment with a supervised combination of the baseline rankers using LTR. Here, we consider each baseline ranker as a separate feature and train a ranking model. The trained model is not able to outperform the best individual baseline, however.

2.6.1 Learning to rank sentences

Next, we provide the results of our method using the features described in Section 2.4.2, exploring whether our learning to rank (LTR) approach improves over sentence retrieval models (**RQ1.2**). We compare an LTR model using the features in Tables 2.1 and 2.2 against the best baseline (B5).

Table 2.4 shows the results. Each group in the table contains the results for the entity pairs that have at least one candidate sentence of that relevance grade for B5 and LTR.

We find that LTR significantly outperforms B5 by a large margin. The absolute performance difference between LTR and B5 becomes larger for all metrics as we move from “fair” to “perfect,” which shows that LTR is more robust than the baseline for entity pairs that have at least one high quality candidate sentence. LTR ranks the best possible sentence at the top of the ranking for $\sim$83% of the cases for entity pairs that contain an “excellent” sentence and for $\sim$72% of the cases for entity pairs that contain a “perfect” sentence.

Note that, as indicated in Section 2.5.1, we discard entity pairs that have only “bad” sentences in our experiments. For the sake of completeness, we report on the results for all entity pairs in our dataset—including those without any relevant sentences—in Table 2.5.

2.6.2 Relationship-dependent models

Relevant sentences may have different properties for different relationship types. For example, a sentence describing two entities being partners would have a different form than one describing that two entities costar in a movie. A similar idea was investigated in the context of QA for associating question and answer types [205]. To answer **RQ1.3** we examine whether learning a relationship-dependent model improves over learning a single model for all types. We split our dataset per relationship type and train a model per type using 5-fold cross-validation within each. Table 2.6 shows the results. Our method is robust across different relationships in terms of NDCG. However, we observe some variation in ERR as this metric is more sensitive to the

Table 2.4: Results for the best baseline (B5) and the learning to rank method (LTR).

Has one	# pairs	# sentences	Method	NDCG@1	NDCG@10	ERR@1	ERR@10	Exc@1	Per@1
fair	1093	4435	B5	0.7801	0.9093	0.3787	0.4682	–	–
			LTR	**0.8489**▲	**0.9375**▲	**0.4242**▲	**0.4980**▲	–	–
good	1038	4285	B5	0.7742	0.9078	0.3958	0.4894	–	–
			LTR	**0.8486**▲	**0.9374**▲	**0.4438**▲	**0.5208**▲	–	–
excellent	752	3387	B5	0.7455	0.8999	0.4858	0.5981	0.7314	–
			LTR	**0.8372**▲	**0.9340**▲	**0.5500**▲	**0.6391**▲	**0.8298**▲	–
perfect	339	1687	B5	0.7082	0.8805	0.6639	0.7878	0.7729	0.6136
			LTR	**0.8150**▲	**0.9245**▲	**0.7640**▲	**0.8518**▲	**0.8909**▲	**0.7227**▲

Table 2.5: Results for the best baseline (B5) and the learning to rank method (LTR), using all entity pairs in the dataset, including those without any relevant sentences.

Has one	# pairs	# sentences	Method	NDCG@1	NDCG@10	ERR@1	ERR@10	Exc@1	Per@1
-	1476	5689	B5	0.5776	0.6733	0.2804	0.3467	–	–
			LTR	**0.6285**$^{\blacktriangle}$	**0.6940**$^{\blacktriangle}$	**0.3155**$^{\blacktriangle}$	**0.3694**$^{\blacktriangle}$	–	–

Table 2.6: Results for relationship-dependent models. Similar relationships are grouped together.

Relationship	# pairs	# sentences	NDCG@1	NDCG@10	ERR@1	ERR@10
$\langle MovieActor, CoCastsWith, MovieActor\rangle$	410	1403	0.8604	0.9436	0.3809	0.4546
$\langle TvActor, CoCastsWith, TvActor\rangle$	210	626	0.8729	0.9482	0.3271	0.3845
$\langle MovieActor, IsDirectedBy, MovieDirector\rangle$ $\langle MovieDirector, Directs, MovieActor\rangle$	112	492	0.8795	0.9396	0.4709	0.5261
$\langle Person, isChildOf, Person\rangle$ $\langle Person, isParentOf, Person\rangle$	108	716	0.8428	0.9081	0.6395	0.7136
$\langle Person, isPartnerOf, Person\rangle$ $\langle Person, isSpouseOf, Person\rangle$	155	877	0.8623	0.9441	0.6153	0.6939
$\langle Athlete, PlaysSameSportTeamAs, Athlete\rangle$	98	321	0.8787	0.9535	0.3350	0.3996
Average results over all data	1093	4435	**0.8661**	**0.9395**	**0.4615**	**0.5287**
LTR (Table 2.4; fair)			0.8489	0.9375	0.4242	0.4980

Table 2.7: Results using relationship-dependent models, removing individual feature types.

Features	NDCG@1	NDCG@10	ERR@1	ERR@10
All	**0.8661**	**0.9395**	**0.4615**	**0.5287**
All\text	0.8620	0.9372	0.4606	0.5274
All\source	0.8598	0.9372	0.4582	0.5261
All\entity	0.8421$^{\triangledown}$	0.9282$^{\blacktriangledown}$	0.4497	0.5202$^{\triangledown}$
All\relation	0.8183$^{\blacktriangledown}$	0.9201$^{\blacktriangledown}$	0.4352$^{\blacktriangledown}$	0.5112$^{\blacktriangledown}$

distribution of relevant items than NDCG—the distribution over relevance grades varies per relationship type. For example, it is much more likely to find candidate sentences that have a high relevance grade for $\langle Person$, $isSpouseOf$, $Person\rangle$ than for $\langle Athlete$, $PlaysSameSportTeamAs$, $Athlete\rangle$ in our dataset. We plan to address this issue by exploring other corpora in the future.

The second-to-last row in Table 2.6 shows the averaged results over the different relationship types, which is a significant improvement over LTR at $p < 0.01$ for all metrics. This method ranks the best possible sentence at the top of the ranking for $\sim$85% of the cases for entity pairs that contain an "excellent" sentence ($\sim$2% absolute improvement over LTR) and for $\sim$75% of the cases for entity pairs that contain a "perfect" sentence ($\sim$3% absolute improvement over LTR).

2.6.3 Feature type analysis

Next, we analyze the impact of the feature types. Table 2.7 shows how performance varies when removing one feature type at a time from the full feature set. Relationship type features are the most important, although entity type features are important as well. This indicates that introducing features based on entities identified in the sentences and the relationship is beneficial for this task. Furthermore, the limited dependency on the source feature type indicates that our method might be able to generalize in other domains. Finally, text type features do contribute to retrieval effectiveness, although not significantly. Note that calculating the sentence features is straightforward, as none of our features requires heavy linguistic analysis.

2.6.4 Error analysis

When looking at errors made by the system, we find that some are due to the fact that entity pairs might have more than one relationship (e.g., actors that costar in movies also being partners) but the selected sentence covers only one of the relationships.[7] For example, `Liza Minnelli` is the daughter of `Judy Garland`, but they have also costarred in a movie, which is the relationship of interest. The model ranks the sentence "Liza Minnelli is the daughter of singer and actress Judy Garland…" at the top, while

[7] The annotators marked sentences that do not refer to the relationship of interest as "bad" but indicated whether they describe another relationship or not. We plan to account for such cases in future work.

the most relevant sentence is: “Judy Garland performed at the London Palladium with her then 18-year-old daughter Liza Minnelli in November 1964.”

Sentences that contain the relationship in which we are interested, but for which this cannot be directly inferred, are another source of error. Consider, for example, the following sentence, which explains director `Christopher Nolan` directed actor `Christian Bale`: “Jackman starred in the 2006 film The Prestige, directed by Christopher Nolan and costarring Christian Bale, Michael Caine, and Scarlett Johansson”. Even though the sentence contains the relationship of interest, it focuses on actor `Hugh Jackman`. The sentence “In 2004, after completing filming for The Machinist, Bale won the coveted role of Batman and his alter ego Bruce Wayne in Christopher Nolan’s Batman Begins. . . ”, in contrast, refers to the two entities and the relationship of interest directly, resulting in a higher relevance grade. Our method, however, ranks the first sentence on top, as it contains more phrases that refer to the relationship.

2.7 Conclusions and Future Work

We have presented a method for explaining relationships between knowledge graph entities with human-readable descriptions. We first extract and enrich sentences that refer to an entity pair and then rank the sentences according to how well they describe the relationship. For ranking, we use learning to rank with a diverse set of features. Evaluation on a manually annotated dataset of “people” entities shows that our method significantly outperforms state-of-the-art sentence retrieval models for this task. Experimental results also show that using relationship-dependent models is beneficial.

In future work we aim to evaluate how our method performs on entities and relationships of any type and popularity, including tail entities and miscellaneous relationships. We also want to investigate moving beyond Wikipedia and extract candidate sentences from documents that are not related to the knowledge graph, such as web pages or news articles. Employing such documents also implies an investigation into more advanced co-reference resolution methods.

Our analysis showed that sentences may cover different relationships between entities or different aspects of a single relationship—we aim to account for such cases in follow-up work. Furthermore, sentences may contain unnecessary information for explaining the relation of interest between two entities. Especially when we want to show the obtained results to end users, we may need to apply further processing of the sentences to improve their quality and readability. We would like to explore sentence compression techniques to address this. Finally, relationships between entities have an inherit temporal nature and we aim to explore ways to explain entity relationships and their changes over time.

In this chapter, we studied the task of retrieving existing KG fact descriptions (explaining entity relationships). In the next chapter, we study how to generate such descriptions instead of retrieving existing ones.

3 Generating Knowledge Graph Fact Descriptions

In the previous chapter, we studied how to retrieve existing KG fact descriptions. However, a scenario where a description for a KG fact does not exist in the underlying text corpus is not unlikely. Therefore, in this chapter, we aim to answer **RQ2**: Given a KG fact, can we automatically generate a textual description of the fact in the absence of an existing description? As in the previous chapter, we use the term "entity relationship" to refer to a KG fact.

3.1 Introduction

Results displayed on a modern search engine result page (SERP) are sourced from multiple, heterogeneous sources. For so-called organic results it has been known for a long time that result snippets, i.e., brief descriptions explaining the result item and its relation to the query, positively influence the user experience [176]. In this chapter, we focus on generating descriptions for results sourced from another important ingredient of modern SERPs: knowledge graphs. Knowledge graphs (KGs) contain information about entities and their relationships. A large and diverse set of search applications utilize KGs to improve the user experience. For instance, web search engines try to identify KG entities in queries and augment their result pages with knowledge graph panels that provide contextual entity information [22, 106]. Such panels usually focus on a single entity and may include attributes of the entity and other, related entities.

Entities can be connected with more than one relationship in a KG, however. For example, two actors might have appeared in the same film, be born in the same country and also be partners. Recent work has focused on finding relationships between a pair of entities and ranking the relationships by a predefined relevance criterion [61]. When using relationships in real-world search applications, with SERPs being the prime example, a crucial problem is that they are typically represented in a formal manner that is not suitable to present to an end user. Instead, human-readable descriptions that verbalize and provide context about entity relationships are more natural to use [66].

This chapter was published as [186].

They can be used, e.g., for entity recommendations [21] or for KG-based timeline generation [9].

Descriptions of KG relationships themselves are usually not included in large-scale knowledge graphs and previous work on automatically generating such descriptions has either relied on hand-crafted templates [9] or on external text corpora [185]. The main limitations of the former are that manually creating these templates is expensive, not generalizable, and thus it does not scale well. The latter approach is limited as the underlying text corpus may not contain descriptions for all certain relationship instances; it will not produce meaningful results for instances that do not appear in the text corpus.

We propose a method that overcomes these limitations by automatically generating descriptions of KG entity relationships. Since there exist textual descriptions of a certain relationship for some relationship instances, we aim to use these descriptions to learn how the relationship is generally expressed in text and use this information to generate descriptions for other instances of the same relationship. Existing relationship descriptions are usually complex and tailored to the entities they discuss. Also, it is likely that the KG does not contain all the information included in a description. For example, the KG might not contain any information about the second part of the following sentence: "*Catherine Zeta-Jones starred in the romantic comedy The Rebound, in which she played a 40-year-old mother of two ...*". Nevertheless, descriptions of the same relationship share patterns that are specific to that relationship. Therefore, we first create sentence templates for a certain relationship and then, for a new relationship instance, we select appropriate templates, which we formulate as a ranking problem, and fill them with the appropriate entities to generate a description.

We propose a method that generates descriptions of entity relationships for a relationship instance given a knowledge graph and a set of relationship instances coupled with their descriptions; we evaluate this method using an automatic and manual evaluation method, and release the datasets used to the community.[1] We show that we generate contextually rich relationship descriptions that are meant to be valid under the KG closed-world assumption. Moreover, our template-based method is naturally robust against KG incompleteness, since in the case of lack of contextual information about the relationship instance, it can still generate a basic description.

3.2 Related Work

Web search engine result pages (SERPs) can be augmented with information about the query and the documents from KGs in order to improve the user experience [106]. Also, SERPs can be augmented with textual descriptions and/or summaries with a prominent example being snippet generation for web search [176, 178]. Closest to our setting, relationship descriptions have been studied in the context of providing evidence for entity recommendation for web search [185] and timeline generation for knowledge base entities [9]. Our task, generating a description of a relationship instance given a KG, is similar to event headline generation, where the task is to generate a short sentence that summarizes a specific event. Similar to our templates, the headline patterns constructed in [138] consist of words and entity slots. Our method differs

[1] `https://github.com/nickvosk/ecir2017-gder-dataset/`

Table 3.1: Glossary.

Symbol	Description
$\mathcal{K}$	knowledge graph
$\mathcal{E}$	set of entities
$\mathcal{P}$	set of predicates
$\langle s, p, o \rangle$	knowledge graph triple with $s, o \in \mathcal{E}$ and $p \in \mathcal{P}$
v	word in vocabulary $\mathcal{V}$
a	sentence
r_i	relationship instance of relationship r
T_r	set of templates $t \in T_r$ for relationship r
R_t	set of relationship instances that support the template t
X	set of pairs $\langle r_{i'}, y' \rangle$, where y' is a textual description (a single sentence)
C	mapping from an entity to an entity cluster
K	entity dependency graph of a sentence
G	compression graph
P	set of paths in G

however, since relationships are more general than events and we thus have to deal with ambiguity at generation time when selecting which template matches a relationship instance.

Our task is also similar to concept-to-text generation, where the task is to generate a textual description given a set of database records [148]. In this context, our task is most closely related to [99, 159]. Saldanha et al. [159] use a template-based approach for generating company descriptions from Freebase. They construct sentence templates by replacing the entities in existing sentences by the Freebase relation of the entity to the company (e.g., $\langle company \rangle$ was founded by $\langle founder \rangle$). They add a preprocessing step where they remove phrases from the sentence that contain entities that are not connected to the company directly. At generation time, the authors replace the entity slots with the appropriate entities. Lebret et al. [99] propose a neural model to generate the first sentence of a person's biography in Wikipedia conditioned on Wikipedia infoboxes. Our setting is different from these papers since our generated descriptions are neither restricted to having entities that are directly connected to the subject entity in a KG nor need they be contained in a Wikipedia infobox.

3.3 Problem Definition

In this section we formally define the task of generating descriptions of entity relationships. Table 3.1 lists the main notation we use in this chapter.

3.3.1 Prelimilaries

Let $\mathcal{E}$ be a set of entities and $\mathcal{P}$ a set of predicates. A *knowledge graph* $\mathcal{K}$ is a set of triples $\langle s, p, o \rangle$, where $s, o \in \mathcal{E}$ and $p \in \mathcal{P}$. We follow the closed-world assumption for

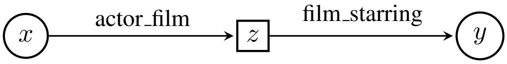


Figure 3.1: Graphical representation of the logical form of the $\mathit{starsInFilm}$ relationship. Lambda variables are shown in circles and existential variables in rectangles.

$\mathcal{K}$ and use Freebase as our knowledge graph [23, 128]. A *sentence* a is a sequence of words $[v_1, \dots, v_n]$, where each $v_i \in a$ is also in $\mathcal{V}$. Non-overlapping sub-sequences of a might refer to a single entity $e \in \mathcal{E}$.

A *relationship* r is a logical form in λ-calculus that consists of two lambda variables (x and y), at least one predicate, and zero or one existential variables [208]. Lambda variables can be substituted with Freebase entities, excluding compound value type (CVT) entities.[2] Existential variables, on the other hand, can be substituted with Freebase entities, including CVT entities. For example, the logical form of the relationship $\mathit{starsInFilm}$ is $\lambda x.\lambda y.\exists z.actor\_film(x, z) \wedge film\_starring(z, y)$. Figure 3.1 shows the equivalent graphical representation of this relationship.

A pair $r_i = r\langle s, o\rangle$ is a *relationship instance* of r for entities $s, o \in \mathcal{E}$ if by substituting $x = s$ and $y = o$ in r and by executing the resulting logical form in the knowledge graph $\mathcal{K}$ we get at least one result. For example, $\mathit{starsInFilm}(\mathit{BradPitt}, \mathit{Troy})$ is a relationship instance of the $\mathit{starsInFilm}$ relationship.

3.3.2 Task definition

We assume that a relationship instance r_i can be expressed with a human-readable description (such as a single sentence) that contains mentions of both s and o and possibly other entities which may provide contextual information for the relationship r or the entities s and o. The task we address in this chapter is to generate such a textual description y of the relationship instance r_i given the KG. For this we leverage a set of pairs X, where each $x \in X$ is a pair of $r_{i'}$ and y', and y' is the description of $r_{i'}$. We describe how we obtain this set in Section 3.5.

We aim to generate descriptions that are valid (expressing a relationship that can be found in the knowledge graph under the closed-world assumption), natural (grammatically correct), and informative, i.e., not just replicating the formal relationship but providing additional contextual information where possible.

We conclude our task definition with an example. Assume that we are given the relationship instance $\mathit{starsInFilm}(\mathit{BradPitt}, Troy)$. A possible description of this relationship instance is the following: “Brad Pitt appeared in the American epic adventure film Troy.” This description not only contains mentions of the entities of the relationship instance and a verbalization of the relationship (“appeared in”), but also mentions of other entities that provide additional context. In particular, it contains mentions of `Troy`’s type (`Film`), its genres (`Epic, Adventure`), and its country of origin.

[2] CVT entities are special entities in Freebase that are used to model attributes of relationships (e.g., date of marriage).

Table 3.2: Additional surface forms per entity type.

Entity type	**Surface form**
Person	“he” or “she”, person’s surname
Film	“the film”
Music album	“the album”
Music composition	“the song”, “the track”

3.4 Generating Textual Descriptions

In this section we detail our method which consists of three main steps. First, we enrich the description y' for each pair $\langle r_{i'}, y' \rangle \in X$ with additional entities from the KG (Section 3.4.1). Second, we use $\mathcal{K}$ and the set X to create a set of sentence templates T_r for the relationship r (Section 3.4.2). Third, given a new relationship instance, we use T_r and $\mathcal{K}$ to generate a description (Section 3.4.3).

3.4.1 Enriching the textual descriptions

In this step we perform entity linking to enrich the description y' for each pair $\langle r_{i'}, y' \rangle \in X$ with additional entities from the KG. This is done in order to facilitate the template creation step (Section 3.4.2). Each y' is a sentence that is about an entity $e \in \mathcal{E}$ and in the context of this chapter we obtain these sentences from Wikipedia as our KG provides explicit links to Wikipedia articles. Although Wikipedia articles already contain explicit links to other articles and thus entities, these links are quite sparse. Therefore, we apply an algorithm for entity linking similar to [185].

Since y' originates from a Wikipedia article that is about a specific entity, we restrict the *candidate entities* (i.e., the entities that we consider adding to enrich y') to e itself, the in-links and out-links of the article of e in the Wikipedia structure, and the one-hop and two-hop neighbors of e in the KG. We infer the *surface forms* of each entity using the Wikipedia link structure, as is common in entity linking [118], and we also use the aliases of each entity provided by the KG.[3] In order to increase coverage for e, we enhance the set of surface forms of entity e using the rules in Table 3.2.

We iterate over the n-grams of the sentence that are not yet linked to an entity in decreasing order of length; if the n-gram matches a surface form of a candidate entity, we *link* the n-gram to the entity. If multiple entity candidates exist for a surface form, we rank the candidate entities by the number of entity neighbors they have in the sentence and select the top-ranked entity. Because of the very restricted set of candidate entities, the linking is usually unambiguous (with only one entity candidate per surface form).[4]

[3] We tag the sentences with POS tags and ignore unigram surface forms that are verbs.

[4] A manual evaluation of this algorithm on a held-out, random sample of 100 sentences in our dataset revealed an average of 93% precision and 85% recall per sentence.

Algorithm 1 Template creation

```
Require: A set $X$, the knowledge graph $\mathcal{K}$
Ensure: A set of templates $T_r$
 1: $X' \leftarrow []$
 2: for $\langle r_{i'}, y' \rangle \in X$ do
 3:     $K \leftarrow$ BUILDENTITYDEPENDENCYGRAPH($y', \mathcal{K}$)
 4:     $X'.append(\langle r_{i'}, y', K \rangle)$
 5: $C \leftarrow$ CLUSTERENTITIES($X'$)
 6: $G \leftarrow$ BUILDCOMPRESSIONGRAPH($X', C$)
 7: $P \leftarrow$ FINDVALIDPATHS($G$)
 8: $T_r \leftarrow \{\}$
 9: for $p \in P$ do
10:     $t \leftarrow$ CONSTRUCTTEMPLATE($p, G, X'$)
11:     if $t \neq NULL$ then
12:         $T_r.add(t)$
```

3.4.2 Creating sentence templates

In this step, we create a set of templates T_r for a relationship r using the KG and the set of $\langle r_{i'}, y' \rangle$ pairs. The templates in T_r will be used in the next step to generate a novel description for the relationship instance r_i.

A *sentence template* t is a tuple (k, l, R_t), where (i) $k = [u_1 u_2 \dots u_n]$ is a sequence, such that $\forall u_i \in l : u_i \in \mathcal{V} \cup \mathcal{E}_t$, (ii) l is a logical form in λ-calculus that consists of all the lambda variables in $\mathcal{E}_t$, at least one predicate and zero or more existential variables, and (iii) R_t is a set of relationship instances that support t.

The procedure we follow is outlined in Algorithm 1. First, we augment each $\langle r_{i'}, y' \rangle$ pair with an entity dependency graph K in order to capture dependencies between entities in a sentence (lines 1–4). Next, we build a mapping C that maps each entity in each sentence to a single cluster id (line 5). This is done in order to facilitate the detection of useful patterns in the sentences since each sentence describes a relationship for a particular entity pair. Then, we build a compression graph G (line 6) and use it to find valid paths P (line 7). Finally, for each path $p \in P$, we construct a template t and add it to the set of templates (lines 8–12). We now describe each procedure in Algorithm 1.

BUILDENTITYDEPENDENCYGRAPH(.) In order to build the graph K for a sentence y', we retrieve all paths between each pair of entities mentioned in y' from the KG and add them to K. We only consider 1-hop paths and 2-hop paths that pass through a CVT entity. Figure 3.2 shows the entity dependency graph for an example sentence.

CLUSTERENTITIES(.) In order to obtain C, we consider all $x' = \langle r_{i'}, y', K \rangle \in X'$ and map two entities in the same cluster if they share at least one incoming or outgoing edge label in their corresponding entity dependency graph K. For example, in the *starsInFilm* relationship, this procedure will create separate clusters for persons, films, dates and CVT entities.

BUILDCOMPRESSIONGRAPH(.) In this step, we build a compression graph $G =$

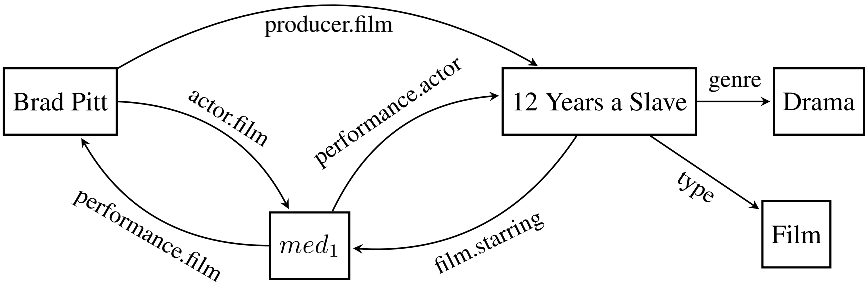


Figure 3.2: Entity dependency graph for the sentence "Brad Pitt appeared in the drama film 12 Years a Slave". Nodes represent entities and edge labels represent predicates (med_1 is a CVT entity).

(V, E) using the sentence y' of each $\langle r_{i'}, y', K\rangle \in X'$. V is a set of nodes and E is a set of edges. We follow a similar procedure to [63], in which each node holds a list of $\langle sid, pid\rangle$ pairs, where sid is a sentence id and pid is the index of the word/entity in the sentence. In our case a node can be a word or an entity cluster. We map two words onto the same node if they have the same lowercase form and the same POS tag. We map two entities on the same node if they have the same cluster id.

FINDVALIDPATHS(.) In order to find valid paths in the graph G, we set all the entity cluster nodes as valid start/end nodes and traverse G to find a set of paths P from a start to an end node. In order to build templates that are natural we enforce the following constraints for the paths in P: (i) the path must contain a verb and (ii) the path must have been seen as a complete sentence at least once in the input sentences. For example, given the following sentences (the corresponding cluster id per entity are listed in brackets):

- y'_1: "Bruce_Willis[c_1] appeared in Moonrise_Kingdom[c_2]"
- y'_2: "Liam_Neeson[c_1] appeared in the action[c_3] film[c_4] Taken[c_2]"
- y'_3: "Brad_Pitt[c_1] appeared in the drama[c_3] film[c_4] 12_Years_a_Slave[c_2]"

we obtain the following valid paths by traversing the graph:

- p_1: "c_1 appeared in c_2"
- p_2: "c_1 appeared in the c_3 c_4 c_2"

CONSTRUCTTEMPLATE(.) Algorithm 2 outlines the procedure for constructing a template t from a path p. First, for each $\langle r_{i'}, y', K\rangle \in X'$, we check whether y' is a (possibly non-continuous) subsequence h of path p by using the positional information of each node in p from G.[5] If it is, we check whether h contains links to both the subject and the object of the relationship instance $r_{i'}$. If it does, we store the entity dependency graph and the relationship instance. Next, if the number of instances is less

[5] For example, the path p_1 is a subsequence of y'_2.

Algorithm 2 ConstructTemplate(.)

```
Require: A path $p$, the compression graph $G$, a set $X'$, parameters $\alpha, \beta$
Ensure: A template $t$
$D_g \leftarrow []$                                  ▷ entity dependency graphs
$R_t \leftarrow []$                                  ▷ relationship instances that support the template
for $\langle r_{i'}, y', K \rangle \in X'$ do
    if IsSubsequence($p, y', G$) then
        $h \leftarrow$ GetSubsequence($p, y', G$)     ▷ get the actual subsequence
        $\langle s, o \rangle \leftarrow r_{i'}$                 ▷ subject/object of the relationship instance
        if ContainsLink($h, s$) and ContainsLink($h, o$) then
            $D_g.append(K)$
            $R_t.append(r_{i'})$
if $|R_t| < \alpha$ then                             ▷ too few relationship instances
    return $NULL$
$l \leftarrow$ BuildLogicalForm($D_g, \beta$)        ▷ aggregate the entity dependency graphs
$k \leftarrow$ ReplaceClusterIdsWithVariables($p$)
$t = (k, l, R_t)$
```

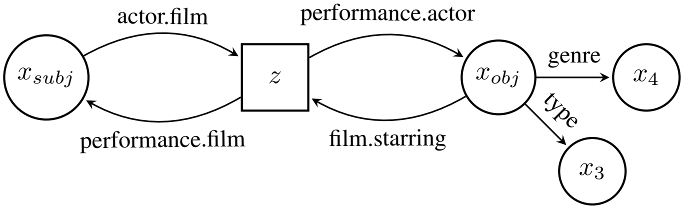


Figure 3.3: Logical form of the template constructed using p_2 and y'_1, y'_2, y'_3 (with their corresponding relationship instances). $k =$"x_{subj} appeared in the x_3 x_4 x_{obj}". Lambda variables are shown in circles and existential variables in rectangles.

than a parameter α, we consider the template to be invalid. Subsequently, we build the logical form l by aggregating the entity dependency graphs D_g. Entity nodes that were part of the path p become lambda variables (nodes constructed from subject and object entities have special identifiers). Entity nodes that were not part of the path p (CVT entities) become existential variables. We ignore edges appearing in less than $|D_g| \cdot \beta$ entity dependency graphs. Lastly, we replace the cluster ids in p with the corresponding lambda variables to obtain a sequence k.

Figure 3.3 shows the logical form of a template constructed using the example sentences y'_1, y'_2 and y'_3 and their corresponding instances in graphical form ($\beta = 0.5$). Note that the edge "producer.film" has been eliminated since it only appears in one out of the three instances.

3.4.3 Generating the description

In this step we generate a novel description for a relationship instance r_i using the set of templates T_r and the knowledge graph $\mathcal{K}$. This comes down to selecting the template from T_r that best describes the relationship instance r_i and filling it with the appropriate entities.

The procedure is as follows. First, we rank the templates in T_r for the relationship instance using a scoring function $f(r_i, t)$. Subsequently, for each template $t = (k, l, R_t)$ we replace the subject and object lambda variables in l to obtain $l' = l[x_{subj} = s, x_{obj} = o]$. We then query the knowledge graph $\mathcal{K}$ using l' and if at least one instantiation of l' exists, we randomly pick one and replace all the entity variables in k with the entity names to generate the description y, otherwise we proceed to the next template. As an example, assume we are given the instance $r_i = \mathit{starsInFilm}(\mathit{Ryan\_Reynolds}, \mathit{Deadpool})$ and we consider the template shown in Figure 3.3. A possible instantiation of the template for this relationship instance will result in the description “Ryan Reynolds appeared in the comedy film Deadpool”.[6]

The template scoring function $f(r_i, t)$ returns a score for a relationship instance r_i and template t. As we want to generate descriptions that are valid under the closed-world assumption of the KG, we promote templates that are semantically closest to the relationship instance. For a new relationship instance r_i we extract binary features for each entity in the r_i. Recall that r_i has two or more entities (subject s, object o and possibly a CVT entity z). For each entity e of r_i, we extract all triples $\langle e, p, e' \rangle$ from the KG $\mathcal{K}$. We restrict the feature space by discriminating between entity attributes and entity relations depending on the predicate p as in [107]. If the predicate p is an attribute (e.g., “gender”), we use the complete triple as a feature (e.g. $\langle s, \mathit{gender}, \mathit{female} \rangle$). If the predicate p is a relation (e.g., “date_of_death”), we only keep the subject and the predicate of the triple as a feature (e.g., $\langle e, \mathit{person.date\_of\_death} \rangle$). We also add a count feature for the relation predicates (e.g., $\langle s, \mathit{person.children}, 2 \rangle$, i.e., a person has two children). We denote the resulting binary vector for r_i as $\mathit{vec}(r_i)$. We obtain a vector $\mathit{vec}(t)$ for template t by summing the vectors of all the instances R_t of t. We also compute a vector $\mathit{vec\_tfidf}(t)$ that is a TF.IDF weighted vector of $\mathit{vec}(t)$, where IDF is calculated at the template level. Based on these ingredients, we define two scoring functions:

- **Cosine** Calculates the cosine similarity between vectors $vec(r_i)$ and $vec\_tfidf(t)$.
- **Supervised** Learns a scoring function using a supervised learning to rank algorithm. We treat r_i as a “query” and t as a “document.”

We create training data for the supervised algorithm as follows. Recall that each r_i is coupled with a description y'. For each r_i, we assign a relevance label of 3 for templates that best match y (measured by the number of entities) and a relevance label of 2 for the rest of the templates that match y. In order to create “negative” training data, we sample templates that are dissimilar to the ones that match y in the following way. First, we calculate the average vector of all the templates that match y and build a distribution of

[6] Note that there might be multiple instantiations (e.g., Deadpool is also a science fiction film) and selecting the optimal one depends on the application—we leave this for future work.

templates based on the cosine distance from the average vector to each of the templates in T_r (excluding the ones that match y). Lastly, we sample at most the number of matching templates from the resulting distribution and assign them a relevance label of 1 (we ignore templates that have a cosine similarity to the average vector greater than 0.9). For the supervised model we use the following features: each element/value pair in $vec(r_i)$, the cosine similarity between vectors $vec(r_i)$ and $vec\_tfidf(t)$, the words in t, the number of entities in t and the size of R_t. We use LambdaMART [199] as the learning algorithm and optimize for NDCG@1.[7]

3.5 Experimental Setup

In this section we describe the experimental setup we designed to answer **RQ2**.

3.5.1 Datasets

We use an English Wikipedia dump dated 5 February 2015 as our document corpus. We perform sentence splitting and POS tagging using the Stanford CoreNLP toolkit. We use a subset of the last version of Freebase as our KG [23]: all the triples in the people, film and music domains, as these are well-represented in Freebase.

In order to create an evaluation dataset for our task, we first need a set of KG relationships. We rank the predicates in each domain by the number of instances and keep the 10 top-ranked predicates. We exclude trivial predicates such as “dateOfDeath”. We then use the predicates to manually construct the logical forms of the relationships (see Figure 3.1 for an example). Second, we need a set of $\langle r_{i'}, y' \rangle$ pairs for each relationship r, where $r_{i'} = r\langle s', o' \rangle$ is an instance of relationship r, s' and o' are entities and y' is a description of $r_{i'}$. To this end, for each relationship r, we randomly sample 12000 relationship instances from the KG. For each relationship instance $r_{i'}$, we pick the first sentence in the Wikipedia article of the subject entity s' that contains links to both s' and o'. If such a sentence does not exist, we proceed to the next instance. We manually inspected a subset of the sentences selected with this heuristic and the quality of the selected sentences was relatively good. Our final dataset contains 10 relationships and 90058 $\langle r_{i'}, y' \rangle$ instances in total and 8187 instances on average per relationship. We randomly select 80% of each relationship sub-dataset for training and 20% for testing.

3.5.2 Evaluation metrics

We perform two types of evaluation: automatic and manual. For automatic evaluation we use METEOR [97], ROUGE-L [105] and BLEU-4 [133] as metrics. METEOR was originally proposed in the context of machine translation but has also been used in a task similar to ours [159]. ROUGE is a standard metric in summarization and BLEU is widely used in machine translation and generation. As is common in text generation [94], we also employ manual evaluation. We ask human annotators to annotate each output sentence on three dimensions: validity under the KG closed-world

[7] For this method we use 20% of the training data as validation data. The same test data is used for all methods.

Table 3.3: Automatic evaluation results, averaged per relationship.

Method	**BLEU**	**METEOR**	**ROUGE**
Random	1.14	16.56	24.13
Most-freq	0.13	13.99	21.96
Cosine	$1.76^{\blacktriangle}$	17.37	$25.84^{\blacktriangle}$
Supervised	$\mathbf{2.14}^{\blacktriangle}$	$\mathbf{19.18}^{\blacktriangle}$	$\mathbf{26.54}^{\blacktriangle}$

assumption (0 or 1), informativeness (1–5) and grammaticality (1–5). One human annotator (not one of the authors) annotated 11 generated sentences per relationship per system (440 sentences in total).

3.5.3 Compared approaches

We compare 4 variations of our method. The variations differ in the way they rank templates for a given relationship instance. The first variation (*Random*) ranks the templates randomly. The second (*Most-freq*) ranks templates by the number of relationship instances that support the template. The third (*Cosine*) ranks templates based on the cosine similarity between the vectors of the relationship instance and the template (Section 3.4.3). The fourth (*Supervised*) ranks templates using a learning to rank model (Section 3.4.3), for which we use LambdaMART with the default number of trees (1000). We set $\alpha = 20$ and $\beta = 0.5$ (Section 3.4.3). We depict a significant improvement in performance over *Random* with $^{\blacktriangle}$ (paired two-tailed t-test, $p < 0.05$).

3.6 Results

In this section we describe our experimental results. We compare all methods discussed previously, using the automatic and manual setups, respectively.

3.6.1 Automatic evaluation

Table 3.3 shows the automatic evaluation results. We observe that *Supervised* and *Cosine* outperform *Random* and *Most-freq* on all metrics. This is expected since the former two try to capture the semantic similarity between a relationship instance and a template. Although *Supervised* consistently outperforms *Cosine*, the differences between *Cosine* and *Supervised* are not significant.

We also observe that the scores for the automatic measures are relatively low. This is because of two reasons: (i) we generally generate much shorter sentences than the reference sentence as not all information that appears in the reference sentence is represented in the KG, and (ii) since the reference sentences are extracted automatically, some of the reference sentences describe a minor aspect of the relationship or do not discuss the relationship at all.

Table 3.4: Manual evaluation results, averaged per relationship.

Method	Validity	Informativeness	Grammaticality
Random	0.4545	1.98	3.67
Most-freq	0.5000	1.60	3.62
Cosine	0.5636▲	2.05	**4.00**
Supervised	**0.5818**▲	**2.18**▲	3.90

3.6.2 Manual evaluation

Table 3.4 shows the results for manual evaluation. The results follow a similar trend as in the automatic evaluation; *Supervised* and *Cosine* outperform *Random* and *Most-freq* on all metrics. *Supervised* significantly outperforms *Random* in terms of validity and informativeness. The differences between *Cosine* and *Supervised* are not significant.

3.6.3 Analysis

We have also examined specific examples and identify cases where the best performing approach (*Supervised*) succeeds or fails. In terms of validity, it succeeds in matching attributes of the relationship instance and the template. E.g., in the context of the relationship *parentOf*, it correctly figures out what the genders of the entities are and the semantically valid expression of the relationship between them, often better than *Cosine*, as illustrated by the following example:

(*Supervised*) "Emperor Francis I (1708 - 1765) was the father of Emperor Leopold II" (VALID)

(*Cosine*) "Emperor Francis I was the son of Emperor Leopold II" (INVALID)

Supervised benefits from training a model that combines multiple features such as the template words with attributes of the relationship instance to describe whether the relationship is still ongoing or not. One of the main cases where *Supervised* fails is in ranking a relationship instance in a temporal dimension with regards to other relationship instances, as illustrated by the following example for the *childOf* relationship:

"Thomas Howard was the second son of Henry Howard and Frances de Vere." (INVALID: Thomas Howard was the *first* son of Henry Howard)

The fact that our best performing approach (*Supervised*) has a relatively low validity score (0.5818) shows that there is room for improvement in capturing the semantic similarity between a relationship instance and a template.

In terms of informativeness, *Supervised* succeeds in offering contextual information about the relationship instance, such as dates, locations, occupations and film genres. The fact that informativeness scores are relatively low is because they are dependent on validity: when a generated sentence was assigned a validity of score 0, it was also assigned an informativeness score of just 1.

Grammaticality scores are high for all the systems with no significant differences. This is expected as the templates were generated using the same procedure for all the compared systems. Mainly, grammaticality is harmed when some entities in the generated sentence have the wrong surface form (e.g., ‘Britain’, ‘British’), which is not surprising as we do simple surface realization (deciding which surface form of the entity best fits with the generated sentence) and only use the entity names as surface forms.

3.7 Conclusion

We have addressed the problem of generating descriptions of entity relationships from KGs. We have introduced a method that first creates sentence templates for a specific relationship, and then, for a new relationship instance, it generates a novel description by selecting the best template and filling the template slots with the appropriate entities from the KG. We have experimented with different scoring functions for ranking templates for a relationship instance and performed an automatic and a manual evaluation.

When using information about the relationship instance and the template taken from the KG, both automatic and manual evaluation outcomes are improved. A supervised method that uses both KG features and other template features (template words, number of entities) consistently outperforms an unsupervised method on all automatic evaluation metrics and also in terms of validity and informativeness.

As to future work, our error analysis showed that we need more sophisticated modeling for capturing the semantic similarity between a relationship instance and a template, especially for capturing temporal dimensions that also involve other relationship instances. We also want to explore more sophisticated methods for selecting the correct surface form for an entity to improve grammaticality. Finally, we aim to evaluate our method on generating descriptions for less popular KG relationships.

In this chapter, we studied the task of generating KG fact (entity relationship) descriptions. In the next chapter, we move on to study how to contextualize KG facts using other, related KG facts.

4 Contextualizing Knowledge Graph Facts

In Chapter 2 and 3, we studied how retrieve and generate descriptions of knowledge graph (KG) facts. KG fact descriptions often contain mentions to other, related KG facts that are not trivial to find given the large size of KGs. In this chapter, we address **RQ3**: Can we contextualize a KG query fact by retrieving other, related KG facts?

4.1 Introduction

Knowledge graphs (KGs) have become essential for applications such as search, query understanding, recommendation and question answering because they provide a unified view of real-world entities and the facts (i.e., relationships) that hold between them [21, 22, 120, 208]. For example, KGs are increasingly being used to provide direct answers to user queries [208], or to construct so-called *entity cards* that provide useful information about the entity identified in the query. Recent work [25, 75] suggests that search engine users find entity cards useful and engage with them when they contain information that is relevant to their search task, for instance in the form of a set of recommended entities and facts that are related to the query [21]. Previous work has focused on augmenting entity cards with facts that are centered around, i.e., one-hop away from, the main entity of the query [75].

However, oftentimes a user is interested in KG facts that by definition involve more than one entity (e.g., "Who founded Microsoft?" ⟶ "Bill Gates"). In such cases, we can exploit the richness of the KG by providing query-specific additional facts that increase the user's understanding of the fact as a whole, and that are not necessarily centered around only one of the entities. Additional relevant facts for the running example would include Bill Gates' profession, Microsoft's founding date, its main industry and its co-founder Paul Allen (see Figure 4.1). In this case, Bill Gates' personal life is less relevant to the fact that he founded Microsoft.

Query-specific relevant facts can also be used in other applications to enrich the user experience. For instance, they can be used to increase the utility of KG question answering (QA) systems that currently only return a single fact as an answer to a natural language question [15, 208]. Beyond QA, systems that focus on automatically generating natural language from KG facts [99] would also benefit from query-specific

This chapter was published as [187].

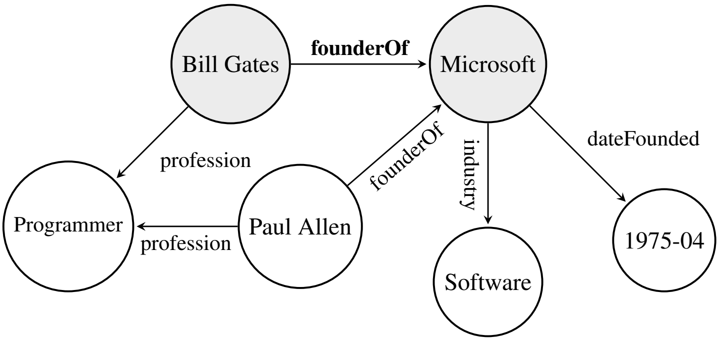


Figure 4.1: A Freebase subgraph that consists of relevant facts to the query fact $founderOf(\text{Bill Gates}, \text{Microsoft})$.

relevant facts, which can make the generated text more natural and human-like. This becomes even more important for KG facts that involve tail entities, for which natural language text might not exist for training [186].

In this chapter, we address the task of KG fact *contextualization*, that is, given a KG fact that consists of two entities and a relation that connects them, retrieve additional facts from the KG that are relevant to that fact. This task is analogous to ad-hoc retrieval: (i) the "query" is a KG fact, (ii) the "documents" are other facts in the KG that are in the neighborhood of the "query". We propose a *neural fact contextualization method* (NFCM), a method that first generates a set of candidate facts that are part of {1,2}-hop paths from the entities of the main fact. NFCM then ranks the candidate facts by how relevant they are for contextualizing the main fact. We estimate our learning to rank model using supervised data. The ranking model combines (i) features we automatically learn from data and (ii) those that represent the query-candidate facts with a set of hand-crafted features we devised or adjusted for this task. Due to the size and heterogeneous nature of KGs, i.e., the large number of entities and relationship types, we turn to distant supervision to gather training data. Using another, human-verified test collection we gauge the performance of our proposed method and compare it with several baselines. We sum up our contributions as follows.

- We introduce the task of KG fact contextualization where the goal is to, given a fact that consists of two entities and a relationship that connects them, rank other facts from a KG that are relevant to that fact.
- We propose NFCM, a method to solve KG fact contextualization using distant supervision and learning to rank. Our results show that: (i) distant supervision is an effective means for gathering training data for this task and (ii) a neural learning to rank model that is trained end-to-end outperforms several baselines on a human-curated evaluation set.
- We provide a detailed result analysis and insights into the nature of our task.

The remainder of this chapter is organized as follows. We first provide a definition of our task in Section 4.2 and then introduce our method in Section 4.3. We describe

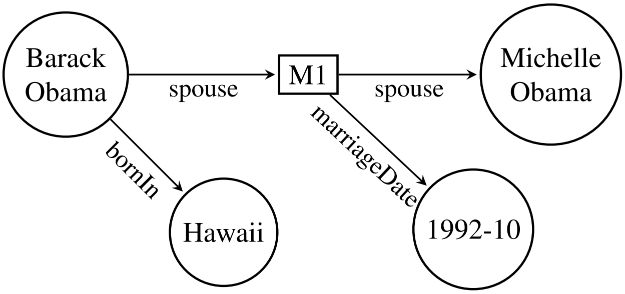


Figure 4.2: KG subgraph that consists of three facts: $\mathit{bornIn}\langle\text{Barack Obama}, \text{Hawaii}\rangle$, $\mathit{spouseOf}\langle\text{Barack Obama}, \text{Michelle Obama}\rangle$ and $\mathit{marriageDate}\langle\text{M1}, \text{1992-10}\rangle$. M1 is a CVT entity. Note that the third fact is an attribute of the second fact.

our experimental setup and detail our results and analyses in Sections 4.4 and 4.5, respectively. We conclude with an overview of related work and an outlook on future directions.

4.2 Problem Statement

In this section we provide background definitions and formally define the task of KG fact contextualization.

4.2.1 Preliminaries

Let $E = E_n \cup E_c$ be a set of entities, where E_n and E_c are disjoint sets of non-CVT and CVT entities, respectively.[1] Furthermore, let P be a set of predicates. A *knowledge graph* K is a set of triples $\langle s, p, o\rangle$, where $s, o \in E$ and $p \in P$. By viewing each triple in K as a labelled directed edge, we can interpret K as a labelled directed graph. We use Freebase as our knowledge graph [23, 128].

A path in K is a non-empty sequence $\langle s_0, p_0, t_0\rangle, \ldots, \langle s_m, p_m, t_m\rangle$ of triples from K such that $t_i = s_{i+1}$ for each $i \in 0, m-1$.

We define a *fact* as a path in K that either: (i) consists of 1 triple, $s_0 \in E$ and $t_0 \in E_n$ (i.e., s_0 may be a CVT entity), or (ii) consists of 2 triples, $s_0, t_1 \in E_n$ and $t_0 = s_1 \in E_c$ (i.e., $t_0 = s_1$ must be a CVT entity). A fact of type (i) can be an attribute of a fact of type (ii), iff they have a common CVT entity (see Figure 4.2 for an example).

Let R be a set of relationships where a *relationship* $r \in R$ is a label for a set of facts that share the same predicates but differ in at least one entity. For example, $\mathit{spouseOf}$ is the label of the fact depicted in the top part of Figure 4.2 and consists of two triples. Our definition of a relationship corresponds to direct relationships between entities, i.e., one-hop paths or two-hop paths through a CVT entity. For the remainder of this chapter, we refer to a specific fact f as $r\langle s, t\rangle$, where $r \in R$ and $s, t \in E$.

[1] Compound Value Type (CVT) entities are special entities frequently used in KGs such as Freebase and Wikidata to model fact attributes. See Figure 4.2 for an example.

Algorithm 3 Fact enumeration for a given query fact f_q.

```
Require: A query fact $f_q = r\langle s, t\rangle$
Ensure: A set of candidate facts $F$
F ← {}
for $e \in \{s, t\}$ do
    for $n \in$ GETOUTNEIGHBORS($e$) + GETINNEIGHBORS($e$) do
        $F.addAll$(GETFACTS($e, n$))
        if ISCLASSORTYPE($n$) then
            continue
        for $n_2 \in$ GETOUTNEIGHBORS($n$) do
            $F.addAll$(GETFACTS($n, n_2$))
        for $n_2 \in$ GETINNEIGHBORS($n$) do
            $F.addAll$(GETFACTS($n_2, n$))
return $F$
```

4.2.2 Task definition

Given a query fact f_q and a KG K, we aim to find a set of other, relevant facts from K. Specifically, we want to enumerate and rank a set of candidate facts $F = \{f_c : f_c \subseteq K, f_c \neq f_q\}$ based on their relevance to f_q. A candidate fact f_c is *relevant* to the query fact f_q if it provides useful and contextual information. Figure 4.1 shows an example part of our KG that is relevant to the query fact $\mathit{founderOf}\langle\text{Bill Gates, Microsoft}\rangle$. Note that a candidate fact does not have to be directly connected to both entities of the query fact to be relevant, e.g., $\mathit{profession}\langle\text{Paul Allen, Programmer}\rangle$. Similarly, a fact can be related to one or more entities in the relationship instance, e.g., $\mathit{parentOf}\langle\text{Bill Gates, Jennifer Katharine Gates}\rangle$, but not provide any context, thus being considered irrelevant.

4.3 Method

In this section we describe our proposed neural fact contextualization method (NFCM) which works in two steps. First, given a query fact f_q, we enumerate a set of candidate facts $F = \{f_c : f_c \subseteq K\}$ (see Section 4.3.1). Second, we rank the facts in F by relevance to f_q to obtain a final ranked list F' using a supervised learning to rank model (see Section 4.3.2). We describe how we use distant supervision to automatically gather the required annotations to train the supervised learning to rank model in Section 4.4.3.

4.3.1 Enumerating KG facts

In this section we describe how we obtain the set of candidate facts F from K given a query fact $f_q = r\langle s, t\rangle$. Because of the large size of real-world KGs—which can easily contain upwards of 50 million entities and 3 billion facts [134]— it is computationally infeasible to add all possible facts of K in F. Therefore, we limit F to the set of facts

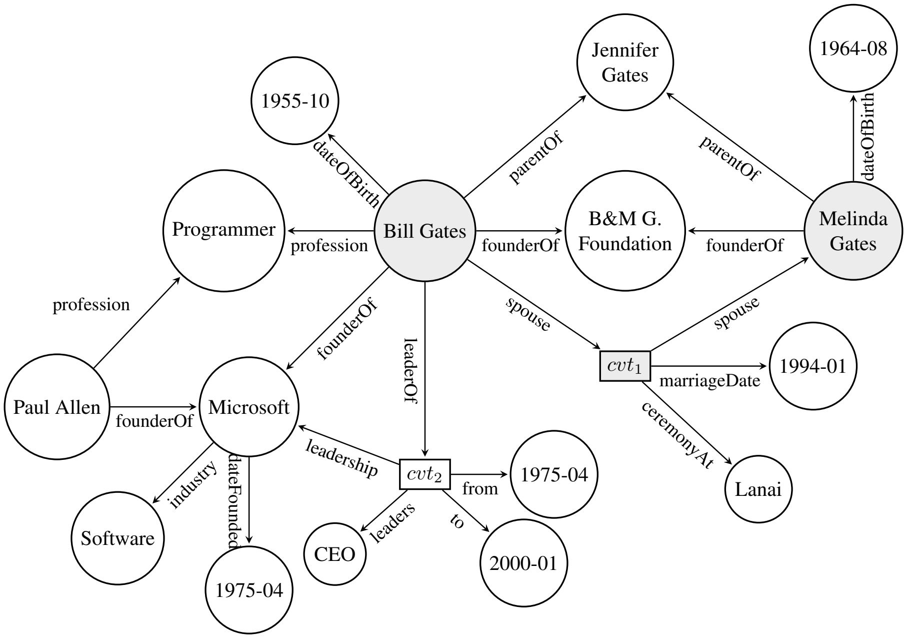


Figure 4.3: Graph with a subset of the facts that are enumerated for the query fact $spouseOf$(Bill Gates, Melinda Gates). The entities of the query fact are shaded.

that are in the broader neighborhood of the two entities s and t. Intuitively, facts that are further away from the two entities of the query fact are less likely to be relevant.

The procedure we follow is outlined in Algorithm 3. This algorithm enumerates the candidate facts for $f_q = r\langle s, t\rangle$ that are at most 2 hops away from either s or t. Three exceptions are made to this rule: (i) CVT entities are not counted as hops, (ii) we do not include f_q in F as it is trivial, and (iii) to reduce the search space, we do not expand intermediate neighbors that represent an entity class or a type (e.g., “actor”) as these can have millions of neighbors. Figure 4.3 shows an example graph with a subset of the facts that we enumerate for the query fact $spouseOf\langle$Bill Gates, Melinda Gates$\rangle$ using Algorithm 3.

4.3.2 Fact ranking

Next, we describe how we rank the set of enumerated candidate facts F with respect to their relevance to the query fact $f_q = r\langle s, t\rangle$. The overall methodology is as follows. For each candidate fact $f_c \in F$, we create a pair (f_q, f_c)—an analog to a query-document pair—and score it using a function $u : (f_q, f_c) \to [0, 1] \in R$ (higher values indicate higher relevance). We then obtain a ranked list of facts F' by sorting the facts in F based on their score.

We begin by describing the training procedure we follow and continue with the network architecture we use for learning our scoring function u.

Learning procedure We train a network that learns the scoring function $u(f_q, f_c)$ end-to-end in mini-batches using stochastic gradient descent (we define the network architecture below). We optimize the model parameters using Adam [92]. During training we minimize a pairwise loss to learn the function u, while during inference we use the learned function u to score a query-candidate fact pair (f_q, f_c). This paradigm has been shown to outperform pointwise learning methods in ranking tasks, while keeping inference efficient [49]. Each batch B consists of query-candidate fact pairs (f_q, f_c) of a single query fact f_q. For constructing B for a query fact f_q, we use all pairs (f_q, f_c) that are labeled as relevant and sample k pairs (f_q, f_c) that are labeled as irrelevant. During training, we minimize the mean pairwise squared error between all pairs of (f_q, f_c) in $B \times B$:

$$L(B, \theta) = \frac{1}{|B|} \sum_{\langle x_1, x_2 \rangle \in B \times B} ([l(x_1) - l(x_2)] - [u(x_1) - u(x_2)])^2, \tag{4.1}$$

where $x_1 = (f_q, f_{c_1})$ and $x_2 = (f_q, f_{c_2})$ are query-candidate fact pairs in the set $B \times B$, $l(x) \in \{0, 1\}$ is the relevance label of a query-candidate fact pair x, $|B|$ is the batch size, and θ are the parameters of the model which we define below.

Network architecture Figure 4.4 shows the network architecture we designed for learning the scoring function $u(f_q, f_c)$. We encode the query fact f_q in a vector $\boldsymbol{v_q}$ using an RNN. As we will explain further in that section, we do not model the entities in the facts independently due to the large number of entities; instead, we model each entity as an aggregation of its types. Therefore, instead of modeling the candidate fact f_c in isolation and losing per-entity information, we first enumerate all the paths up to two hops away from both the entities of the query fact f_q (s and t) to all the entities of the candidate fact f_c (s' and t'). Let A_s denote the set of paths from s to all the entities of f_c. Let A_t denote the set of paths from t to all the entities of f_c. For each $A \in \{A_s, A_t\}$, we first encode all the paths in A using an RNN, and then combine the resulting encoded paths using the procedure described later in this section. We denote the vectors obtained from the above procedure for A_s and A_t as $\boldsymbol{v}_{as}$ and $\boldsymbol{v}_{at}$, respectively. Then we obtain a vector $\boldsymbol{v}_a = [\boldsymbol{v}_{as}, \boldsymbol{v}_{at}]$, where $[\cdot, \cdot]$ denotes the concatenation operation (middle part of Figure 4.4). Note that we use the same RNN parameters for all the above operations. To further inform the scoring function, we design a set of hand-crafted features $\boldsymbol{x}$ (right-most part of Figure 4.4). We detail the hand-crafted features later in this section.

Finally, MLP-o($[\boldsymbol{v}_q, \boldsymbol{v}_a, \boldsymbol{x}]$) is a multi-layer perceptron with α hidden layers of dimension β and one output layer that outputs $u(f_q, f_c)$. We use a ReLU activation function in the hidden layers and a sigmoid activation function in the output layer. We vary the number of layers to capture non-linear interactions between the features in $\boldsymbol{v}_q$, $\boldsymbol{v}_a$, and $\boldsymbol{x}$.

The remainder of this section describes how we encode a single fact, how we combine the representations of a set of facts, and, finally, the hand-crafted features.

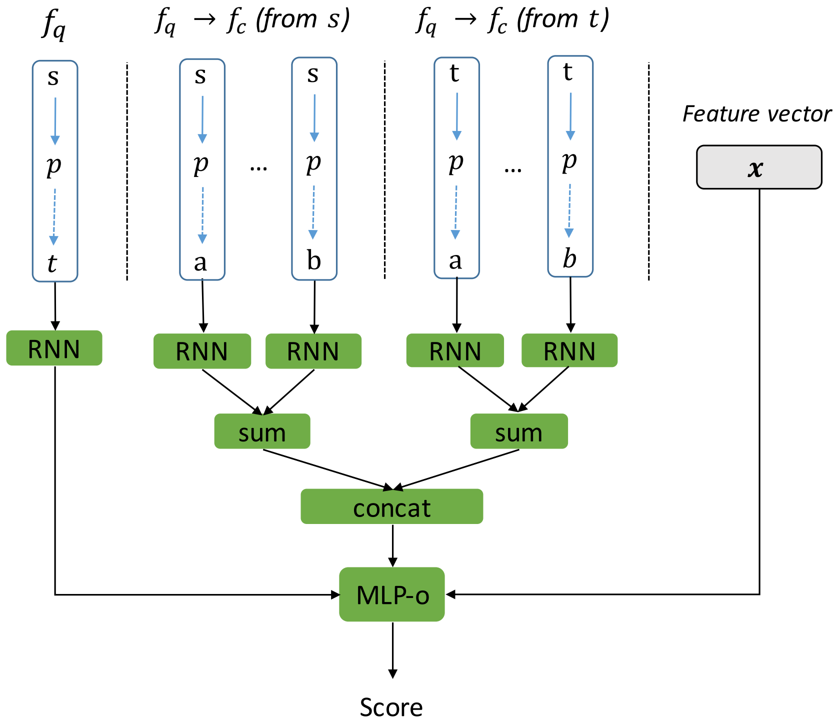


Figure 4.4: Network architecture that learns a scoring function $u(f_q, f_c)$. Given a query fact $f_q = r\langle \mathrm{s}, \mathrm{t}\rangle$ and a candidate fact $f_c = r'\langle \mathrm{a}, \mathrm{b}\rangle$ it outputs a score $u(f_q, f_c)$. "$f_q \rightarrow f_c$ (from e)" is a label for the paths that start from an entity e of the query fact (either s or t) and end at an entity e' of the candidate fact f_c. Note that p is a variable in this figure, i.e., it might refer to different predicates.

Encoding a single fact

Recall from Section 4.2.1 that a fact f is a path in the KG. In order to model paths we turn to neural representation learning. More specifically, since paths are sequential by nature we employ recurrent neural networks (RNNs) to encode them in a single vector [48, 71]. This type of modeling has proven successful in predicting missing links in KGs [48]. One restriction that we have in modeling such paths is the very large number of entities ($\sim$ 1.5 million entities in our dataset) and, since learning an embedding for such large numbers of entities requires prohibitively large amounts of memory and data, we represent each entity using an aggregation of its types [48]. Formally, let $\boldsymbol{W}_z$ denote a $|Z| \times d_z$ matrix, where each row is an embedding of an entity type z, $|Z|$ is the number of entity types in our dataset and d_z is the entity type embedding dimension. Let $\boldsymbol{W}_p$ denote a $|P| \times d_p$ matrix, where each row is an embedding of a predicate p, $|P|$ is the number of predicates in our dataset, and d_p is the predicate embedding dimension. In order to model *inverse* predicates in paths (e.g.,

Table 4.1: Notation

Name	Description	Definition
$\mathit{NumTriples}$	Number of triples in K	$\lvert\{\langle s,p,t\rangle : \langle s,p,t\rangle \in K\}\rvert$
$\mathit{TriplesPred}(p)$	Set of triples that have predicate p	$\{\langle s,p',t\rangle : \langle s,p',t\rangle \in K, p' = p\}$
$\mathit{TriplesEnt}(e)$	Set of triples that have entity e	$\{\langle s,p,t\rangle : \langle s,p,t\rangle \in K, s = e \vee t = e\}$
$\mathit{TriplesSubj}(e)$	Set of triples that have entity e as subject	$\{\langle s,p,t\rangle : \langle s,p,t\rangle \in K, s = e\}$
$\mathit{TriplesObj}(e)$	Set of triples that have entity e as object	$\{\langle s,p,t\rangle : \langle s,p,t\rangle \in K, t = e\}$
$\mathit{UniqEnt}(T)$	The unique set of entities in a set of triples T	$\bigcup\{\{s,t\} : \langle s,p,t\rangle \in T\}$
$\mathit{Types}(e)$	The set of types of entity e	$\{z : \langle e, \mathit{type}, z\rangle \in K\}$
$\mathit{Entities}(f)$	The set of entities of fact f	$\bigcup\{\{s,t\} : \forall\langle s,p,t\rangle \in f\}$
$\mathit{Preds}(f)$	The set of predicates of fact f	$\{p : \langle s,p,t\rangle \in f\}$

Microsoft → $\mathit{founderOf}^{-1}$ → Paul Allen), we also define a $|P| \times d_p$ matrix $\boldsymbol{W}_{p_i}$, which corresponds to embeddings of the inverse of each predicate [71].

The procedure we follow for modeling a fact f is as follows. For simplicity in the notation, in this Section we denote a path as a sequence of alternate entities and predicates $[s_0, p_0, \dots t_m]$, instead of a sequence of triples as defined in Section 4.2.1. For each entity $e \in f$, we first retrieve the types of e in K. From these, we only keep the 7 most frequent types in K, which we denote as Z_e [48]. We then project each $z \in Z_e$ to its corresponding type embedding $\boldsymbol{w}_z \in \boldsymbol{W}_z$ and perform element-wise sum on these embeddings to obtain an embedding w_e for entity e. We project each predicate $p \in f$ to its corresponding embedding $\boldsymbol{w}_p$ ($\boldsymbol{w}_p \in \boldsymbol{W}_{p_i}$ if p is inverse, $\boldsymbol{w}_p \in \boldsymbol{W}_p$ otherwise).

The resulting projected sequence $X_f = [\boldsymbol{w}_{s_0}, \boldsymbol{w}_{p_0}, \dots, \boldsymbol{w}_{t_m}]$ is passed to a unidirectional recurrent neural network (RNN). The RNN has a sequence of hidden states $[\boldsymbol{h}_1, \boldsymbol{h}_2, \dots, \boldsymbol{h}_n]$, where $\boldsymbol{h}_i = tanh(\boldsymbol{W_{hh}}\boldsymbol{h}_{i-1} + \boldsymbol{W_{xh}}\boldsymbol{x}_i)$, and $\boldsymbol{W_{hh}}$ and $\boldsymbol{W_{xh}}$ are the parameters of the RNN. The RNN is initialized with zero state values. We use the last state of the RNN $\boldsymbol{h}_n$ as the representation of the fact f.

Combining a set of facts

We obtain the representation of the set of encoded facts using element-wise summation of the encoded facts (vectors). We leave more elaborate methods for combining facts such as attention mechanisms [12, 48] for future work.

Hand-crafted features

Here, we detail the hand-crafted features $\boldsymbol{x}$ we designed or adjusted for this task. Table 4.1 lists the notation we use. We generate features based on feature templates that are divided into three groups: (i) those that give us a sense of *importance* of a fact, (ii) those that give us a sense of *relevance* of (f_q, f_c), and (iii) a set of miscellaneous features. Note that we use log-computations to avoid underflows.

(i) Fact importance This group of feature templates give us a sense on how important a fact f is when taking statistics of the knowledge graph K into account at a global level. Note that we calculate these features for both facts f_q and f_c. The first of these feature templates measures *normalized predicate frequency* of each predicate p that participates in fact f (we also include the minimum, maximum and average value for each fact as metafeatures [24]). This is defined as the ratio of the size of the set of triples that have predicate p in the KG to the total number of triples:

$$\mathit{PredFreq}(p) = \frac{|\mathit{TriplesPred}(p)|}{\mathit{NumTriples}}. \tag{4.2}$$

The second feature template is the *normalized entity frequency* for each entity e that participates in fact f (we also include the minimum, maximum and average value for each fact as metafeatures). This is defined as the ratio of the number of triples in which e occurs in the KG over the number of triples in the KG:

$$\mathit{EntFreq}(e) = \frac{|\mathit{TriplesEnt}(e)|}{\mathit{NumTriples}}. \tag{4.3}$$

The final feature template in this feature group is *path informativeness*, proposed by Pirrò [139], which we apply for both f_q and f_c (recall from Section 4.2.1 that a fact f is a path in the KG). This feature is analog to TF.IDF and aims to estimate the importance of predicates for an entity. The informativeness of a path π is defined as follows [139]:

$$I(\pi) = \frac{1}{2|\pi|} \sum_{\langle s,p,t\rangle \in \pi} \mathit{PFITF}_{out}(p, s, K) + \mathit{PFITF}_{in}(p, t, K), \tag{4.4}$$

where:

$$\mathit{PFITF}_x(p, e, K) = \mathit{PF}_x(p, e) * \mathit{ITF}(p), x \in \{in, out\},$$

where $\mathit{ITF}(p)$ is the inverse triple frequency of predicate p:

$$\mathit{ITF}(p) = \log \frac{\mathit{NumTriples}}{|\mathit{TriplesPred}(p)|},$$

$\mathit{PF}_{out}(p, e)$ is the outgoing predicate frequency of e when p is the predicate:

$$\mathit{PF}_{out}(p, e) = \frac{|\mathit{TriplesSubj}(e) \cap \mathit{TriplesPred}(p)|}{|\mathit{TriplesSubj}(e)|},$$

and $\mathit{PF}_{in}(p, e)$ is the incoming predicate frequency of e when p is the predicate:

$$\mathit{PF}_{in}(p, e) = \frac{|\mathit{TriplesObj}(e) \cap \mathit{TriplesPred}(p)|}{|\mathit{TriplesObj}(e)|}.$$

(ii) Relevance This group of feature templates gives us signal on the relevance of a candidate fact f_c w.r.t. the query fact f_q. The first of these feature templates measures *entity similarity* for each pair $(e_1, e_2) \in \mathit{Entities}(f_q) \times \mathit{Entities}(f_c)$ (we also include the minimum, maximum and average entity similarity as metafeatures). We measure entity similarity using type-based Jaccard similarity:

$$\mathit{EntTypeSim}(e_1, e_2) = \mathit{JaccardSim}(\mathit{Types}(e_1), \mathit{Types}(e_2)). \tag{4.5}$$

The next feature template in the *relevance* category is *entity distance*, which allows us to reason about the distance of two entities $(e_1, e_2) \in \mathit{Entities}(f_q) \times \mathit{Entities}(f_c)$ (we also include the minimum, maximum and average entity distance as metafeatures). This feature is defined as the length of the shortest path between e_1 and e_2 in K. The intuition is that we can get a signal for the relevance of f_c by measuring how “close” the entities in f_c are to the entities of f_q in the KG.

The next set of features measure *predicate similarity* between every pair of predicates $(p_1, p_2) \in \mathit{Preds}(f_q) \times \mathit{Preds}(f_c)$ (we also include the minimum, maximum and average predicate similarity as metafeatures). The intuition is that if f_c has predicates that are highly similar to the predicates in f_q, then f_c might be relevant to f_q. We measure predicate similarity in two ways. First, by measuring the co-occurrence of entities that participate in the predicates p_1 and p_2:

$$\begin{aligned}&\mathit{PredCooccSim}(p_1, p_2) = \\ &\mathit{JaccardSim}(\mathit{UniqEnt}(\mathit{TriplesPred}(p_1)), \mathit{UniqEnt}(\mathit{TriplesPred}(p_2))).\end{aligned} \tag{4.6}$$

For instance, $\mathit{PredCooccSim}(p_1, p_2)$ would be high for $p1 = \mathit{starredIn}$ and $p2 = \mathit{directedBy}$. Second, by measuring the jaccard similarity of the set of predicates in f_q with the set of predicates in f_c [139]:

$$\begin{aligned}&\mathit{SetPredicatesJaccardSim}(f_q, f_c) = \\ &\mathit{JaccardSim}(\mathit{Preds}(f_q), \mathit{Preds}(f_c)).\end{aligned} \tag{4.7}$$

Finally, we add a binary feature that captures whether f_q and f_c have the same CVT entity, i.e., f_c is an attribute of f_q.

(iii) Miscellaneous This set of features includes whether f_q has a CVT entity (same for f_c). We also include whether an entity is a date (for all entities of f_q and f_c). Finally, we include the concatenation of the predicates of f_q as a feature using one-hot encoding.

4.4 Experimental Setup

In this section we describe the setup of our experiments that aim to answer **RQ3**, which we break down to the following research sub-questions:

RQ3.1 How does NFCM perform compared to a set of heuristic baselines on a crowdsourced dataset?

RQ3.2 How does NFCM perform compared to a scoring function that scores candidate facts w.r.t. a query fact using the relevance labels gathered from distant supervision on a crowdsourced dataset?

Table 4.2: Examples of relationships used in this work.

Domain	**Relationship**
People	$spouseOf(person, person)$ $parentOf(person, person)$ $educatedAt(person, organization)$
Business	$founderOf(person, organization)$ $boardMemberOf(person, organization)$ $leaderOf(person, organization)$
Film	$starredIn(person, film)$ $directorOf(person, film)$ $producerOf(person, film)$

RQ3.3 Does NFCM benefit from both the handcrafted features and the automatically learned features?

RQ3.4 What is the per-relationship performance of NFCM? How does the number of instances per relationship affect the ranking performance?

4.4.1 Knowledge graph

We use the latest edition of Freebase as our knowledge graph [23]. We include Freebase relations from the following set of domains: *People, Film, Music, Award, Government, Business, Organization, Education*. Following previous work [122], we exclude triples that have an equivalent reversed triple.

4.4.2 Dataset

Our dataset consists of query facts, candidate facts, and a relevance label for each query-candidate fact pair. In order to construct our evaluation dataset we need to start with a set of relationships. Given that most of our domains are people-centric, we obtain this set by extracting all relationships from Freebase that have an entity of type *Person* as one of the entities. In the end, we are left with 65 unique relationships in total (see Table 4.2 for example relationships). We then proceed to gather our set of query facts. For each relationship, we sample at most 2,000 query facts, provided that they have at least one relevant fact after applying the procedure described in Section 4.4.3. In total, the dataset contains 62,044 query facts (954.52 on average per relationship). After gathering query facts for each relation, we enumerate candidate facts for each query fact using the procedure described in Section 4.3.1. Finally, we randomly split the dataset per relationship (70% of the query facts for training, 10% for validation, 20% for testing). Table 4.3 shows statistics of the resulting dataset.

Note that we train and tune the fact ranking models with the training and validation sets in Table 4.3 respectively, using the automatically gathered relevance labels (see Section 4.4.3). The test set was only used for preliminary experiments (not reported) and for constructing our manually curated evaluation dataset (see Section 4.4.4). We

Table 4.3: Statistics of the dataset gathered using distant supervision (see Section 4.4.3).

Part	**# query facts**	**# candidate facts**			
		average	median	max.	min.
Training	44,632	1,420	741	9,937	2
Validation	4,983	1,424	749	9,796	3
Test	12,429	1,427	771	9,924	3

describe how we automatically gather noisy relevance labels for our dataset in the next section.

4.4.3 Gathering noisy relevance labels

Gathering relevance labels for our task is challenging due to the size and heterogeneous nature of KGs, i.e., having a large number of facts and relationship types. Therefore, we turn to distant supervision [122] to gather relevance labels at scale. We choose to get a supervision signal from Wikipedia for the following reasons: (i) it has a high overlap of entities with the KG we use, and (ii) facts that are in KGs are usually expressed in Wikipedia articles alongside other, related facts. We filter Wikipedia to select articles whose main entity is in Freebase, and the entity type corresponds to one of the domains listed in Section 4.4.1. This results in a set of 1,743,191 Wikipedia articles.

The procedure we follow for gathering relevance labels given a query fact f_q and its set of candidate facts F is as follows. For a query fact $f_q = r\langle s, t\rangle$, we focus on the Wikipedia article of entity s. First, as Wikipedia style guidelines dictate that only the first mention of another entity should be linked, we augment the articles with additional entity links using an entity linking method proposed in [186]. Next, we retain only segments of the Wikipedia article that contain references to t. Here, a segment refers to the sentence that has a reference to t and also one sentence before and one after the sentence. For each such extracted segment, we assume that it expresses the fact f_q, which is a common assumption in gathering noisy training data for relation extraction [122]. From the segments, we then collect a set of other entities, O, that occur in the same sentence that mentions t: for computational efficiency, we enforce $|O| \leq 20$. Then, we extract facts for all possible pairs of entities $\langle e_1, e_2\rangle \in \{O \cup \{s, t\}\} \times \{O \cup \{s, t\}\}$. If there is a single fact f_c in K that connects e_1 and e_2, we deem f_c relevant for f_q. However, if there are multiple facts connecting e_1 and e_2 in K, the mention of the fact in the specific segment is ambiguous and thus we do not deem any of these facts as relevant [170]. The rest of the facts in F are deemed irrelevant for f_q.

The distribution of relevant/non-relevant labels in the distantly supervised data is heavily skewed: out of 87,998,956 facts in total, only 225,032 are deemed to be relevant (0.26%). This is expected since the candidate fact enumeration step can generate thousands of facts for a certain query fact (see Section 4.3.1).

As a sanity check, we evaluate the performance of our approach to collect distant supervision data by sampling 5 query facts for each relation in our dataset. For these query facts, we perform manual annotations on the extracted candidate facts that were deemed as relevant by the distant supervision procedure. We obtain an overall precision of 76% when comparing the relevance labels of the distant supervision against our manual annotations. This demonstrates the potential of our distant supervision strategy for creating training data.

4.4.4 Manually curated evaluation dataset

In order to evaluate the performance of NFCM on the KG fact contextualization task, we perform crowdsourcing to collect a human-curated evaluation dataset. The procedure we use to construct this evaluation dataset is as follows. First, for each of the 65 relationships we consider, we sample five query facts of the relationship from the test set (see Section 4.4.2). Since fact enumeration for a query fact can yield hundreds or thousands of facts (Section 4.3.1), it is infeasible to consider all the candidate facts for manual annotation. Therefore, we only include a candidate fact in the set of facts to be annotated if: (i) the candidate fact was deemed relevant by the automatic data gathering procedure (Section 4.4.3), or (ii) the candidate fact matches a fact pattern that is built using relevant facts that appear in at least 10% of the query facts of a certain relationship. An example fact pattern is $\mathit{parentOf}\langle ?, ? \rangle$, which would match the fact $\mathit{parentOf}\langle$Bill Gates, Jennifer Gates$\rangle$.

We use the CrowdFlower platform, and ask the annotators to judge a candidate fact w.r.t. its relevance to a query fact. We provide the annotators with the following scenario (details omitted for brevity):

> *We are given a specific real-world fact, e.g., "Bill Gates is the founder of Microsoft", which we call the query fact. We are interested in writing a description of the query fact (a sentence or a small paragraph). The purpose of this assessment task is to identify other facts that could be included in a description of the query fact. Note that even though all facts presented for assessment will be accurate, not all will be relevant or equally important to the description of the main fact.*

We ask the annotators to assess the relevance of a candidate fact in a 3-graded scale:

- *very relevant*: I would include the candidate fact in the description of the query fact; the candidate fact provides additional context to the query fact.
- *somewhat relevant*: I would include the candidate fact in the description of the query fact, but only if there is space.
- *irrelevant*: I would not include the candidate fact in the description of the query fact.

Alongside each query-candidate fact pair, we provide a set of extra facts that could possibly be used to decide on the relevance of a candidate fact. These include facts that connect the entities in the query fact with the entities in the candidate fact. For example, if we present the annotators with the query fact $\mathit{spouseOf}\langle$Bill Gates, Melinda Gates$\rangle$ and the candidate fact $\mathit{parentOf}\langle$Melinda Gates, Jennifer Gates$\rangle$ we also show the fact $\mathit{parentOf}\langle$Bill Gates, Jennifer Gates$\rangle$.

Table 4.4: Relevance label distribution of the crowdsourced evaluation dataset.

Relevance	**Non-attribute facts (%)**	**Attribute facts (%)**
Irrelevant	60.86	34.34
Somewhat relevant	34.49	57.81
Very relevant	4.63	7.84

Each query-candidate fact pair is annotated by three annotators. We use majority voting to obtain the gold labels, breaking ties arbitrarily. The annotators get a payment of 0.03 dollars per query-candidate fact pair.

By following the crowdsourcing procedure described above, we obtain 28,281 fact judgments for 2,275 query facts (65 relations, 5 query facts each). Table 4.4 details the distribution of the relevance labels. One interesting observation is that facts that are attributes of other facts (see Section 4.2.1) tend to have relatively more relevant judgments than the ones that are not. This is expected since some of them are attributes of the query fact (e.g., date of marriage for a *spouseOf* query fact). Finally, Fleiss' kappa is $\kappa = 0.4307$, which is considered moderate agreement. Note that all the results reported in Section 4.5 are on the manually curated dataset described here.

Evaluation metrics We use the following standard retrieval evaluation metrics: MAP, NDCG@5, NDCG@10 and MRR. In the case of MAP and MRR, which expect binary labels, we consider "very relevant" and"somewhat relevant" as "relevant". We report on statistical significance with a paired two-tailed t-test.

4.4.5 Heuristic baselines

To the best of our knowledge, there is no previously published method that addresses the task introduced in this chapter. Therefore, we devise a set of intuitive baselines that are used to showcase that our task is not trivial. We derive them by combining features we introduced in Section 4.3.2. We define these heuristic functions below:

- *Fact informativeness (FI).* Informativeness of the candidate fact f_c [139, Eq. 4.4]. This baseline is independent of f_q.

- *Average predicate similarity (APS).* Average predicate similarity of all pairs of predicates $(p_1, p_2) \in \mathit{Preds}(f_q) \times \mathit{Preds}(f_c)$ (Eq. 4.6). The intuition here is that f_c might be relevant to f_q if it contains predicates that are similar to the predicates of f_q.

- *Average entity similarity (AES).* Average entity similarity of all pairs of entities in $(e_1, e_2) \in \mathit{Entities}(f_q) \times \mathit{Entities}(f_c)$ (Eq. 4.5). The assumption here is that f_c might be relevant to f_q if it contains entities that are similar to the entities of f_q.

4.4.6 Implementation details

The models described in Section 4.3.2 are implemented in TensorFlow v.1.4.1 [1]. Table 4.5 lists the hyperparameters of NFCM. We tune the variable hyper-parameters of this table on the validation set and optimize for NDCG@5.

Table 4.5: Hyperparameters of NFCM, tuned on the validation set.

Description	**Value(s)**
# negative samples k during training	[1, 10, 100]
Learning rate	[0.01, 0.001, 0.0001]
d_z: entity type embedding dimension	[64, 128, 256]
d_p: Predicate embedding dimension	[64, 128, 256]
RNN cell size	[64, 128, 256]
RNN cell dropout	[0.0, 0.2]
α: # hidden layers of MLP-o	[0, 1, 2]
β: # dimension of MLP-o hidden layers	[50, 100]
L2 regularization factor for MLP-o kernel	[0.0, 0.1, 0.2]

4.5 Results and Discussion

In this section we discuss and analyze the results of our evaluation, answering the research questions listed in Section 4.4.

In our first experiment, we compare NFCM to a set of heuristic baselines we derived to answer **RQ3.1**. Table 4.6 shows the results. We observe that NFCM significantly outperforms the heuristic baselines by a large margin. We have also experimented with linear combinations of the above heuristics but the performance does not improve over the individual ones and therefore we omit those results. We conclude that the task we define in this chapter is not trivial to solve and simple heuristic functions are not sufficient.

In our second experiment we compare NFCM with distant supervision and aim to answer **RQ3.2**. That is, how does NFCM perform compared to DistSup, a scoring function that scores candidate facts w.r.t. a query fact using the relevance labels gathered from distant supervision. The aim of this experiment is to investigate whether it is beneficial to learn ranking functions based on the signal gathered from distant supervision, and to see if we can improve performance over the latter. Table 4.7 shows the results. We observe that NFCM significantly outperforms DistSup on MAP, NDCG@5, and NDCG@10 and conclude that learning ranking functions (and in particular NFCM) based on the signal gathered from distant supervision is beneficial for this task. We also observe that NFCM performs significantly worse than DistSup on MRR. One possible reason for this is that NFCM returns facts that are indeed relevant but were not selected for annotation and thus assumed not relevant, since the data annotation procedure is biased towards DistSup (see Section 4.4.4). We aim to validate this hypothesis by conducting an additional user study in future work.

Table 4.6: Comparison between NFCM and the heuristic baselines. Significance is tested between NFCM and AES, the best performing baseline. We depict a significant improvement of NFCM over AES for $p < 0.05$ as $^\blacktriangle$.

Method	MAP	NDCG@5	NDCG@10	MRR
FI	0.1222	0.0978	0.1149	0.1928
APS	0.2147	0.2175	0.2354	0.3760
AES	0.2950	0.3284	0.3391	0.5214
NFCM	**0.4874**$^\blacktriangle$	**0.5110**$^\blacktriangle$	**0.5289**$^\blacktriangle$	**0.7749**$^\blacktriangle$

Table 4.7: Comparison between NFCM and the distant supervision baseline. We depict a significant improvement of NFCM over DistSup as $^\blacktriangle$ and a significant decrease as $^\blacktriangledown$ ($p < 0.05$).

Method	MAP	NDCG@5	NDCG@10	MRR
DistSup	0.2831	0.4489	0.3983	**0.8256**
NFCM	**0.4874**$^\blacktriangle$	**0.5110**$^\blacktriangle$	**0.5289**$^\blacktriangle$	0.7749$^\blacktriangledown$

Nevertheless, having an automatic method for KG fact contextualization trained with distant supervision becomes increasingly important for tail entities for which we might only have information in the KG itself and not in external text corpora or other sources.

In order to answer **RQ3.3**, that is, whether NFCM benefits from both the hand-crafted features and the learned features, we perform an ablation study. Specifically, we test the following variations of NFCM that only modify the final layer of the architecture (see Section 4.3.2):

(i) LF: Keeps the learned features ($\boldsymbol{v_q}$ and $\boldsymbol{v_a}$), and ignores the hand-crafted features $\boldsymbol{x}$.

(ii) HF: Keeps the hand-crafted features ($\boldsymbol{x}$) and ignores the learned features ($\boldsymbol{v_q}$ and $\boldsymbol{v_a}$).

We tune the parameters of LF and HF on the validation set. Table 4.8 shows the results. First, we observe that NFCM outperforms HF by a large margin. Also, NFCM outperforms LF on all metrics (significantly so for MAP and NDCG@10) which means that by combining HF and LF we are able to obtain more relevant results at lower positions of the ranking. We aim to explore more sophisticated ways of combining LF and HF in future work. In order to verify whether LF and HF have complementary signals, we plot the per-query differences in NDCG@5 for LF and HF in Figure 4.5. We observe that the performance of LF and HF varies across query facts, confirming the hypotheses that LF and HF yield complementary signals.

In order to answer **RQ3.4**, we conduct a performance analysis per relationship. Figure 4.6 shows the per-relationship NDCG@5 performance of NFCM – query fact scores are averaged per relationship. The relationship for which NFCM performs best is $profession$, which has a NDCG@5 score of 0.8275. The relationship for which NFCM

Table 4.8: Comparison between the full NFCM model and its variations. Significance is tested between NFCM and its best variation (LF). We depict a significant improvement of NFCM over LF for $p < 0.05$ as $^\blacktriangle$.

Method	MAP	NDCG@5	NDCG@10	MRR
HF	0.4620	0.4753	0.4989	0.7180
LF	0.4676	0.4993	0.5134	0.7647
NFCM	**0.4874**$^\blacktriangle$	**0.5110**	**0.5289**$^\blacktriangle$	**0.7749**

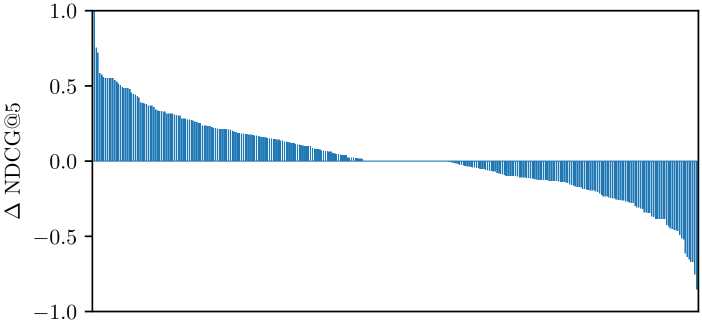


Figure 4.5: Per query fact differences in NDCG@5 between the variation of NFCM that only uses the learned features (LF) and the best-performing variation of NFCM that only uses the hand-crafted features (HF). A positive value indicates that LF performs better than HF on a query fact and vice versa.

performs worst at is *awardNominated*, which has a NDCG@5 score of 0.1. Further analysis showed that *awardNominated* has a very large number of candidate facts on average, which might explain the poor performance on that relationship.

Furthermore, we investigate how the number of queries we have in the training set for each relationship affects the ranking performance. Figure 4.7 shows the results. From this figure we conclude that there is no clear relationship and thus that NFCM is robust to the size of the training data for each relationship.

Next, we analyse the performance of NFCM with respect to the number of candidates per query fact; Figure 4.8 shows the results. We observe that the performance decreases when we have more candidate facts for a query, although not by a large margin, and that there does not seem to be a clear relationship between performance and the number of candidates to rank.

4.6 Related Work

The specific task we introduce in this chapter has not been addressed before, but there is related work in three main areas: entity relationship explanation, distant supervision,

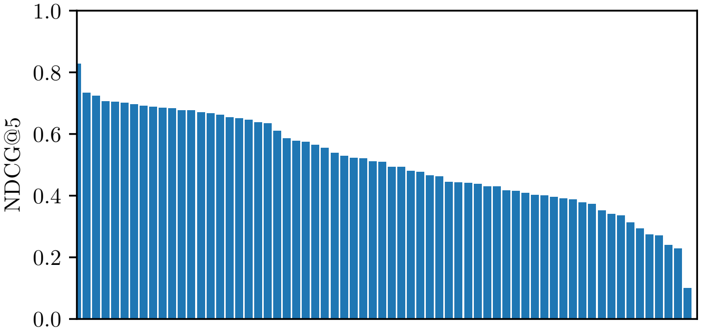


Figure 4.6: NDCG@5 for NFCM per relationship.

and fact ranking.

4.6.1 Relationship explanation

Explanations for relationships between pairs of entities can be provided in two ways: *structurally*, i.e., by providing paths or sub-graphs in a KG containing the entities, or *textually*, by ranking or generating text snippets that explain the connection.

Fang et al. [61] focus on explaining connections between entities by mining relationship explanation patterns from the KG. Their approach consists of two main components: explanation enumeration and explanation ranking. The first phase generates all patterns in the form of paths connecting the two entities in the KG, which are then combined to form explanations. In the final stage, the candidate explanations are ranked using notions of interestingness. Seufert et al. [164] propose a similar approach for entity sets. Their method focuses on explaining the connections between entity sets based on the concept of relatedness cores, i.e., dense subgraphs that have strong relations with both query sets. Pirrò [139] also provide explanations of the relation between entities in terms of the top-k most informative paths between a query pair of entities; such paths are ranked and selected based on path informativeness and diversity, and pattern informativeness.

As to textual explanations for entity relationships, Voskarides et al. [185] focus on human-readable descriptions. They model the task as a learning to rank problem for sentences and employ a rich set of features. Huang et al. [81] build on the aforementioned work and propose a pairwise ranking model that leverages clickthrough data and uses a convolutional neural network architecture. While these approaches rank existing candidate explanations, Voskarides et al. [186] focus on generating explanations from scratch. They automatically identify the most common sentence templates for a particular relationship and, for each new relationship instance, these templates are ranked and instantiated using contextual information from the KG.

The work described above focuses on explaining entity relationships in KGs; no previous work has focused on ranking additional KG facts for an input entity relationship as we do in this chapter.

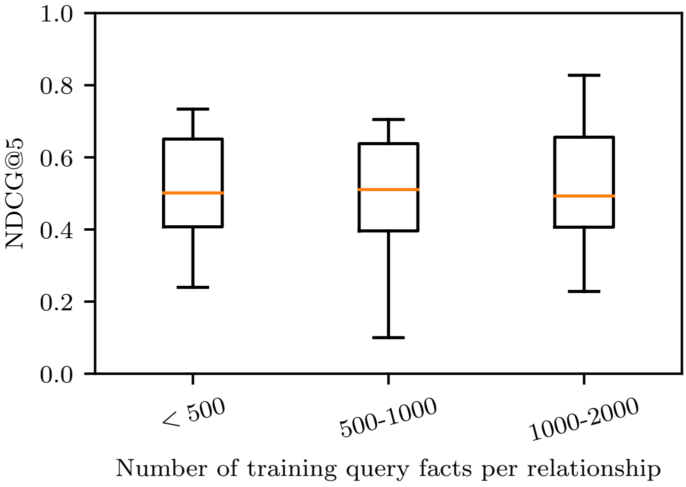


Figure 4.7: Box plot that shows NDCG@5 per number of training query facts of each relationship (binned). Each box shows the median score with an orange line and the upper and lower quartiles (maximum and lower values shown outside each box).

4.6.2 Distant supervision

When obtaining labeled data is expensive, training data can be generated automatically. Mintz et al. [122] introduce distant supervision for relation extraction; for a pair of entities that is connected by a KG relation, they treat all sentences that contain those entities in a text corpus as positive examples for that relation. Follow-up work on relation extraction address the issue of noise related to distant supervision. Alfonseca et al. [5], Riedel et al. [151], Surdeanu et al. [172] refine the model by relaxing the assumptions in the original method or by modeling noisy labels.

Beyond relation extraction, distant supervision has also been applied in other KG-related tasks. Ren et al. [150] introduce a joint approach entity recognition and classification based on distant supervision. Ling and Weld [109] used distant supervision to automatically label data for fine-grained entity recognition.

4.6.3 Fact ranking

In fact ranking, the goal is to rank a set of attributes with respect to an entity. Hasibi et al. [75] consider fact ranking as a component for entity summarization for entity cards. They approach fact ranking as a learning to rank problem. They learn a ranking model based on importance, relevance, and other features relating a query and the facts. Aleman-Meza et al. [4] explore a similar task, but rank facts with respect to a pair of entities to discover paths that contain informative facts between the pair.

Graph matching involves matching two graphs and discovering the patterns of relationships between them to infer their similarity [34]. Although our task can be

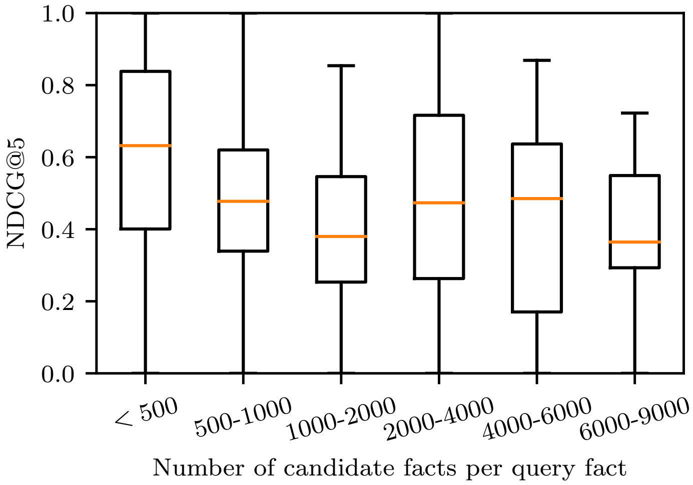


Figure 4.8: Box plot that shows NDCG@5 per number of candidate facts of each query fact (binned). Each box shows the median score with an orange line and the upper and lower quartiles (maximum and lower values shown outside each box).

considered as comparing a small query subgraph (i.e., query triples) and a knowledge graph, the goal is different from graph matching which mainly concerns aligning two graphs rather than enhancing one query graph.

Our work differs from the work discussed above in the following major ways. First, we enrich a query fact between two entities by providing relevant additional facts in the context of the query fact, taking into account both the entities and the relation of the query fact. Second, we rank whole facts from the KG instead of just entities. Last, we provide a distant supervision framework for generating the training data so as to make our approach scalable.

4.7 Conclusion

In this chapter, we introduced the knowledge graph fact contextualization task and proposed NFCM, a weakly-supervised method to address it. NFCM first generates a candidate set for a query fact by looking at 1 or 2-hop neighbors and then ranks the candidate facts using supervised machine learning. NFCM combines handcrafted features with features that are automatically identified using deep learning. We use distant supervision to boost the gathering of training data by using a large entity-tagged text corpus that has a high overlap with entities in the KG we use. Our experimental results show that (i) distant supervision is an effective means for gathering training data for this task, (ii) NFCM significantly outperforms several heuristic baselines for this task, and (iii) both the handcrafted and automatically-learned features contribute to the retrieval effectiveness of NFCM.

For future work, we aim to explore more sophisticated ways of combining handcrafted with automatically learned features for ranking. Additionally, we want to explore other data sources for gathering training data, such as news articles and click logs. Finally, we want to explore methods for combining and presenting the ranked facts in search engine result pages in a diversified fashion.

This chapter concludes our study on the first part of the thesis, which focuses on how to make structured knowledge more accessible to the user. Next, in Chapter 5, we address a different research theme, namely improving interactive knowledge gathering.

Part II

Improving Interactive Knowledge Gathering

5

Query Resolution for Conversational Search with Limited Supervision

In the second part of this thesis, we move to the research theme of improving interactive knowledge gathering and focus on conversational search. In this Chapter, we aim to answer **RQ4**: Can we use query resolution to identify relevant context and thereby improve retrieval in conversational search?

5.1 Introduction

Conversational AI deals with developing dialogue systems that enable interactive knowledge gathering [64]. A large portion of work in this area has focused on building dialogue systems that are capable of engaging with the user through chit-chat [104] or helping the user complete small well-specified tasks [135]. In order to improve the capability of such systems to engage in complex information seeking conversations [142], researchers have proposed information seeking tasks such as conversational question answering (QA) over simple contexts, such as a single-paragraph text [35, 146]. In contrast to conversational QA over simple contexts, in conversational search, a user aims to interactively find information stored in a large document collection [45].

In this chapter, we study multi-turn passage retrieval as an instance of conversational search: given the conversation history (the previous turns) and the current turn query, we aim to retrieve passage-length texts that satisfy the user's underlying information need [46]. Here, the current turn query may be under-specified and thus, we need to take into account context from the conversation history to arrive at a better expression of the current turn query. Thus, we need to perform *query resolution*, that is, add missing context from the conversation history to the current turn query, if needed. An example of an under-specified query can be seen in Table 5.1, turn #4, for which the gold standard query resolution is: "*when was saosin 's first album released?*". In this example, context from all turns #1 ("saosin"), #2 ("band") and #3 ("first") have to be taken into account to arrive to the query resolution.

Designing automatic query resolution systems is challenging because of phenomena such as zero anaphora, topic change and topic return, which are prominent in information

This chapter was published as [189].

Table 5.1: Excerpt from an example conversational dialog. Co-occurring terms in the conversation history and the relevant passage to the current turn (#4) are shown in bold-face.

Turn	Query
1	who formed **saosin**?
2	when was the band founded?
3	what was their **first** album?
4	when was the album released? *resolved:* when was saosin 's first album released?
Relevant passage to turn #4: The original lineup for **Saosin**, consisting of Burchell, Shekoski, Kennedy and Green, was formed in the summer of 2003. On June 17, the **band** released their **first** commercial production, the EP Translating the Name.	

seeking conversations [207]. These phenomena are not easy to capture with standard NLP tools (e.g., coreference resolution). Also, heuristics such as appending (part of) the conversation history to the current turn query are likely to lead to query drift [123]. Recent work has modeled query resolution as a sequence generation task [58, 96, 145]. Another way of implicitly solving query resolution is by query modeling [69, 181, 201], which has been studied and developed under the setup of session-based search [29, 30].

In this chapter, we propose to model query resolution for conversational search as a binary term classification task: for each term in the previous turns of the conversation decide whether to add it to the current turn query or not. We propose QuReTeC (**Qu**ery **Re**solution by **Te**rm **C**lassification), a query resolution model based on bidirectional transformers [182] – more specifically BERT [50]. The model encodes the conversation history and the current turn query and uses a term classification layer to predict a binary label for each term in the conversation history. We integrate QuReTeC in a standard two-step cascade architecture that consists of an initial retrieval step and a reranking step. This is done by using the set of terms predicted as relevant by QuReTeC as query expansion terms.

Training QuReTeC requires binary labels for each term in the conversation history. One way to obtain such labels is to use human-curated gold standard query resolutions [58]. However, these labels might be cumbersome to obtain in practice. On the other hand, researchers and practitioners have been collecting general-purpose passage relevance labels, either by the means of human annotations or by the means of weak signals, e.g., clicks or mouse movements [88]. We propose a distant supervision method to automatically generate training data, on the basis of such passage relevance labels. The key assumption is that passages that are relevant to the current turn share context with the conversation history that is missing from the current turn query. Table 5.1 illustrates this assumption: the relevant passage to turn #4 shares terms with the conversation history. Thus, we label the terms that co-occur in the relevant passages[1] and the conversation history as relevant for the current turn.

[1] A relevance passage contains not only the answer to the question but also context and supporting facts that allow the algorithm or the human to reach to this answer.

Our main contributions can be summarized as follows:

1. We model the task of query resolution as a binary term classification task and propose to address it with a neural model based on bidirectional transformers, QuReTeC.
2. We propose a distant supervision approach that can use general-purpose passage relevance data to substantially reduce the amount of human-curated data required to train QuReTeC.
3. We experimentally show that when integrating the QuReTeC model in a multi-stage ranking architecture we significantly outperform baseline models. Also, we conduct extensive ablation studies and analyses to shed light into the workings of our query resolution model and its impact on retrieval performance.

5.2 Related work

5.2.1 Conversational search

Early studies on conversational search have focused on characterizing information seeking strategies and building interactive IR systems [16, 17, 43, 131]. Vtyurina et al. [191] investigated human behaviour in conversational systems through a user study and find that existing conversational assistants cannot be effectively used for conversational search with complex information needs. Radlinski and Craswell [143] present a theoretical framework for conversational search, which highlights the need for multi-turn interactions. Dalton et al. [46] organize the Conversational Assistance Track (CAsT) at TREC 2019. The goal of the track is to establish a concrete and standard collection of data with information needs to make systems directly comparable. They release a multi-turn passage retrieval dataset annotated by experts, which we use to compate our method to the baseline methods.

5.2.2 Query resolution

Query resolution has been studied in the context of dialogue systems. Raghu et al. [145] develop a pipeline model for query resolution in dialogues as text generation. Kumar and Joshi [96] follow up on that work by using a sequence to sequence model combined with a retrieval model. However, both these works rely on templates that are not available in our setting. More related to our work, Elgohary et al. [58] studied query resolution in the context of conversational QA over a single paragraph text. They use a sequence to sequence model augmented with a copy and an attention mechanism and a coverage loss. They annotate part of the QuAC dataset [35] with gold standard query resolutions on which they apply their model and obtain competitive performance. In contrast to all the aforementioned works that model query resolution as text generation, we model query resolution as binary term classification in the conversation history.

5.2.3 Query modeling

Query modeling has been used in session search, where the task is to retrieve documents for a given query by utilizing previous queries and user interactions with the retrieval system [29]. Guan et al. [69] extract substrings from the current and previous turn queries to construct a new query for the current turn. Yang et al. [201] propose a query change model that models both edits between consecutive queries and the ranked list returned by the previous turn query. Van Gysel et al. [181] compare the lexical matching session search approaches and find that naive methods based on term frequency weighing perform on par with specialized session search models. The methods described above are informed by studies of how users reformulate their queries and why [167], which, in principle, is different in nature from conversational search. For instance, in session search users tend to add query terms more than removing query terms, which is not the case in (spoken) conversational search. Another form of query modeling is query expansion. Pseudo-relevance feedback is a query expansion technique that first retrieves a set of documents that are assumed to be relevant to the query, and then selects terms from the retrieved documents that are used to expand the query [2, 98, 130]. Note that pseudo-relevance feedback is fundamentally different from query resolution: in order to revise the query, the former relies on the top-ranked documents, while the latter only relies on the conversation history.

Distant supervision Distant supervision can be used to obtain large amounts of noisy training data. One of its most successful applications is relation extraction, first proposed by Mintz et al. [122]. They take as input two entities and a relation between them, gather sentences where the two entities co-occur from a large text corpus, and treat those as positive examples for training a relation extraction system. Beyond relation extraction, distant supervision has also been used to automatically generate noisy training data for other tasks such as named entity recognition [204], sentiment classification [152], knowledge graph fact contextualization [187] and dialogue response generation [149]. In our work, we follow the distant supervision paradigm to automatically generate training data for query resolution in conversational search by using query-passage relevance labels.

5.3 Multi-turn Passage Retrieval Pipeline

In this section we provide formal definitions and describe our multi-turn passage retrieval pipeline. Table 5.2 lists notation used in this chapter.

5.3.1 Definitions

Multi-turn passage ranking Let $[q_1, \ldots, q_{i-1}, q_i]$ be a sequence of conversational queries that share a common topic T. Let q_i be the current turn query and $q_{1:i-1}$ be the conversation history. Given q_i and $q_{1:i-1}$, the task is to retrieve a ranked list of passages L from a passage collection D that satisfy the user's information need.[2]

In the multi-turn passage ranking task, the current turn query q_i is often underspecified due to phenomena such as zero anaphora, topic change, and topic return. Thus, context from the conversation history $q_{1:i-1}$ must be taken into account to arrive at a

[2] We follow the TREC CAsT setup and only take into account $q_{1:i-1}$ but not the passages retrieved for $q_{1:i-1}$.

Table 5.2: Notation used in the chapter.

Name	Description
$terms(x)$	Set of terms in term sequence x
D	Passage collection
q_i	Query at the current turn i
$q_{1:i-1}$	Sequence of previous turn queries
q_i^*	Gold standard resolution of q_i
$E_{q_i}^*$	Gold standard resolution terms for q_i, see Eq. (5.2)
$\hat{q}_i$	Predicted resolution of q_i
$p_{q_i}^*$	A relevant passage for q_i

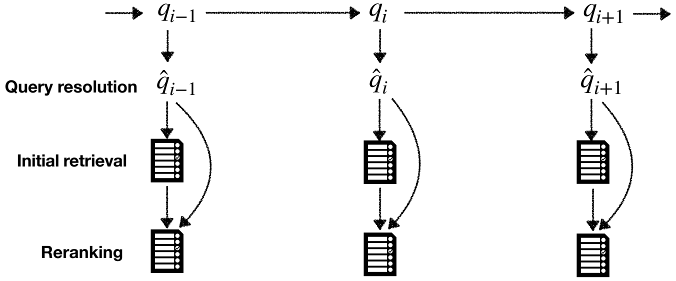


Figure 5.1: Illustration of our multi-turn passage retrieval pipeline for three turns.

better expression of the current turn query q_i. This challenge can be addressed by query resolution.

Query resolution Given the conversation history $q_{1:i-1}$ and the current turn query q_i, output a query $\hat{q}_i$ that includes both the existing information in q_i and the missing context of q_i that exists in the conversation history $q_{1:i-1}$.

5.3.2 Multi-turn passage retrieval pipeline

Figure 5.1 illustrates our multi-turn passage retrieval pipeline. We use a two-step cascade ranking architecture [192], which we augment with a query resolution module (Section 5.4). First, the unsupervised initial retrieval step outputs the initial ranked list L_1 (Section 5.3.2). Second, the re-ranking step outputs the final ranked list L (Section 5.3.2). Below we describe the two steps of the cascade ranking architecture.

Initial retrieval step

In this step we obtain the initial ranked list L_1 by scoring each passage p in the passage collection D with respect to the resolved query $\hat{q}_i$ using a lexical matching ranking function f_1. We use query likelihood (QL) with Dirichlet smoothing [212] as f_1, since it outperformed other ranking functions such as BM25 in preliminary experiments over the TREC CAsT dataset.

Reranking step

In this step, we re-rank the list L_1 by scoring each passage $p \in L_1$ with a ranking function f_2 to obtain the final ranked list L. To construct f_2, we use rank fusion and combine the scores obtained by f_1 (used in initial retrieval step) and a supervised neural ranker f_n. Next, we describe the neural ranker f_n.

Supervised neural ranker We use BERT [50] as the neural ranker f_n, as it has been shown to achieve state-of-the-art performance in ad-hoc retrieval [112, 141, 203]. Also, BERT has been shown to prefer semantic matches [141], and thereby can be complementary to f_1, which is a lexical matching method. As is standard when using BERT for pairs of sequences, the input to the model is formatted as [`<CLS>`, $\hat{q}_i$ `<SEP>`, p], where `<CLS>` is a special token, $\hat{q}_i$ is the resolved current turn query, p is the passage. We add a dropout layer and a linear layer l_a on top of the representation of the `<CLS>` token in the last layer, followed by a $\tanh$ function to obtain f_n [112]. We score each passage $p \in L_1$ using f_n to obtain L_n . We fine-tune the pretrained BERT model using pairwise ranking loss on a large-scale single-turn passage ranking dataset [203]. During training we sample as many negative as positive passages per query.

Rank fusion We design f_2 such that it combines lexical matching and semantic matching [132]. We use Reciprocal Rank Fusion (RRF) [41] to combine the score obtained by the lexical matching ranking function f_1, and the semantic matching supervised neural ranker f_n. We choose RRF because of its effectiveness in combining individual rankers in ad-hoc retrieval and because of its simplicity (it has only one hyper-parameter). We define f_2 as the RRF of L_1 and L_n [41]:

$$f_2(p) = \sum_{L' \in \{L_1, L_n\}} \frac{1}{k + rank(p, L')}, \tag{5.1}$$

where $rank(p, L')$ is the rank of passage p in a ranked list L', and k is a hyperparameter.[3] We score each passage p in the initial ranked list L_1 with f_2 to obtain the final ranked list L.

Since developing specialized re-rankers for the task at hand is not the focus of this work, we leave more sophisticated methods for choosing the neural ranker f_n and for combining multiple rankers as future work. In the next section, we describe our query resolution model, QuReTeC, which is the focus of this work.

5.4 Query Resolution

In this section we first describe how we model query resolution as term classification (Section 5.4.1), then present our query resolution model, QuReTeC, (Section 5.4.2), and finally describe how we generate distant supervision labels for the model (Section 5.4.3).

[3] We set $k = 60$ and do not tune it.

5.4.1 Query resolution as term classification in the conversation history

Previous work has modeled query resolution as a sequence to sequence task [58, 96], where the source sequence is $q_{1:i}$ and the target sequence is q_i^*, where q_i^* is a gold standard resolution of the current turn query q_i. For instance, the gold standard resolution of turn #4 in Table 5.1 is: "When was Saosin's first album released?"

However, since (i) the initial retrieval step of our pipeline (Section 5.3.2) is a term-based model that treats queries as bag of words, and (ii) the supervised neural ranker we use in the re-ranking step (Section 5.3.2) is robust to queries that are not well-formed natural language texts [203], our query resolution model does not necessarily need to output a well-formed natural language query but rather a set of terms to expand the query. Besides, sequence to sequence based models generally need a massive amount of data for training in order to get reasonable performance due to their generation objective [59]. Therefore, we model query resolution as a term classification task: given the conversation history $q_{1:i-1}$ and the current turn query q_i, output a binary label (relevant or non-relevant) for each term in $q_{1:i-1}$. Terms in the conversation history $q_{1:i-1}$ that are tagged as relevant are appended to the current turn query q_i to form the predicted current turn query resolution $\hat{q}_i$.

We define the set of relevant resolution terms $E^*(q_i)$ as:

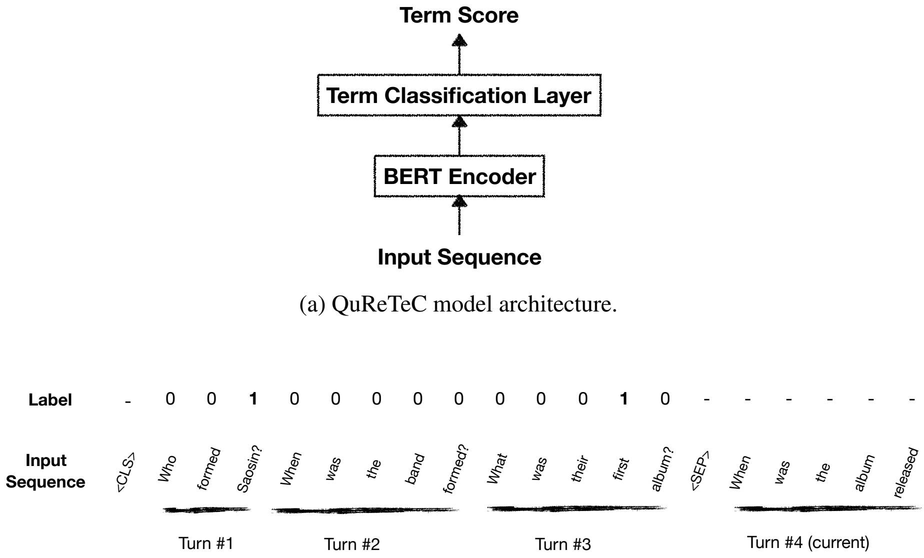
$$E^*_{q_i} = terms(q_i^*) \cap terms(q_{1:i-1}) \setminus terms(q_i), \tag{5.2}$$

where q_i^* is a gold standard resolution of the current turn query q_i. Under this formulation, the set of relevant terms $E^*_{q_i}$ represents the missing context from the conversation history $q_{1:i-1}$. For instance, the set of gold standard resolution terms $E^*_{q_i}$ for turn #4 in Table 5.1 is $\{$Saosin, first$\}$. Note that $E^*_{q_i}$ can be empty if $q_i = q_i^*$, i.e., the current turn query does not need to be resolved, or if $terms(q_i^*) \cap terms(q_{1:i-1})$ is empty. In our experiments $terms(q_i^*) \cap terms(q_{1:i-1}) \approx terms(q_i^*)$, and therefore almost all the gold standard resolution terms can be found in the conversation history.

5.4.2 Query resolution model

In this section, we describe our query resolution model, QuReTeC. Figure 5.2a shows the model architecture of QuReTeC. Each term in the input sequence is first encoded using bidirectional transformers [182] – more specifically BERT [50]. Then, a term classification layer takes each encoded term as input and outputs a score for each term. We use BERT as the encoder since it has been successfully applied in tasks similar to ours, such as named entity recognition and coreference resolution [50, 89, 108]. Next we describe the main parts of QuReTeC in detail, i.e., input sequence, BERT encoder and Term classification layer.

1. *Input sequence.* The input sequence consists of all the terms in the queries of the previous turns $q_{1:i-1}$ and the current turn q_i. It is formed as: [`<CLS>`, $terms(q_1)$, $\ldots$, $terms(q_{i-1})$, `<SEP>`, $terms(q_i)$], where `<CLS>` and `<SEP>` are special tokens. We add a special separator token `<SEP>` between the previous turn q_{i-1} and the current turn q_i in order to inform the model where the current turn begins. Figure 5.2b shows an example input sequence and the gold standard term labels.


Term Score
Term Classification Layer
BERT Encoder
Input Sequence
Label
- 0 0 1 0 0 0 0 0 0 0 1 0 - - - - - -
Input Sequence
<CLS> Who formed Saosin? When was the band formed? What was their first album? <SEP> When was the album released
Turn #1
Turn #2
Turn #3
Turn #4 (current)

(a) QuReTeC model architecture.

(b) Example input sequence and gold standard term labels (1: relevant, 0: non-relevant) for QuReTeC.

Figure 5.2

2. *BERT encoder.* BERT first represents the input terms with WordPiece embeddings using a 30K vocabulary. After applying multiple transformer blocks, BERT outputs an encoding for each term. We refer the interested reader to the original paper for a detailed description of BERT [50].
3. *Term classification layer.* The term classification layer is applied on top of the representation of the first sub-token of each term [50]. It consists of a dropout layer, a linear layer and a sigmoid function and outputs a scalar for each term. We mask out the output of `<CLS>` and the current turn terms, since we are not interested in predicting a label for those (see Equation (5.2) for the definition and Figure 5.2b for an example).

In order to train QuReTeC we need a dataset containing gold standard resolution terms $E^*_{q_i}$ for each q_i. The terms in $E^*_{q_i}$ are labeled as relevant and the rest of the terms $(terms(q_{1:i-1}) \setminus E^*_{q_i})$ as non-relevant. Assuming there exists a gold standard resolution q^*_i for each q_i, we can derive $E^*_{q_i}$ using Equation (5.2). We use standard binary cross entropy as the loss function.

5.4.3 Generating distant supervision for query resolution

Recall that the gold standard resolution q^*_i includes the information in q_i and the missing context of q_i that exists in the conversation history $q_{1:i-1}$. As described above, we can train QuReTeC if we have a gold standard resolution q^*_i for each q_i. Obtaining such special-purpose gold standard resolutions is cumbersome compared to almost readily available general-purpose passage relevance labels for q_i. We propose a distant supervision method to generate labels to train QuReTeC. Specifically, we simply replace q_i^* with a relevant passage $p_{q_i}^*$ in Equation (5.2) to extract the set of relevant resolution terms $E_{q_i}^*$. Table 5.1 illustrates this idea with an example dialogue and the relevant passage to the current turn query. The gold standard resolution terms extracted with this distant supervision procedure for this example are $\{\text{Saosin, first, band}\}$.

Intuitively, the above procedure is noisy and can result in adding terms to $E_{q_i}^*$ that are non-relevant, or adding too few relevant terms to $E_{q_i}^*$. Nevertheless, we experimentally show in Section 5.6.2 that this distant supervision signal can be used to substantially reduce the number of human-curated gold standard resolutions required for training QuReTeC.

The distant supervision method we describe here makes QuReTeC more generally applicable than other supervised methods such as the method in Elgohary et al. [58] that can only be trained with gold standard query resolutions. This is because, apart from manual annotation, query-passage relevance labels can be potentially obtained at scale by using click logs [88], or weak supervision [49].

5.5 Experimental Setup

5.5.1 Research questions

We aim to answer **RQ4**, which we break down to the following research sub-questions:

RQ4.1 How does the QuReTeC model perform compared to other state-of-the-art methods?

RQ4.2 Can we use distant supervision to reduce the amount of human-curated training data required to train QuReTeC?

RQ4.3 How does QuReTeC's performance vary depending on the turn of the conversation?

For all the research questions listed above we measure performance in both an intrinsic and an extrinsic sense. *Intrinsic* evaluation measures query resolution performance on term classification. *Extrinsic* evaluation measures retrieval performance at both the initial retrieval and the reranking steps.

5.5.2 Datasets

Extrinsic evaluation – retrieval

The TREC CAsT dataset is a multi-turn passage retrieval dataset [46]. It is the only such dataset that is publicly available. Each topic consists of a sequence of queries. The topics are open-domain and diverse in terms of their information need. The topics are curated manually to reflect information seeking conversational structure patterns. Later turn queries in a topic depend only on the previous turn queries, and not on the returned passages of the previous turns, which is a limitation of this dataset. Nonetheless, the dataset is sufficiently challenging for comparing automatic systems, as we will show in Section 5.6.1. Table 5.3 shows statistics of the dataset. The original dataset consists of 30 training and 50 evaluation topics. 20 of 50 topics in the evaluation set were annotated for relevance by NIST assessors on a 5-point relevance scale. We use this set as the

Table 5.3: TREC CAsT 2019 multi-turn passage retrieval dataset statistics.

Split	#Topics	#Queries	#Labelled passages per topic	#Relevant passages per topic	#Labelled passages per query	#Relevant passages per query
Test	20	173	1,467.50 ± 252.86	406.00 ± 190.18	169.65 ± 36.69	46.94 ± 31.53

Table 5.4: Query resolution datasets statistics. In the Split column, we indicate the where the positive term labels originate from: either gold (gold standard resolutions) or distant (Section 5.4.3).

Dataset	Split	#Queries	#Terms (per query)	
			Total	Positive
QuAC	Train (gold)	20,181	97.96 ± 61.02	4.56 ± 3.88
	Train (distant)	31,538	99.78 ± 62.36	6.90 ± 5.59
	Dev (gold)	2,196	95.49 ± 58.79	4.49 ± 3.90
	Test (gold)	3,373	96.96 ± 59.24	4.30 ± 3.86
CAsT	Test (gold)	153	39.97 ± 17.97	1.89 ± 1.62

TREC CAsT test set. The organizers also provided a small set of judgements for the training set, however we do not use it in our pipeline. The passage collection is the union of two passage corpora, the MS MARCO [127] (Bing), and the TREC CAR [53] (Wikipedia passages).[4]

Intrinsic evaluation – query resolution

The original QuAC dataset [35] contains dialogues on a single Wikipedia article section regarding people (e.g., early life of a singer). Each dialogue contains up to 12 questions and their corresponding answer spans in the section. It was constructed by asking two crowdworkers (a student and a teacher) to perform an interactive dialogue about a specific topic. Elgohary et al. [58] crowdsourced question resolutions for a subset of the original QuAC dataset [35]. All the questions in the *dev* and *test* splits of [58] have gold standard resolutions. We use the *dev* split for early stopping when training QuReTeC and evaluate on the *test* set. When training with gold supervision (gold standard query resolutions), we use the *train* split from [58], which is a subset of the train split of [35]; all the questions therein have gold standard resolutions. Since QuAC is not a passage retrieval collection, in order to obtain distant supervision labels (Section 5.4.3), we use a window of 50 characters around the answer span to extract passage-length texts, and we treat the extracted passage as the relevant passage. When training with distant labels, we use the part of the *train* split of [35] that does not have gold standard resolutions.

The TREC CAsT dataset [46] also contains gold standard query resolutions for its test set. However, it is too small to train a supervised query resolution model, and we only use it as a complementary *test* set.

The two query resolution datasets described above have three main differences. First, the conversations in QuAC are centered around a single Wikipedia article section about people whereas the conversations in CAsT are centered around an arbitrary topic. Second, the answers of the QuAC questions are spans in the Wikipedia section whereas the CAsT queries have relevant passages that originate from different Web resources besides Wikipedia. Third, later turns in QuAC do depend on the answers in previous

[4] The Washington Post collection was also part of the original collection but it was excluded from the official TREC evaluation process and therefore we do not use it.

turns, while in CAsT they do not (Section 5.3.1). Interestingly, in Section 5.6.1 we demonstrate that despite these differences, training QuReTeC on QuAC generalizes well to the CAsT dataset.

Table 5.4 provides statistics for the two datasets.[5] First, we observe that the QuAC dataset is much larger than CAsT. Also, QuAC has a larger number of terms on average than CAsT ($\sim$97 vs $\sim$40) and a larger negative-positive ratio ($\sim$20:1 vs $\sim$40:1). This is because in QuAC the answers to the previous turns are included in the conversation history whereas in CAsT they are not. For this reason, we expect query resolution on QuAC to be more challenging than on CAsT.

5.5.3 Evaluation metrics

Extrinsic evaluation – retrieval

We report NDCG@3 (the official TREC CAsT evaluation metric), Recall, MAP, and MRR at rank 1000. We also provide performance metrics averaged per turn to show how retrieval performance varies across turns.

We report on statistical significance with a paired two-tailed t-test. We depict a significant increase for $p < 0.01$ as $^{\blacktriangle}$.

Intrinsic evaluation – query resolution

We report on Micro-Precision (P), Micro-Recall (R) and Micro-F1 (F1), i.e., metrics calculated per query and then averaged across all turns and topics. We ignore queries that are the first turn of the conversation when calculating the mean, since we do not predict term labels for those.

5.5.4 Baselines

We perform intrinsic and extrinsic evaluation by comparing against a number of query resolution baselines. Next, we provide a detailed description of each baseline:

- **Original** This method uses the original form of the query. We explore different variations for constructing $\hat{q}_i$: (1) current turn only (cur), (2) current turn expanded by the previous turn (cur+prev), (3) current turn expanded by the first turn (cur+first), and (4) all turns.
- **RM3 [2]** A state-of-the-art unsupervised pseudo-relevance feedback model.[6] RM3 first performs retrieval and treats the top-n ranked passages as relevant. Then, it estimates a query language model based on the top-n results, and finally adds the top-k terms to the original query. As with Original, we report on different variations for constructing the query: cur, cur+prev, cur+first and all turns. In order to apply RM3 for query resolution we append the top-k terms to the original query q_i to obtain $\hat{q}_i$.

[5] Note that the first turn in each topic does not need query resolution because there is no conversation history at that point and thus the query resolution CAsT test has 20 (the number of topics) fewer queries than in Table 5.3.

[6] Note that given the very small size of the TREC CAsT training set we do not compare to more sophisticated yet data-hungry pseudo-relevance feedback models such as [130].

- **NeuralCoref**[7] A coreference resolution method designed for chatbots. It uses a rule-based system for mention detection and a feed-forward neural network that predicts coreference scores. We perform coreference resolution on the conversation history $q_{1:i-1}$ and the current turn query q_i. The output $\hat{q}_i$ consists of q_i and the predicted terms in $q_{1:i-1}$ where terms in q_i refer to.
- **BiLSTM-copy [58]** A neural sequence to sequence model for query resolution. It uses a BiLSTM encoder and decoder augmented with attention and copy mechanisms and also a coverage loss [162]. It initializes the input embeddings with pretrained GloVe embeddings.[8] Given $q_{1:i-1}$ and q_i, it outputs $\hat{q}_i$. It was optimized on the QuAC gold standard resolutions.

Intrinsic evaluation – query resolution

In order to perform intrinsic evaluation on the aforementioned baselines, we take the query resolution they output ($\hat{q}_i$) and apply Equation (5.2) by replacing q_i^* with $\hat{q}_i$ to obtain the set of predicted resolution terms.

Extrinsic evaluation – initial retrieval

Here, apart from the aforementioned baselines, we also use the following baselines:

- **Nugget** [69]. Extracts substrings from the current and previous turn queries to build a new query for the current turn.[9]
- **QCM** [201]. Models the edits between consecutive queries and the results list returned by the previous turn query to construct a new query for the current turn.
- **Oracle** Performs initial retrieval using the gold standard resolution query. Released by the TREC CAsT organizers.

Extrinsic evaluation – reranking

Since developing specialized rerankers for multi-turn passage retrieval is not the focus of this chapter, we evaluate the reranking step using ablation studies. For reference, we also report on the performance of the top-ranked TREC CAsT 2019 systems [46]:

- **TREC-top-auto** Uses an automatic system for query resolution and BERT-large for reranking.
- **TREC-top-manual** Uses the gold standard query resolution and BERT-large for reranking.

5.5.5 Implementation & hyperparameters

Multi-turn passage retrieval We index the TREC CAsT collections using Anserini with stopword removal and stemming.[10] In the initial retrieval step (section 5.3.2) we retrieve the top 1000 passages using QL with Dirichlet smoothing (we set $\mu =$

[7] `https://medium.com/huggingface/state-of-the-art-neural-coreference-resolution-for-chatbots-3302365dcf30`

[8] `https://nlp.stanford.edu/projects/glove/`

[9] We use the nugget version that does not depend on anchors text since they are not available in our setting.

[10] `https://github.com/castorini/anserini`

2500). We use the default value for the fusion parameter $k = 60$ [41] in Eq. (5.1). In the reranking step (section 5.3.2) we use a PyTorch implementation of BERT for retrieval [112]. We use the `bert-base-uncased` pretrained BERT model. We fine-tune the BERT reranker with MSMARCO passage ranking dataset [14]. We train on 100K randomly sampled training triples from its training set and evaluate on 100 randomly sampled queries of its development set. We use the Adam optimizer with a learning rate of 0.001 except for the BERT layers for which we use a learning rate of $3\text{e}{-6}$. We apply dropout with a probability of 0.2 on the output linear layer. We apply early stopping on the development set with a patience of 2 epochs based on MRR.

Query resolution We use the `bert-large-uncased` model. We implement QuReTeC on top of HuggingFace's PyTorch implementation of BERT.[11] We use the Adam optimizer and tune the learning rate in the range $\{2\text{e}{-5}, 3\text{e}{-5}, 3\text{e}{-6}\}$. We use a batch size of 4 and do gradient clipping with the value of 1. We apply dropout on the term classification layer and the BERT layers in the range $\{0.1, 0.2, 0.3, 0.4\}$. We optimize for F1 on the QuAC dev (gold) set.

Baselines For RM3, we tune the following parameters: $n \in \{3, 5, 10, 20, 30\}$ and $k \in \{5, 10\}$ and set the original query weight to the default value of 0.8. For Nugget, we set $k_{snippet} = 10$ and tune $\theta \in \{0.95, 0.97, 0.99\}$. For QCM, we tune $\alpha \in \{1.0, 2.2, 3.0\}$, $\beta \in \{1.6, 1.8, 2.0\}$, $\epsilon \in \{0.06, 0.07, 0.08\}$ and $\delta \in \{0.2, 0.4, 0.6\}$. For both Nugget and QCM we use Van Gysel et al. [181]'s implementation. For fair comparison, we retrieve over the whole collection rather than just reranking the top-1000 results. The aformentioned methods are tuned on the small annotated training set of TREC CAsT. For query resolution, we tune the greedyness parameter of NeuralCoref in the range $\{0.5, 0.75\}$. We use the model of BiLSTM-copy released by [58], as it was optimized specifically for QuAC with gold standard resolutions.

Preprocessing We apply lowercase, lemmatization and stopword removal to q_i^*, $q_{1:i-1}$ and q_i using Spacy[12] before calculating term overlap in Equation 5.2.

5.6 Results & Discussion

In this section we present and discuss our experimental results.

5.6.1 Query resolution for multi-turn retrieval

In this subsection we answer **RQ4.1**: we study how QuReTeC performs compared to other state-of-the-art methods when evaluated on term classification (Section 5.6.1), when incorporated in the initial retrieval step (Section 5.6.1) and in the reranking step (Section 5.6.1).

[11] `https://github.com/huggingface/transformers`

[12] `http://spacy.io/`

Table 5.5: Intrinsic evaluation for query resolution on the QuAC test set. Cur, prev, first and all refer to using the current, previous, first or all turns respectively.

Method	**P**	**R**	**F1**
Original (cur+prev)	22.3	46.4	30.1
Original (cur+first)	41.1	49.5	44.9
Original (all)	12.3	**100.0**	21.9
NeuralCoref	65.5	30.0	41.2
BiLSTM-copy	67.0	53.2	59.3
QuReTeC	**71.5**	66.1	**68.7**

Table 5.6: Intrinsic evaluation for query resolution on the TREC CAsT test set. Cur, prev, first and all refer to using the current, previous, first, or all turns respectively.

Method	**P**	**R**	**F1**
Original (cur+prev)	32.5	43.9	37.4
Original (cur+first)	43.0	74.0	54.4
Original (all)	18.6	**100.0**	31.4
RM3 (cur)	35.8	8.3	13.5
RM3 (cur+prev)	34.6	32.5	33.5
RM3 (cur+first)	40.9	32.9	36.5
RM3 (all)	41.5	38.8	40.1
NeuralCoref	**83.0**	28.7	42.7
BiLSTM-copy	51.5	36.0	42.4
QuReTeC	77.2	79.9	**78.5**

Intrinsic evaluation

In this experiment we evaluate query resolution as a term classification task.[13] Table 5.5 shows the query resolution results on the QuAC dataset. We observe that QuReTeC outperforms all the variations of Original and the NeuralCoref by a large margin in terms of F1, precision and recall – except for Original (all) that has perfect recall but at the cost of very poor precision. Also, QuReTeC substantially outperforms BiLSTM-copy on all metrics. Note that BiLSTM-copy was optimized on the same training set as QuReTeC (see Section 5.5.5). This shows that QuReTeC is more effective in finding missing contextual information from previous turns.

Table 5.6 shows the query resolution results on the CAsT dataset. Generally, we observe similar patterns in terms of overall performance as in Table 5.5. Interestingly, we observe that QuReTeC generalizes very well to the CAsT dataset (even though it

[13] Note that the performance of Original (cur) is zero by definition when using the current turn only (see Eq. 5.2). Thus, we do not include it in Tables 5.5 and 5.6. Also, RM3 is not applicable in Table 5.5 since QuAC is not a retrieval dataset.

Table 5.7: Initial retrieval performance on the TREC CAsT test set for different query resolution methods. The retrieval model is fixed (same as in Section 5.3.2). Significance is tested against RM3 (cur+first) since it has the best NDCG@3 among the baselines.

Method	Recall	MAP	MRR	NDCG@3
Original (cur)	0.438	0.129	0.310	0.155
Original (cur+prev)	0.572	0.181	0.475	0.235
Original (cur+first)	0.655	0.214	0.561	0.282
Original (all)	0.694	0.190	0.552	0.256
RM3 (cur)	0.440	0.140	0.320	0.158
RM3 (cur+prev)	0.575	0.200	0.482	0.254
RM3 (cur+first)	0.656	0.225	0.551	0.300
RM3 (all)	0.666	0.195	0.544	0.266
Nugget	0.426	0.101	0.334	0.145
QCM	0.392	0.091	0.317	0.127
NeuralCoref	0.565	0.176	0.423	0.212
BiLSTM-copy	0.552	0.171	0.403	0.205
QuReTeC	**0.754**$^\blacktriangle$	**0.272**$^\blacktriangle$	**0.637**$^\blacktriangle$	**0.341**$^\blacktriangle$
Oracle	0.785	0.309	0.660	0.361

was only trained on QuAC) and outperforms all the baselines in terms of F1 by a large margin. In contrast, BiLSTM-copy fails to generalize and performs worse than Original (cur+first) in terms of F1. NeuralCoref has higher precision but much lower recall compared to QuReTeC. Finally, RM3 has relatively poor query resolution performance. This indicates that pseudo-relevance feedback is not suitable for the task of query resolution.

Query resolution for initial retrieval

In this experiment, we evaluate query resolution when incorporated in the initial retrieval step (Section 5.3.2). We compare QuReTeC to the baseline methods in terms of initial retrieval performance. Table 5.7 shows the results. First, we observe that QuReTeC outperforms all the baselines by a large margin on all metrics. Also, interestingly, QuReTeC achieves performance close to the one achieved by the Oracle performance (gold standard resolutions). Note that there is still plenty of room for improvement even when using Oracle, which indicates that exploring other ranking functions for initial retrieval is a promising direction for future work. QuReTeC outperforms all Original and RM3 variations, which perform similarly. The session search methods (Nugget and QCM) perform poorly even compared to the Original variations, which indicates that session search is different in nature than conversational search. BiLSTM-copy performs poorly compared to QuReTeC but also compared to the Original variations, which means that it does not generalize well to CAsT.

Table 5.8: Reranking performance on the TREC CAsT test set. All the methods in the first group use QuReTeC for query resolution. Significance is tested against BERT-base.

Method	MAP	MRR	NDCG@3
Initial	0.272	0.637	0.341
BERT-base	0.272	0.693	0.408
RRF (Initial + BERT-base)	**0.355**$^{\blacktriangle}$	**0.787**$^{\blacktriangle}$	**0.476**$^{\blacktriangle}$
Oracle	0.754	0.956	0.926
TREC-top-auto	0.267	0.715	0.436
TREC-top-manual	0.405	0.879	0.589

Query resolution for reranking

In this experiment, we study the effect of QuReTeC when incorporated in the reranking step (Section 5.3.2). We keep the initial ranker fixed for all QuReTeC models. Table 5.8 shows the results. First, we see that BERT-base improves over the initial retrieval model that uses QuReTeC for query resolution on the top positions (second line). Second, when we fuse the ranked listed retrieved by BERT-base and the ranked list retrieval by the initial retrieval ranker using RRF, we significantly outperform BERT-base on all metrics (third line). This shows that the two rankers can be effectively combined with RRF, which is a very simple fusion method that only has one parameter which we do not tune. We also see that our best model outperforms TREC-top-auto on all metrics. Furthermore, by comparing RRF (line 3) to Oracle (line 4) we see that there is still plenty of room for improvement for reranking, which is a clear direction for future work. This also shows that the TREC CAsT dataset is sufficiently challenging for comparing automatic systems. Note that TREC-top-manual uses the gold standard query resolutions and is thereby not directly comparable with the rest of the methods.

5.6.2 Distant supervision for query resolution

In this section we answer **RQ4.2**: Can we use distant supervision to reduce the amount of human-curated query resolution data required to train QuReTeC? Figure 5.3 shows the query resolution performance when training QuReTeC under different settings (see figure caption for a more detailed description). For QuReTeC (distant full & gold partial) we first pretrain QuReTeC on distant and then resume training with different fractions of gold. First, we see that QuReTeC performs competitively with BiLSTM-copy even when it does not use any gold resolutions (distant full).[14] More importantly, when only trained on distant, QuReTeC performs remarkably well in the low data regime. In fact, it outperforms BiLSTM-copy (trained on gold) even when using a surprisingly low number of gold standard query resolutions (200, which is $\sim$1% of gold). Last, we see that as we add more labelled data, the effect of distant supervision becomes smaller.

[14] Also, when trained with distant full, QuReTeC performs better than an artificial method that uses the label of the distant supervision signal as the prediction in terms of F1 (56.5 vs 41.6). This is in line with previous work that successfully uses noisy supervision signals for retrieval tasks [49, 187].

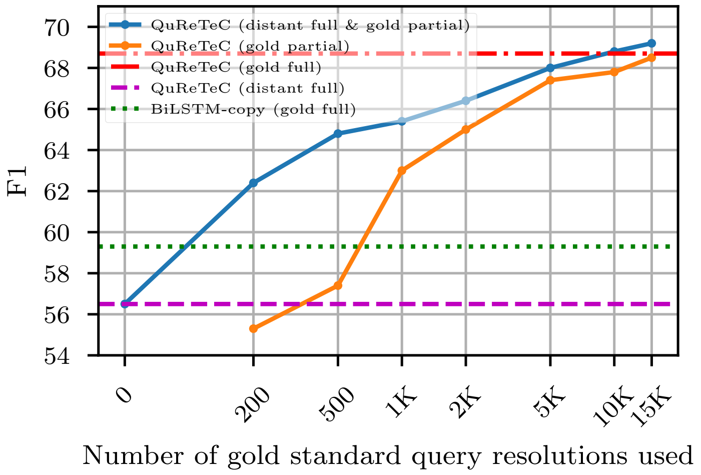


Figure 5.3: Query resolution performance (intrinsic) on the QuAC test set on different supervision settings. Gold refers to the QuAC train (gold) dataset and distant refers to the QuAC train (distant) dataset. Full refers to the whole and partial refers to a part of the corresponding dataset (gold or distant). The x-axis is plotted in log-scale.

This is expected and is also the case for the model trained on QuAC train (gold).[15]

In order to test whether our distant supervision method can be applied on different encoders, we performed an additional experiment where we replaced BERT with a simple BiLSTM as the encoder in QuReTeC. Similarly to the previous experiment, we observed a substantial increase in F1 when retraining with 2K gold standard resolutions (+12 F1) over when only using gold resolutions.

In conclusion, our distant supervision method can be used to substantially decrease the amount of human-curated training data required to train QuReTeC. This is especially important in low resource scenarios (e.g. new domains or languages), where large-scale human-curated training data might not be readily available.

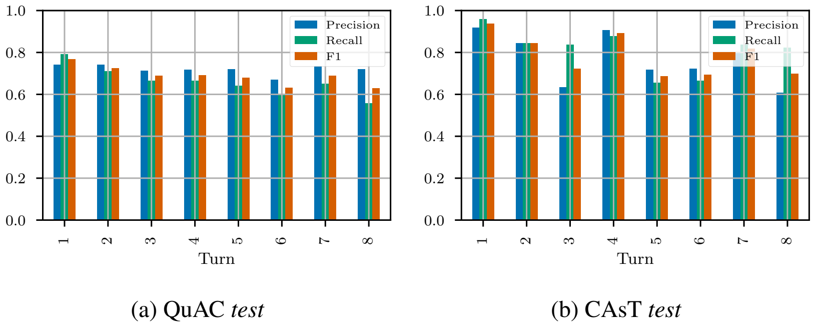
5.6.3 Analysis

In this section we perform analysis on QuReTeC when trained with gold standard supervision.

Query resolution performance per turn

Here we answer **RQ4.3** by analyzing the robustness of QuReTeC at later conversation turns. We expect query resolution to become more challenging as the conversation history becomes larger (later in the conversation).

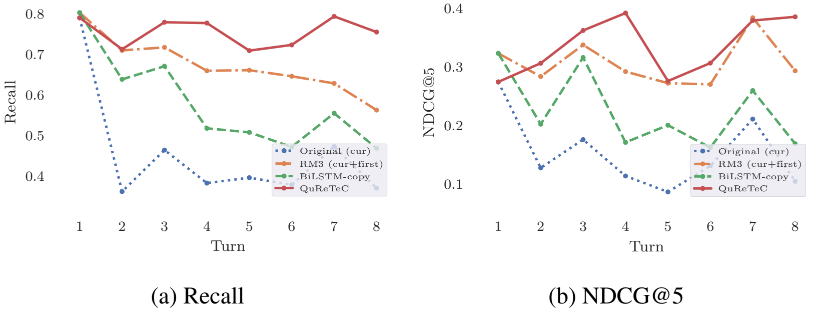
[15] In fact (not shown in Figure 5.3), performance stabilizes after 15K query resolutions (∼75% of gold full).



(a) QuAC *test*

(b) CAsT *test*

Figure 5.4: Intrinsic query resolution evaluation (term classification performance) for QuReTeC, averaged per turn.



(a) Recall

(b) NDCG@5

Figure 5.5: Initial retrieval performance per turn for different query resolution methods CAsT *test*

Intrinsic Figure 5.4 shows the QuReTeC performance averaged per turn on the QuAC and CAsT datasets. Even though performance decreases towards later turns as expected, we observe that it decreases very gradually, and thus we can conclude that QuReTeC is relatively robust across turns.

Extrinsic – initial retrieval Figure 5.5 shows the performance of different query resolution methods when incorporated in the initial retrieval step. We observe that QuReTeC is robust to later turns in the conversation, whereas the performance of all the baseline models decreases faster (especially in terms of recall). For reranking, we observe similar patterns as with initial retrieval; we do not include those results for brevity.

Qualitative analysis

Here we perform qualitative analysis by sampling specific instances from the data. **Intrinsic** Table 5.9 shows one success and one failure case for QuReTeC from the QuAC dev set. In the success case (top) we observe that QuReTeC succeeds in resolving "she" → {"Bipasha", "Basu"} and "reviews" → "Anjabee". Note that "Anjabee" is a movie

Table 5.9: Qualitative analysis for QuReTeC on query resolution (intrinsic). We denote true positive terms with underline and false negative terms in italics. The examples are sampled from the QuAC dev set.

Success case – no mistakes
Q1: What was Bipasha Basu's debut? A1: In 2001, Basu finally made her debut opposite Akshay Kumar in Vijay Galani 's Ajnabee. Q2: Did this help her become well known? A2: It was a moderate box-office success and attracted unfavorable reviews from critics. Q3 (current): Why did she receive unfavorable reviews?
Failure case – misses two relevant terms: *dehusking*, *machine*
Q1: How old was Alexander Graham Bell when he made his first invention? A1: The age of 12. Q2: What did he invent? A2: Bell built a homemade device that combined rotating paddles with sets of nail brushes. Q3: What was it for? A3: A simple *dehusking machine*. Q4 (current): By inventing this, what happened to allow him to continue inventing things?

in which Basu acted but is not mentioned explicitly in the current turn. In the failure case (bottom) we observe that QuReTeC succeeds in resolving "him" → {"Alexander", "Graham" "Bell"} but misses the connection between "this" and "dehusking machine".

Extrinsic – initial retrieval Table 5.10 shows an example from the CAsT test set where QuReTeC succeeds and RM3 (cur+first), the best performing baseline for initial retrieval, fails. First, note that a topic change happens at Q7 (the topic changes from general real-time databases to Firebase DB). We observe that QuReTeC predicts the correct terms, and a relevant passage is retrieved at the top position. In contrast, RM3 (cur+first) fails to detect this topic change and therefore an irrelevant passage is retrieved at the top position that is about real-time databases on mobile apps but not about Firebase DB.

5.7 Conclusion

In this chapter, we studied the task of query resolution for conversational search. We proposed to model query resolution as a binary term classification task: whether to add terms from the conversation history to the current turn query. We proposed QuReTeC, a neural query resolution model based on bidirectional transformers. We proposed a distant supervision method to gather training data for QuReTeC. We found that QuReTeC significantly outperforms multiple baselines of different nature and is robust

Table 5.10: Qualitative analysis for initial retrieval (extrinsic) when using QuReTeC or RM3 (cur+first) for query resolution. The example is sampled from the TREC CAsT dataset.

Q1: What is a real-time database? Q2: How does it differ from traditional ones? Q3: What are the advantages of real-time processing? Q4: What are examples of important ones? Q5: What are important applications? Q6: What are important cloud options? Q7: Tell me about the Firebase DB? Q8 (current): How is it used in mobile apps?
Predicted terms – QuReTeC: {“database”, “firebase”, “db” } **Top-ranked passage – QuReTeC** Firebase is a mobile and web application platform … Firebase’s initial product was a realtime database, … Over time, it has expanded its product line to become a full suite for app development …
Predicted terms – RM3 (cur+first): {“real”, “time”, “database”} **Top-ranked passage – RM3 (cur+first)** There are two options in Jedox to access the central OLAP database and software functionality on mobile devices: Users can access reports through the touch-optimized Jedox Web Server … on their smart phones and tablets.

across conversation turns. Also, we found that our distant supervision method can substantially reduce the required amount of gold standard query resolutions required for training QuReTeC, using only query-passage relevance labels. This result is especially important in low resource scenarios, where gold standard query resolutions might not be readily available.

As for future work, we aim to develop specialized rankers for both the initial retrieval and the reranking steps that incorporate QuReTeC in a more sophisticated way. Also, we want to study how to effectively combine QuReTeC with text generation query resolution methods as well as pseudo-relevance feedback methods. Finally, we aim to explore weak supervision signals for training QuReTeC [49].

In this chapter, we focused on how to improve interactive knowledge gathering and studied multi-turn passage retrieval as an instance of conversational search. In Chapter 6, we focus on a different research theme, namely supporting knowledge exploration for narrative creation.

Part III

Supporting Knowledge Exploration for Narrative Creation

6 News Article Retrieval in Context for Event-centric Narrative Creation

In the third and final part of this thesis we study the research theme of supporting knowledge exploration for narrative creation. In this chapter, we address **RQ5**: Can we support knowledge exploration for event-centric narrative creation by performing news article retrieval in context?

6.1 Introduction

Professional writers such as journalists generate narratives centered around specific events or topics. As shown in recent studies, such writers envision automatic systems that suggest material relevant to the narrative they are creating [51, 83]. This material may provide background information or connections that can help writers generate new angles on the narrative and thus help engage the reader [93].

Previous work has focused on developing automatic systems to support writers explore content relevant to the narrative they are writing about. Such systems use content originating from various sources such as as social media [44, 52, 213], political speeches and conference transcripts [113], or news articles [114].

Writers in the news domain often develop narratives around a single main event, and refer to other, related events that can serve different functions in relation to the narrative [180]. These include explaining the cause or the context of the main event or providing supporting information, among others [37]. Recent work has focused on automatically profiling news article content (i.e., paragraphs or sentences) in relation to their discourse function [37, 206].

In this chapter, instead of profiling existing narratives, we consider a scenario where a writer has generated an incomplete narrative about a specific event up to a certain point, and aims to explore other news articles that discuss relevant events to include in their narrative. A news article that discusses a different event from the past is relevant to the writer's incomplete narrative if it relates to the narrative's main event and to the *narrative's context*. Relevance to the narrative's main event is topical in nature but, importantly, relevance to the narrative's context is not only topical: to be relevant to the

This chapter was published as [190].

Table 6.1: Example incomplete narrative q (consisting of a main event e and context c), and a news article d^* that is relevant to q because it is relevant to both the main event e and to the narrative context c in the sense explained in the main text.

Incomplete narrative q
– Main event (e)
(#1) Malta's armed forces storm merchant ship taken over by rescued migrants.
(#2) Maltese armed forces on Thursday stormed a merchant vessel taken over by rescued migrants who were allegedly demanding to be transported to Europe, rather than back to Libya.
– Narrative context (c)
(#3) In earlier years of Europe's migration crisis—when flows from the Middle East and North Africa were much higher—the Mediterranean was patrolled by Italian and European vessels, as well as by humanitarian groups, which would rescue migrants from flimsy dinghies and transport them to safety, typically to Italy.

Relevant news article (d^*)
(#4) Italy's new government sends immigration message by rejecting rescue ship
(#5) Italy's new populist government has delivered a jolt to European migration politics, prompting a diplomatic standoff with its refusal to accept a rescue vessel overloaded with migrants.

narrative's context, a news article should enable the continuation of the narrative by expanding the narrative discourse [31]. Table 6.1 shows an example of an incomplete narrative and a news article relevant to it. The relevant article discusses an event about a subject mentioned in the narrative context (*Italy*). Here, the relevant news article is relevant to the topic of the incomplete narrative (*migration crisis*) and also relevant to the narrative context in the sense that it is used by the writer to expand the narrative by making a comparison: the previous government of Italy was more welcoming to immigrants than the current. To avoid confusion, in the remainder of this chapter *relevance* without further restriction or scope is taken to mean both *topical relevance* and *relevance to the narrative context*.

We model the problem of finding a relevant news article given an incomplete narrative as a retrieval task where the query is an incomplete narrative and the unit of retrieval is a news article. We automatically generate retrieval datasets for this task by harvesting links from existing narratives manually created by journalists. Using the generated datasets, we analyze the characteristics of this task and study the performance of different rankers on this task. We find that state-of-the-art lexical and semantic rankers are not sufficient for this task and that combining those with a ranker that ranks articles by their reverse chronological order outperforms those rankers alone.

Our main contributions are: (i) we propose the task of news article retrieval in context for event-centric narrative creation; (ii) we propose an automatic retrieval dataset construction procedure for this task; and (iii) we empirically evaluate the performance of different rankers on this task and perform an in-depth analysis of the results to better understand the characteristics of this task.

6.2 Problem Statement

6.2.1 Preliminaries

A *news article* d published at time t consists of its headline H—which introduces the topic of the article [180]—and a sequence of paragraphs $p_1, p_2, \ldots$. Each paragraph p_i consists of a sequence of sentences $a_{i,1}, a_{i,2}, \ldots$.

The *lead paragraph* L of a news article d is its first paragraph p_1, which summarizes the main topic of the article [180].

An *event* e is characterized by interactions between entities such as countries, organizations, or individuals—that deviate from typical interaction patterns [32]. We assume that each news article d is associated with a single main event e.

A *link sentence* $a_{i,j}$ in article d is a sentence that contains a hyperlink to a news article d^*.

A *context* is a sequence of sentences already generated by the writer that introduces a new idea or subtopic in a narrative.

A *query* $q = (e, c, t)$ is an incomplete narrative at time t that consists of an event e and a context c.

6.2.2 Task definition

The task of *news article retrieval in context for event-centric narrative creation* is defined as follows. Given a query $q = (e, c, t)$ and a collection of news articles D published before time t, we need to rank articles in D w.r.t. their relevance to $q = (e, c, t)$. Here, "relevance to e" is to be interpreted as topical, whereas "relevance to c" is not only topical, but it should also enable the continuation of the narrative by expanding the narrative discourse [31]. "Relevance to q" is taken to mean the same as "relevance to e and to c". An article relevant to q can thus be used by the writer to create the next sentence in the yet incomplete narrative. Table 6.1 shows an example query q and a relevant news article d^* published at time $t^* < t$.

6.3 Retrieval Dataset Construction

6.3.1 Dataset construction procedure

In order to construct a retrieval dataset for our news article retrieval task, we rely on existing news articles to simulate incomplete narratives as well as relevant documents. We capitalize on the fact that (complete) news articles often contain links to other news articles manually inserted by journalists in the form of hyperlinks.

The automatic retrieval dataset construction procedure that we propose takes as input a news article d and outputs a set of (q, d^*) pairs, where $q = (e, c, t)$ is a query and d^* is the (unique) relevant news article to q. We assume that the event e associated with d is described by the headline H and the lead paragraph L of d [37].

In order to construct the context c of q, we iteratively look for link sentences $a_{i,j}$ in d that contain a hyperlink to another news article d^*. We enforce $i > 1$ so that the paragraph where the link sentence appears is after the lead paragraph. We also

Table 6.2: Statistics of the retrieval datasets derived from the WaPo and Guardian newspaper collections. Because of the way we construct the retrieval datasets (see Section 6.3.1), each query has a single relevant news article.

Dataset	**Split**	**#** q	**# uniq.** d	**# uniq.** d^*	**Link sentence** $(a_{i,j})$	
					i mean/ median	j mean/ median
WaPo	Train	32,963	23,537	24,279	7.9/7	2.5/2
	Dev.	1,831	1,286	1,585	8.4/8	2.4/2
	Test	1,832	1,216	1,555	9.1/9	2.4/2
Guardian	Train	31,329	21,730	22,935	7.3/6	2.4/2
	Dev.	1,740	1,128	1,526	8.0/7	2.4/2
	Test	1,742	1,064	1,532	7.3/7	2.5/2

enforce $j > 1$ motivated by the fact that links after the first sentence of a paragraph are tightly related to the main idea of the paragraph, therefore the sentences preceding the link sentence can be considered as context [73]. If such a link sentence $a_{i,j}$ exists, we consider the sentences $a_{i,1}, \ldots, a_{i,j-1}$ as the narrative context c and the article d^* as the relevant article for q.

Example To illustrate the procedure described above, consider the example in Table 6.1. Sentences #1 and #2 in Table 6.1 are the headline and lead paragraph of a news article d respectively. Sentence #3 in Table 6.1 is the first sentence $a_{i,j-1}$ of a paragraph p_i, $i > 1$ in d, which constitutes the narrative context c. The link sentence $a_{i,j}$ (not shown in the table) is:

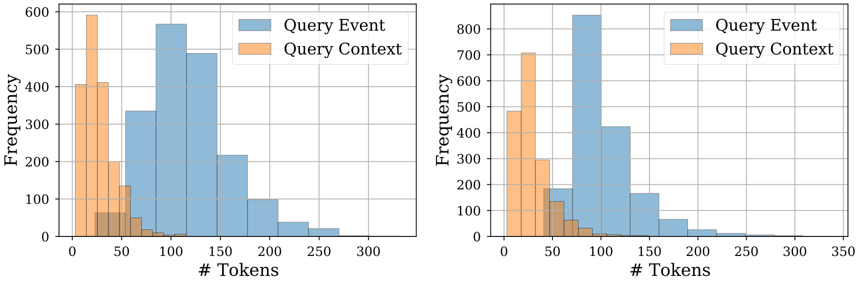
> But over the past year, *Italy has closed its ports* to migrants rescued by humanitarian boats.

where the part in italics is (the anchor text of) a hyperlink to the relevant news article d^* shown in Table 6.1, where sentences #4 and #5 are the headline and lead of d^*, respectively.

6.3.2 Retrieval dataset description

We consider two collections of news articles written in English and published by major newspapers. The first is a set of news articles published by The Washington Post (WaPo), released by the TREC News Track [169]. It contains 671,947 news articles and blog posts published from January 2012 to December 2019. The second is a set of news articles published by The Guardian, between November 2013 to June 2017, which we crawl ourselves. We also crawl the out-links of each article in this set; the final set contains 572,639 news articles published between January 2000 and March 2018.

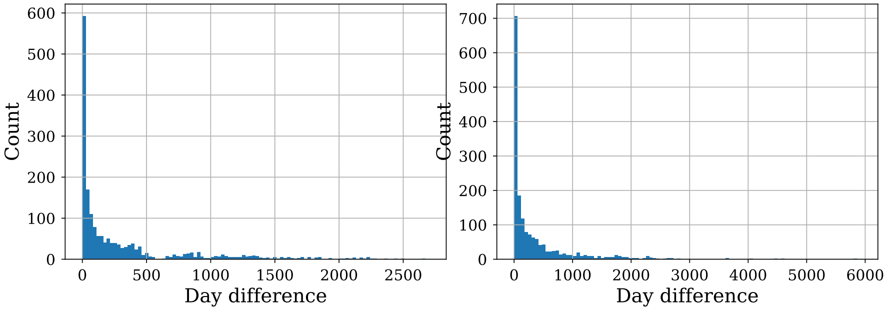
The articles in both newspapers cover multiple genres and domains. In order to ensure that the news articles describe real-world events, we filter out blog posts and opinion news articles, and only keep articles in the following domains: *news*,



(a) WaPo dev.

(b) Guardian dev.

Figure 6.1: Histogram of the number of tokens in the query event e and the query context e.



(a) WaPo dev.

(b) Guardian dev.

Figure 6.2: Histogram of day difference between the query and its relevant news article.

world, *business*, *environment*, *technology*, *society*, *science*, *culture*, *education*, *global*, *healthcare*, *media*, *money*, *teacher*, *local*, *national*. After filtering for genre and domain, we are left with 386,196 articles in WaPo and 185,034 in The Guardian.

We then apply the dataset construction procedure described in Section 6.3.1 to construct a retrieval dataset for both collections. We split the retrieval datasets chronologically and keep the first 90% for training, the next 5% for development, and the last 5% for testing. Table 6.2 shows basic statistics for both retrieval datasets. Figure 6.1 shows a histogram of the number of tokens in the query event e and the query context c. We observe that the query context is shorter than the query event in both datasets. Also, the query event is longer in WaPo than in Guardian because the way those newspapers perform paragraph splitting is different.

Figure 6.2 shows a histogram of the difference in number of days between the publication date of the query and the publication date of the relevant news article on the development sets of the two datasets. The retrieval datasets have a strong recency bias, which is in line with studies on content generation in the news domain [129]. Typical examples of recent, relevant articles are those discussing a previous development of a

Table 6.3: Results of the annotation exercise: assessing relevance of document d^* w.r.t. e only (Task 1), and then c (Task 2). We show the fraction of times the annotator labeled a sample as positive for the task.

Dataset	Task 1	Task 2	Either
WaPo	0.90	0.77	0.91
Guardian	0.85	0.83	0.92

query event or of an event mentioned in the narrative context. And a typical example of a less recent, relevant article can be found when discussing an event that is similar to one mentioned in the query (e.g., an earthquake) but involving different entities (e.g., a person, location, or organization).

6.3.3 Retrieval dataset quality

The dataset construction procedure we described in Section 6.3.1 assumes that an article d^* is relevant to q since the writer has chosen to link to it in a particular context, which is a fair assumption to make. Nevertheless, we further assess the quality of the automatically constructed retrieval datasets with respect to our task definition (Section 6.2.2) by performing two annotation tasks. In the first task, we show e and d^* to a human annotator and ask whether they understand their connection (binary). In the second task, which is done after the completion of the first task, we additionally show the context c and ask whether it enhances their understanding of the connection of e and d^* (binary). The two tasks can help us validate whether d^* is topically relevant to e, and relevant to c in a way that enables the continuation of the narrative (Section 6.2.2).

One annotator annotated 100 examples from the development set of each dataset (i.e., 200 examples in total). In order to assess the quality of the annotations, a second assessor annotated a subset of 50 examples from each dataset (100 examples in total). The Cohen's κ [40] score is 0.61 for Task 1 and 0.50 for Task 2, both of which are considered moderate agreement.

The results can be seen in Table 6.3. We see that, for both datasets, the context c enhances the understanding of the connection to d^* for more than 3/4 of the cases (Task 2). Also, for the vast majority of the cases, either the event e or the context c is sufficient to understand the connection (third column). We conclude that the automatic dataset construction procedure we proposed in Section 6.3.1 can produce reliable datasets for this task.

6.4 Retrieval Method

We follow a standard two-step retrieval pipeline that consists of (1) an unsupervised initial retrieval step and (2) a re-ranking step [192]. Note that we do not focus on proposing new methods but rather on studying existing ones on this novel task.

6.4.1 Initial retrieval

In this step, we score each news article d in D w.r.t. $q = (e, c, t)$ to obtain the initial ranked list L_1. Here, we are interested in achieving high recall at lower depths in the ranking, since this step is followed by a more sophisticated reranking step. We use BM25 [153], an unsupervised lexical matching function, which is effective for ad-hoc retrieval and other tasks, such as question answering [202]. In order to construct the lexical query, we simply concatenate e and c.

6.4.2 Reranking

Here we rerank the initial ranked list L_1 obtained in the previous step by combining the results of multiple rankers using Reciprocal Rank Fusion (RRF), an unsupervised ranking fusion function [41]:

$$\sum_{L \in \mathcal{L}} \frac{1}{k + rank(d, L)}, \tag{6.1}$$

where $\mathcal{L}$ is a set of ranked lists, $rank(d, L)$ is the rank of article d in the ranked list L, and k is a parameter, set to its default value (60).

We use the following rankers:

BM25 The initial retrieval step ranker (Section 6.4.1), often used in combination with more sophisticated ranking models [111].

BERT BERT [50] has recently achieved state-of-the-art performance for retrieval and recommendation tasks in the news domain [198, 203]. BERT has been shown to prefer semantic matches and it is often used in combination with lexical matching ranking functions [141]. Given the query q and a candidate news article d, we follow [113] and construct the input to BERT as follows: [`<CLS>` e `<unused>` c `<SEP>` d], where `<CLS>` is a special token, `<unused>` is a special token that informs the model where the context begins and `<SEP>` is a special token that informs the model where the document d begins. We add a dropout layer on top of the `<CLS>` token, and a linear layer with a scalar output to obtain the final matching score, which is used to rank the articles in L_1. Note that, because of the limit of BERT in the number of tokens, we only take into account the headline and lead of d.

Recency This ranker simply sorts the candidate articles in L_1 by their reversed chronological order.

Note that we have also experimented with combining the scores of the above rankers as features in supervised learning to rank models but they only gave minor improvements over RRF. Thus we do not discuss them in this chapter.

6.5 Experimental setup

6.5.1 Evaluation metrics

We use standard IR metrics: Mean Reciprocal Rank (MRR) and recall at different cut-offs (R@20, R@1000). Note that because of the way we construct our dataset (Section 6.3.1), we only have one relevant news article per query and thus MRR is equivalent to MAP. We use a cut-off of 20 at recall since we expect writers to be willing to navigate the ranked list to lower positions [91]. We report on statistical significance with a paired two-tailed t-test.

6.5.2 Implementation and hyperparameters

We use the BM25 implementation of Anserini [202] with default parameters and retrieve the top-1000 articles (Section 6.4.1).

We use the OpenNIR implementation of BERT for retrieval [111]. We fine-tune the *bert-base* pre-trained model on the training set of each of our datasets separately. We assign a maximum 300 tokens for the query q and 200 for the article d. We use a batch size of 16 with gradient accumulation of 2, we apply max grad norm of 1 and tune the following hyperparameters for MRR on the development set of each dataset separately: number of negatives $\{1, 2, 3\}$ and learning rate $\{5e-6, 1e-5, 2e-5\}$. During training we sample one negative example from the initial ranked list obtained in Section 6.4.1, and train the model with pairwise ranking loss.

Preprocessing and word vectors We use Spacy[1] for sentence splitting, POS tagging and Named Entity Recognition. We use the *en_core_web_lg* model to obtain word vectors.

6.6 Results

In this section we present our experimental results.

6.6.1 Initial retrieval

We examine the performance of the initial retrieval step when different variations of the query q are used. Table 6.4 shows the results. We observe that, for both datasets, when using both the event e and the context c we get better results than when using either of the two alone, especially in terms of R@1000. This shows that both the event e and the context c are important for our task.

In Table 6.4 (bottom row) we also show ranking performance when using the link sentence as the query (see Section 6.3.1). Even though we do not use the link sentence as part of the query in our task definition (Section 6.2.2), this can give us a reference point for the "upper bound" performance in this step, since the link sentence has a high lexical overlap with the relevant article d^* [136]. We observe that, indeed, when using

[1] http://spacy.io/

Table 6.4: Initial retrieval performance of BM25 on the test sets for different variations of the query $q = (e, c, t)$, or the link sentence (LS).

	WaPo		**Guardian**	
Query	MRR	R@1000	MRR	R@1000
e	0.117	0.745	0.104	0.723
c	0.167	0.737	**0.154**	0.714
e & c	**0.172**	**0.832**	0.149	**0.806**
LS	0.459	0.944	0.427	0.929

Table 6.5: Retrieval performance when reranking the ranked list obtained by BM25 (first row).

	WaPo		**Guardian**	
Method	MRR	R@20	MRR	R@20
BM25	0.172	0.433	0.149	0.382
Recency	0.086	0.284	0.065	0.065
BERT	0.182	0.451	0.173	0.447
RRF-recency	0.206	0.509	0.195	0.477
RRF	**0.236**	**0.588**	**0.212**	**0.533**

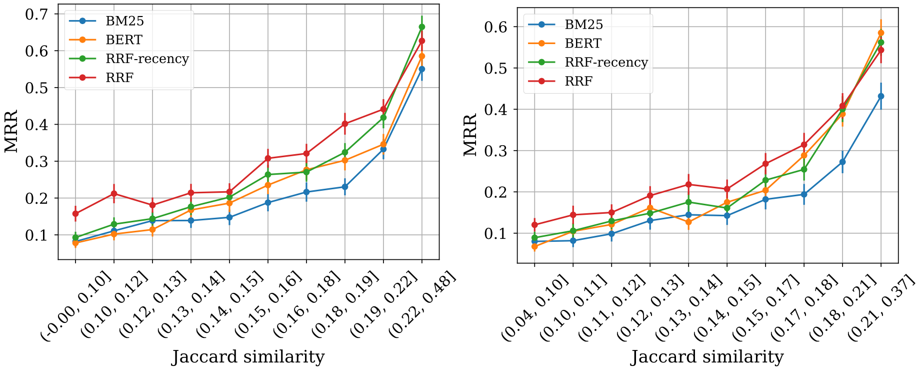
the link sentence as the query, ranking performance is much higher than when using q, achieving close to perfect R@1000. Nevertheless, R@1000 when using e & c is relatively close to when using LS, which is an encouraging result given that in this step we are more interested in recall.

6.6.2 Reranking

Here, we report results on the individual rankers described in Section 6.4.2 and their combinations with RRF. Table 6.5 shows the results. First, we see that the performance of the Recency ranker is poor. Also, we see that BERT outperforms BM25 on both datasets, while only using the headline and the lead of the candidate news article. RRF-recency combines BERT and BM25 achieves an increase over BERT. Finally, when also adding the Recency ranker in RRF, we observe a significant ($p < 0.01$) increase on all metrics. We conclude that RRF, albeit simple, is effective in combining the three rankers and that all three rankers are useful for this task.

6.7 Analysis

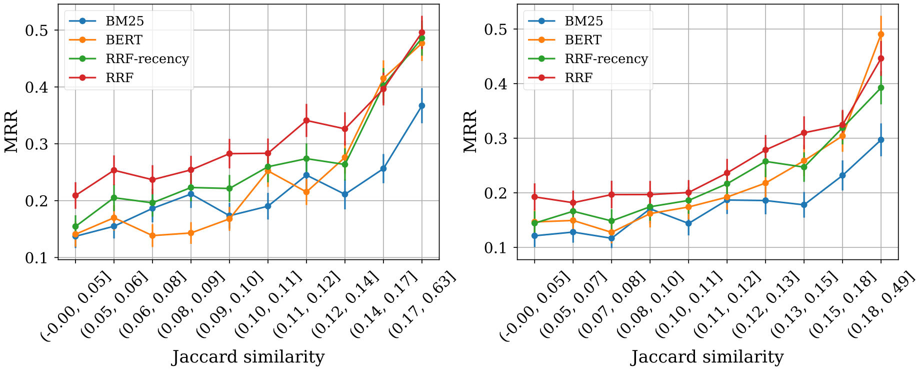
In this section we analyze our results along different dimensions to gain further insights into this task. For our analysis we use the development set of the WaPo and Guardian



(a) WaPo

(b) Guardian

Figure 6.3: MRR vs Jaccard similarity between query q and d^*.



(a) WaPo

(b) Guardian

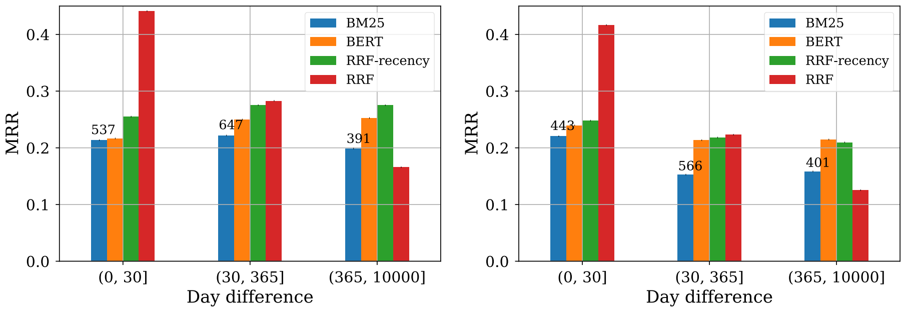
Figure 6.4: MRR vs Jaccard similarity between narrative's context c and d^*.

datasets.

6.7.1 Vocabulary gap

The vocabulary gap is a well known challenge in information retrieval [103]. Here, we analyze the performance of the rankers under comparison for this task based on the vocabulary gap between the query q and the relevant article d^*.

In Figure 6.3 we observe that the higher the lexical overlap between q and d^* (small vocabulary gap) the higher the performance for all rankers, for both datasets. Also, we see that when the lexical overlap is low (large vocabulary gap), all rankers fail to bring the relevant article at the top positions of the ranking. This shows that more sophisticated methods are needed to handle the large vocabulary gap in this task. In Figure 6.4 we show the lexical overlap between the narrative's context c only and the



(a) WaPo

(b) Guardian

Figure 6.5: MRR for retrieval methods grouped per day difference of the query and the relevant article.

relevant d^*. Even though it follows the same trend as in Figure 6.3, we see that BERT is consistently better than BM25 as the term overlap between the narrative's context c and d^* increases, for both datasets. This shows that BERT is able to better take into account the narrative's context c than BM25.

We next show examples of high/low lexical overlap between q and d^* in Table 6.6. In the first example (high lexical overlap), we see that because of high term overlap, all rankers are able to rank d^* at the top 1–2 positions. In the second example (low lexical overlap), the relevant article d^* discusses the execution of Alfredo Prieto: this is a case in which Morrogh, a prosecutor in Virginia, was involved in (Morrogh is mentioned in the narrative's context c). However, the fact that Morrogh is involved in the case is not mentioned explicitly in d^* and thus all rankers fail to rank the relevant article at the top positions. Incorporating the fact that Morrogh is related to Prieto in the ranking model could potentially be achieved by exploiting knowledge graphs that store event information [67, 154]. We leave the exploration towards this direction for future work.

6.7.2 Temporal aspects

As discussed in Section 6.3.1, the retrieval datasets we derived for this task have a strong recency bias. Here, we analyze the performance of the rankers under comparison based on the temporal aspect, i.e., how recent the relevant article is.

In Figure 6.5 we show the performance of the retrieval methods for different day differences between the query q and the relevant article d^*. As expected, we observe that for RRF, which uses the recency signal, the performance increases substantially on average when the relevant article is recent, and decreases when it is older.

We next look at specific examples to better understand the results. Table 6.7 shows examples where the relevant article is recent and RRF ranks it at the top of the ranking, while RRF-recency ranks it lower. In both examples, RRF-recency's top-ranked article seems to also be relevant to q, however the writer chose to refer to a more recent event [129]. Note that the fact that only one article is relevant to each query is an artifact of our dataset and not of the task itself. Table 6.8 shows examples where the relevant

Table 6.6: Examples from the WaPo dev. set with high/low lexical overlap between q and d^* (top/bottom).

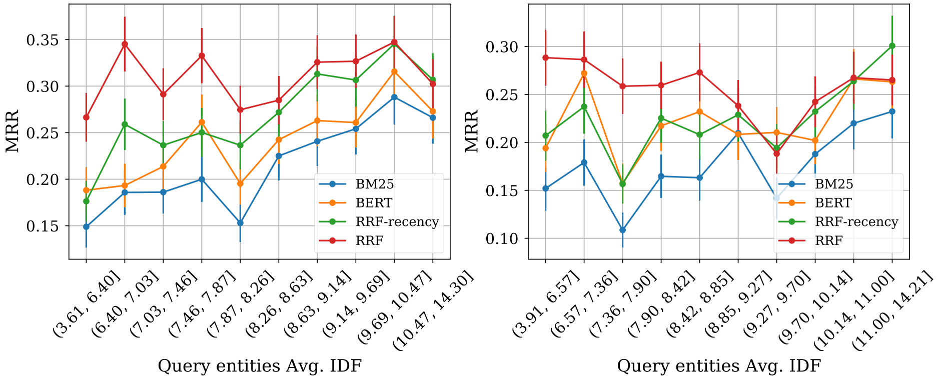
Query q		**Link sentence**	**Relevant article d^***		**Top-ranked article RRF**		**Rank of d^***			
Query event e	**narrative's context c**		Headline & Lead	Day diff.	Headline & Lead	Day diff.	BM25	BERT	RRF-recency	RRF
What 'arrest' means for the Canadians detained in China — and the epic battle over Huawei . BEIJING — Over the past five months, as Beijing and Washington have exchanged fire on trade and technology, two Canadian men have been held in near-isolation in Chinese detention facilities.	Last week, a Chinese court scheduled Schellenberg's appeal hearing to begin hours after Meng faced an extradition hearing in Vancouver.	After a Canadian court pushed back a decision in Meng's case, the Chinese court announced it would delay a ruling on whether Schellenberg would be put to death.	Chinese court delays ruling on Canadian's death sentence appeal. BEIJING — A Chinese court has delayed ruling on a Canadian man's appeal against his death sentence for drug smuggling, just hours after a Canadian court set a September date for the next hearing in an extradition case against a top Chinese executive.	7	d^*		1	2	1	1
Fairfax race for prosecutor puts focus on pace of criminal justice reform. Political races are usually about striking contrasts, but in the first Democratic primary for prosecutor in Virginia's largest county in 55 years, both candidates give themselves the same title: progressive.	Morrogh, who was first elected commonwealth's attorney in 2007, has spent nearly all of his career in the prosecutor's office, where he has won some high-profile cases and avoided major scandal.	Morrogh helped secure the convictions of D.C. sniper Lee Boyd Malvo, serial killer Alfred Prieto and more recently the MS-13 gang members who killed a 15-year-old girl.	The execution of Alfredo Prieto: Witnessing a serial killer's final moments. JARRATT, Va. — It is undeniably disturbing to drive to the scheduled killing of another. A hurricane brewing in the distance, slicing steady rain through the gray day.	1339	Money from PAC funded by George Soros shakes up prosecutor races in Northern Virginia. A political action committee funded by Democratic megadonor and billionaire George Soros has made large contributions to two upstart progressive candidates attempting to unseat Democratic prosecutors in Northern Virginia primary races.	38	371	98	151	252

Table 6.7: Examples from the WaPo dev. set with a recent relevant article where RRF ranks the relevant article at the top, while RRF-Recency ranks it lower.

Query q		Link sentence	Relevant article d^*		Top-ranked article RRF-recency		Rank of d^*			
Query event e	**narrative's context** c		Headline & Lead	Day diff.	Headline & Lead	Day diff.	BM25	BERT	RRF-recency	RRF
China's influence on campus chills free speech in Australia, New Zealand. SYDNEY — Chinese students poured into Australia and New Zealand in the hundreds of thousands over the past 20 years, paying sticker prices for university degrees that made higher education among both countries' top export earners.	After years of feeling fortunate about their economic relationship with China, Australians are starting to worry about the cost.	On Thursday, a ruling party lawmaker, Andrew Hastie, compared China's expansion to the rise of Germany before World War II, suggesting it posed a direct military threat.	Threat from China recalls that of Nazi Germany, Australian lawmaker says. The West's approach to containing China is akin to its failure to prevent Nazi Germany's aggression, an influential Australian lawmaker warned, earning a rebuke from Beijing while highlighting the difficulty the U.S. ally faces in weighing its security needs against economic interests.	3	China's meddling in Australia — and what the U.S. should learn from it. While American attention remains focused on Russia's interference in the 2016 presidential election, Australia — perhaps the United States' closest ally — is debating the designs that a different country altogether has on its political system, economy and public opinion. That country is China.	788	32	38	14	1
Despite national security concerns, GOP leader McCarthy blocked bipartisan bid to limit China's role in U.S. transit. House Minority Leader Kevin McCarthy (R-Calif.) blocked a bipartisan attempt to limit Chinese companies from contracting with U.S. transit systems, a move that benefited a Chinese government-backed manufacturer with a plant in his district, according to multiple people familiar with the matter.	Lawmakers frequently take a stance on legislation that could affect campaign contributors or hometown companies. But McCarthy's intervention was striking because the close ally of President Trump sought to protect Chinese interests at a time when Trump and many lawmakers on Capitol Hill are attempting to curb Beijing's access to U.S. markets, particularly in industries deemed vital to national security.	Just last week, Trump put Chinese telecom giant Huawei on a trade "blacklist" that severely restricts its access to U.S. technology.	Trump administration cracks down on giant Chinese tech firm, escalating clash with Beijing. The Trump administration on Wednesday slapped a major Chinese firm with an extreme penalty that makes it very difficult for it to do business with any U.S. company, a dramatic escalation of the economic clash between the two nations.	5	Trump says he'll spare Chinese telecom firm ZTE from collapse, defying lawmakers. President Trump said late Friday he had allowed embattled Chinese telecommunications giant ZTE Corp. to remain open despite fierce bipartisan opposition on Capitol Hill, defying lawmakers who have warned that the huge technology company should be severely punished for breaking U.S. law.	360	98	9	3	1

Table 6.8: Examples from the WaPo dev. set with an old relevant article where RRF-recency ranks the relevant article at the top, while RRF ranks it lower.

Query q		Link sentence	Relevant article d^*		Top-ranked article RRF		Rank of d^*			
Query event e	**narrative's context** c		Headline & Lead	Day diff.	Headline & Lead	Day diff.	BM25	BERT	RRF-recency	RRF
Max Scherzer's knuckle injury might keep him from being ready for Opening Day. The knuckle at the base of Max Scherzer's right ring finger became the most analyzed joint in the Washington Nationals‘ clubhouse on Thursday, knocking Stephen Strasburg's right elbow out of its familiar spotlight, and delivering an unexpected blow to the early-season stability of the Nationals' rotation.	Scherzer expected the sprain to heal with regular rest in the offseason. But the symptoms did not improve by December, when another MRI exam revealed the fracture.	A month later, the fracture still had not healed, so he told Team USA Manager Jim Leyland he would not be able to pitch in the World Baseball Classic.	Max Scherzer won't pitch in WBC because of stress fracture in finger. Nationals ace Max Scherzer, one of the first and highest-profile players to commit to play for the United States in the upcoming World Baseball Classic, will not participate in the tournament because of "the ongoing rehabilitation stress fracture in the knuckle of his right ring finger," the club announced Monday afternoon in a statement.	927	Nationals place Stephen Strasburg on disabled list (again) with pinched nerve. MIAMI — Nothing appeared amiss for Stephen Strasburg on Wednesday. He played catch at Miller Park in Milwaukee as scheduled, a day before he was to take the mound for the Washington Nationals against the Miami Marlins on Thursday night. But the throwing session didn't go well.	363	12	1	1	3
A mysterious sickness has killed nearly 100 children in India. Could litchi fruit be the cause? NEW DELHI — The children go to sleep as best they can in the sweltering heat. Early in the morning, the fever spikes and the seizures begin.	In August 2017, India witnessed a notorious outbreak of encephalitis in the city of Gorakhpur in the neighboring state of Uttar Pradesh.	More than 30 children died over two days at one hospital after its oxygen ran out.	'It's a massacre': At least 30 children die in Indian hospital after oxygen is cut off. NEW DELHI — One by one, the infants and children slipped away Thursday night, their parents watching helplessly as oxygen supplies at the government hospital ran dangerously low.	674	'It is horrid': India roasts under heat wave with temperatures above 120 degrees. NEW DELHI — When the temperature topped 120 degrees (49 Celsius), residents of the northern Indian city of Churu stopped going outside and authorities started hosing down the baking streets with water.	11	1	1	1	3



(a) WaPo

(b) Guardian

Figure 6.6: MRR vs avg. IDF of the entities in the query q.

article is old and RRF-recency ranks it at the top of the ranking, while RRF ranks it lower. In the first example, the relevant article discusses a development on the injury of Scherzer, a player of the Washington Nationals team, and RRF-recency correctly brings that at the top position. However, RRF ranks a more recent event at the top position that discusses an injury of a different player of the same team. In the second example, RRF brings at the top position an article that discusses an event about India that is more recent than the one that the relevant article discusses, however the article is off-topic.

The above phenomena suggest that more sophisticated methods that model recency should be explored for this task. For instance, it would be interesting to try to predict which queries are of temporal nature based on the characteristics of the underlying collection [90]. However, methods that build on features derived from user interactions are not applicable to our setting [55].

6.7.3 Entity popularity

Entities play a central role in event-centric narratives, especially in the news domain [154]. We examine whether entity popularity affects retrieval performance in our task by measuring the IDF (Inverse Document Frequency) of entities in the query [115]. An entity with a high IDF in the collection is less popular than an entity with a low IDF.

In Figure 6.6 we show the performance depending on the average IDF of the entities in the query in the underlying collection. We observe that the rankers that use the query and article text (BM25, BERT, RRF-recency) perform worse for queries with more popular entities (low IDF) than for queries with less popular entities. This is because popular entities appear in multiple events, and thus there are many potentially relevant articles for a query. We also see that RRF, which takes recency into account, is more robust to entity popularity. This might also be related to the fact that a recent event that involves a popular entity is more likely to be relevant in general than a less recent event that involves the same entity (also see examples in Section 6.7.2, Table 6.7).

6.7.4 Link sentence

Recall that we do not use the link sentence as part of the query (see Section 6.3.1). Thus, our rankers are not aware of its content. However, we found that in some cases the link sentence contains information that is crucial for the connection of the *complete* narrative and the relevant news article. Thus, in such cases, the query event e and the narrative's context c are not sufficient. Table 6.9 shows examples of such cases. Note that in the first example, the relevant article was not even retrieved in the top-1000 of the initial retrieval step (see Section 6.4.1). In the second example, the relevant article is ranked very low by all rankers.

One direction for future work would be to detect parts of the link sentence that contain such crucial information and add them to the narrative's context c. This could be performed as a manual annotation task or modeled as a prediction task [87].

6.8 Related work

6.8.1 Supporting narrative creation

Recent work on developing automatic applications to support writers has focused on designing tools that track and filter information from social media to support journalists [52, 213]. Cucchiarelli et al. [44] track the Twitter stream and Wikipedia edits to suggest potentially interesting topics that relate to a new event that a writer can include in their narrative when reporting on the event. In contrast, instead of relying on external sources, we aim to retrieve news articles that describe events from the past that can help the writer expand the incomplete narrative about a specific event.

Perhaps the closest to our task are the works by Maiden and Zachos [114] and MacLaughlin et al. [113]. Maiden and Zachos [114] focus on suggesting articles that would help journalists discover new, creative angles on a current incomplete narrative. The difference with our work is that they aim to suggest creative angles on articles and retrieve articles depending on the angle the writer selects. In addition, they evaluate their system in a living lab scenario, whereas we create static retrieval datasets from historical data and use them to train ranking functions. Evaluating our system in a living lab scenario would be a promising direction for future work.

MacLaughlin et al. [113] retrieve paragraphs that contain quotes from political speeches and conference transcripts, so that writers can use them in their incomplete narratives. Even though their retrieval task definition is similar to ours, our task differs in that our unit of retrieval is a news article from a large news article collection instead of a paragraphs from a single document (e.g., a political speech). Moreover, our unit of retrieval (article) is timestamped, which makes the temporal aspect prominent in our task.

6.8.2 Context-aware citation recommendation

The task of context-aware citation recommendation is to find articles that are relevant to a specific piece of text a writer has generated [76]. It has mainly been studied in the scientific domain [57, 82, 86, 158], but also in the news domain [102]. The

Table 6.9: Examples from the WaPo dev. set where the link sentence contains crucial information for the connection of the complete narrative and the relevant article.

Query q		Link sentence	Relevant article d^*		Top-ranked article RRF		Rank of d^*			
Query event e	**narrative's context** c		Headline & Lead	Day diff.	Headline & Lead	Day diff.	BM25	BERT	RRF-recency	RRF
Americans are drinking more 'gourmet' coffee. This doesn't mean they're drinking great coffee. The National Coffee Association USA recently dropped its annual survey results, and, as usual, there's a wealth of information to sift through to better understand the state of coffee drinking in America.	According to this year's finding, coffee remains the No. 1 drink: Sixty-three percent of the respondents said they drank a coffee beverage (drip coffee, espresso, latte, cold brew, Unicorn Frappuccino, etc.) the previous day, a click down from 64 percent in 2018.	By the way, the second-most consumed beverage was unflavored bottled water, which might help explain the Great Pacific Garbage Patch.	Plastic within the Great Pacific Garbage Patch is 'increasing exponentially,' scientists find. Seventy-nine thousand tons of plastic debris, in the form of 1.8 trillion pieces, now occupy an area three times the size of France in the Pacific Ocean between California and Hawaii, a scientific team reported on Thursday.	371	N/A	N/A	N/A	N/A	N/A	N/A
Turkey's elections show the limits of Erdogan's nationalism. Ahead of local elections throughout his country last weekend, Turkish President Recep Tayyip Erdogan resorted to his usual tactics. He cast some of his ruling party's opponents as traitors in league with terrorists.	There's a broader story to be told, as well.	Well before President Trump, Indian Prime Minister Narendra Modi or Hungarian Prime Minister Viktor Orban, Erdogan arrived at the politics of the zeitgeist.	Trump's populism is about creating division, not unity. President Trump begins his third week in office with the worst approval ratings of any new American president since polls began tracking such results.	786	Stunning setbacks in Turkey's elections dent Erdogan's aura of invincibility. ISTANBUL — Turkish President Recep Tayyip Erdogan faced the prospect Monday of a stinging electoral defeat in Istanbul, the city whose politics he dominated for a quarter of a century, with vote results showing what appeared to be an opposition victory in the race for the city's mayor.	1	447	471	487	535

main difference of the aforementioned works and our task is that we aim to retrieve articles to expand existing incomplete narratives instead of finding citations for complete narratives.

6.8.3 Event extraction & retrieval

Events are the starting points of narrative news items. Recent work has focused on extracting and characterizing events from large streams of documents [32] and extracting the most dominant events from news articles [36]. In our work, we assume that a news article is associated with a single main event, which is described by the article's headline and lead paragraph [37].

More related to our task is work focused on retrieving events given a query event [102, 163]. However, this work does not consider additional context in the query as we do and thus it is not directly comparable to ours.

6.9 Conclusion and Future Work

In this chapter, we proposed and studied the task of news article retrieval in context for event-centric narrative creation. We proposed an automatic dataset construction procedure and showed that it can generate reliable evaluation sets for this task. Using the generated datasets, we compared lexical and semantic rankers and found that they are insufficient. We found that combining those rankers with one that ranks articles by their reverse chronological order significantly improves retrieval performance over those rankers alone.

Our analysis showed that the vocabulary gap for this task is large, and therefore more advanced methods for semantic matching are needed. This could be achieved by exploiting external knowledge about events stored in knowledge graphs [67]. To this end we aim to build on insights gained from our studies in Chapters 2, 3 and 4 to improve semantic matching. For instance, we could first detect KG facts in the query and the articles and then use the method we proposed in Chapter 2 to retrieve descriptions of the detected KG facts. The retrieved descriptions can then be used to provide additional knowledge to the BERT ranker and thus improve semantic matching [137].

Moreover, our analysis showed that the temporal aspect is prominent in this retrieval task, which was not the case for the tasks we studied in the previous chapters of this thesis. Therefore, future work would aim to find more robust ways to incorporate the temporal aspect in the ranking function [90].

Furthermore, we found that this task is more challenging when the query event involves entities that appear more frequently in the collection, which we plan to further study in the future. Another direction for future work is to categorize queries in relation to their discourse function in the narrative [174, 175], for example in relation to their function with respect to the main event of the narrative [37], and develop specialized rankers for each category.

We found that in some cases the link sentence contains crucial information for the connection between the *complete* narrative and the relevant news article. However, since the link sentence is not part of the query according to our dataset construction procedure, the constructed query may miss this key piece of information to capture the connection. For future work, we aim to address this limitation by detecting such information in the link sentence and adding it to the query, or by using natural language generation techniques to fill in these blanks automatically.

Finally, it is important to note that even though our dataset construction procedure can generate reliable retrieval datasets, the fact that we only have a single relevant article for each query may be limiting as more than one article may be relevant. Thus, some of our findings might be an artifact of that procedure and not the task itself. We plan to overcome this limitation in future work by asking journalists to qualitatively assess the output of different rankers to enrich the automatically constructed datasets with more relevant articles per query [113, 114].

7 Conclusions

In this thesis, we studied three research themes aimed at supporting search engines with knowledge and context: (1) making structured knowledge more accessible to the user, (2) improving interactive knowledge gathering, and (3) supporting knowledge exploration for narrative creation. We studied several algorithmic tasks within these themes and proposed solutions to address them.

In this concluding chapter, we first revisit the research questions that we introduced in Chapter 1 and describe our main findings in Section 7.1. In Section 7.2, we discuss limitations and future directions.

7.1 Main Findings

7.1.1 Making structured knowledge more accessible to the user

Within this research theme we asked and answered three research questions motivated by the need of presenting knowledge graph (KG) facts to users in a natural way. In Chapter 2, we asked the following question:

RQ1 Given a KG fact and a text corpus, can we retrieve textual descriptions of the fact from the text corpus?

To answer this question, we formalized the task of retrieving textual descriptions of KG facts from a corpus of sentences. We developed a method for this task that consists of two steps. First, we extract and enrich candidate sentences from the corpus and then rank them by how well they describe the KG fact. In the first step, we detect sentences in the corpus that contain surface forms of any of the two entities in the KG fact and apply coreference resolution and entity linking to enrich them. In the second step, we rank the extracted sentences using learning to rank, that combines a rich set of features of different types. To evaluate our method, we construct a manually annotated dataset that contains descriptions of KG facts that involve people. We found that our method improves performance over state-of-the-art sentence retrieval methods and that all groups of features contribute to retrieval performance, with relation-based features being the most important. Moreover, we found that training relationship-dependent rankers is beneficial to improving retrieval performance. Importantly, we also found that almost one third of the facts in our dataset did not correspond to any relevant sentence in the corpus. This is usually the case for facts of which the entities are less popular. Not being able to provide a meaningful description of certain facts limits the applicability of our method in real world scenarios. This finding led us to the following research question in Chapter 3:

RQ2 Given a KG fact, can we automatically generate a textual description of the fact in the absence of an existing description?

To answer this question, we formalized the task of generating textual descriptions of KG facts. We proposed a method that first generates sentence templates for a specific relationship and then, given a specific KG fact selects the most relevant template and fills it with information from the KG to create a novel sentence. In order to create sentence templates, we designed a graph-based algorithm that combines information contained in existing sentences and the KG. In order to select the most relevant template for a KG fact, we designed a supervised feature-based scoring function. To evaluate our method, we automatically extracted a dataset for KG fact description generation and performed both automatic and manual evaluation. We found that our method can generate grammatically correct and generally informative descriptions, and that a supervised scoring function outperforms an unsupervised one for selecting templates. In addition, our error analysis showed that generating KG fact descriptions that are valid under the KG closed-world assumption is challenging and needs to receive more attention.

Next, in Chapter 4 we turned to a closely related problem and asked the following question:

RQ3 Can we contextualize a KG query fact by retrieving other, related KG facts?

To answer this question, we formalized the problem of contextualizing KG facts as a retrieval task. We designed NFCM, a neural fact contextualization method that first generates a set of candidate facts that are part of the immediate neighborhood of the query fact in the KG, and subsequently ranks the candidate facts by how relevant they are to the query fact. We designed a neural network ranking model that combines information from multiple paths connecting the query and the candidate facts in the KG using recurrent neural networks to learn automatic features. We further augmented the representation power of this model by using existing and novel hand-crafted features. Since it is expensive to manually obtain human-curated training data to train this model, we turned to distant supervision to automatically generate training data for this task. We evaluated NFCM using a human-curated dataset separate from the one used for distant supervision. We found that when trained on distant supervision, NFCM significantly outperforms several heuristic baselines on this task. Additionally, we found that NFCM benefits from both automatically learned and hand-crafted features. Finally, we found that NFCM is relatively robust to the number of training data for each relationship.

7.1.2 Improving interactive knowledge gathering

We then moved to the theme of improving interactive knowledge gathering and studied multi-turn passage retrieval as an instance of conversational search. In Chapter 5, we asked the following question:

RQ4 Can we use query resolution to identify relevant context and thereby improve retrieval in conversational search?

To answer this question, we formulated the task of query resolution for conversational search as a term classification task. We proposed QuReTeC, a neural term classification model based on bidirectional transformers, more specifically BERT. QuReTeC encodes the conversation history and the current turn query and predicts which terms from the history are relevant to the current turn. We integrated QuReTeC in a standard, two-step retrieval pipeline by appending the terms predicted as relevant to the current turn query. We performed evaluation both in terms of term classification and retrieval performance using a recently constructed multi-turn passage retrieval dataset. We found that QuReTeC significantly outperforms state-of-the-art methods on this task when trained on gold standard query resolutions. Furthermore, we found that QuReTeC is robust across conversation turns. Since collecting such gold standard query resolutions for training QuReTeC might be cumbersome, we designed a distant supervision method that automatically generates training data for query resolution using query-passage relevance labels. We found that this distant supervision method can substantially reduce the number of gold standard query resolutions required for training QuReTeC, a result especially important in low resource scenarios.

7.1.3 Supporting knowledge exploration for narrative creation

Our next study was in the theme of supporting knowledge exploration for narrative creation. In Chapter 6, we asked the following question:

RQ5 Can we support knowledge exploration for event-centric narrative creation by performing news article retrieval in context?

To answer this question, we formalized the task of event-centric news article retrieval in context. We proposed an automatic retrieval dataset construction procedure that can produce reliable datasets for this task. We generated two retrieval datasets using this procedure and used the generated datasets to evaluate automatic methods for this task. We found that an unsupervised combination of state-of-the-art lexical and semantic rankers and a ranker that ranks articles by reverse chronological order outperforms those rankers alone. We performed an in-depth quantitative and qualitative analysis to acquire insights into the characteristics of this task. We found that this task has a large vocabulary gap, which highlights the need for semantic matching that takes into account structured knowledge about events. In addition, we found that the temporal aspect is prominent in this task and thus more advanced temporal query and collection characteristics need to be explored. Moreover, we found that this task is more challenging for queries that contain entities that appear more frequently in the underlying news article collection. Last, we found that our dataset construction procedure is sometimes prone to generating queries that are not sufficiently defined, which is a clear future work direction.

We now reflect on the main question we asked in Chapter 1, namely how to support search engines in leveraging knowledge while accounting for different types of context. In the first part of this thesis (Chapters 2, 3 and 4), we proposed tasks and methods that make structured knowledge more accessible to the user (when the search engine proactively provides context to enrich search results) by retrieving existing or generating novel descriptions of KG facts, and also by contextualizing KG facts with other, related facts. In the second part of this thesis (Chapter 5), we proposed a method for query resolution that improves interactive knowledge gathering in conversational search by adding missing context from the conversation history to the current turn query. In the third part of this thesis (Chapter 6), we proposed and studied the task of retrieving news articles that are relevant to the user's broad query (the query event) and a context that further specifies the query, thereby supporting knowledge exploration for narrative creation.

7.2 Future Directions

In this section, we discuss limitations of our study and directions for future work that would overcome those limitations and further expand our work.

7.2.1 Making structured knowledge more accessible to the user

Validity of KG fact descriptions Ensuring the validity of automatically generated KG fact descriptions is crucial when presenting such descriptions to the user [65]. In Chapter 3 we found that generating valid KG fact descriptions is a challenging task. This is a challenge not only for template-based generation methods such as ours but also for neural sequence to sequence generation methods [12, 116, 200]. A possible direction towards overcoming this challenge is to learn discrete templates jointly with learning how to generate [195]. Another possible direction is to learn to edit existing descriptions instead of generating descriptions from scratch [72]. Moreover, it would be interesting to assess the ability of recently developed large-scale pretrained language models for generating valid KG fact descriptions [144].

Richness of KG fact descriptions In Chapter 4, we proposed NFCM, a neural fact contextualization method. Relevant facts retrieved by NFCM can be used to improve KG fact description retrieval by better modeling the relevance of existing descriptions (Chapter 2). In addition, they can be used to select more informative templates in KG fact description generation (Chapter 3).

Source of KG fact descriptions In Chapters 2, 3 and 4 we used Wikipedia as the source of existing descriptions of KG facts. Using other sources of such descriptions could widen the applicability of our proposed tasks and methods to less popular entities. Huang et al. [81] performed an initial exploration towards this direction by using web pages as the source of descriptions, with an application to KG fact description retrieval. Their results showed that using the web as the source of descriptions poses further challenges that would be interesting to explore even further.

Query-dependent KG fact information Deciding what information about a KG fact to present in a SERP may depend on the user's query [75]. Future work could develop query-dependent methods for all three tasks we considered: KG fact description retrieval and generation, and KG fact contextualization. A study with real search engine users interacting with KG fact information on SERPs could provide further insights into this direction.

7.2.2 Improving interactive knowledge gathering

Incorporating the system's response In Chapter 5, we followed the TREC CAsT 2019 setup and only took into account the previous turn queries (the ones that preceded the current turn query) but not the passages retrieved by the system for those queries (the system's response). In future work, we will evaluate QuReTeC on a more realistic scenario where the passages retrieved for the previous turn queries are also taken into account.

Distant supervision for query resolution In Chapter 5, we proposed a distant supervision method for reducing the amount of query resolution training data required to train QuReTeC. Our distant supervision method relies on query-passage relevance labels. Future work could address how to combine our distant supervision method with methods that generate relevance labels with weak supervision [49], pseudo-relevance feedback [98] or user signals [88]. Also, we would like to explore noise reduction methods to improve the quality of the distant supervision signal [156].

Term classification and rewriting for query resolution In Chapter 5, we formulated query resolution for conversational search as a term classification task. This gave us flexibility not only in terms of modeling but also in terms of where we can get the supervision signal from. In two studies contemporaneous to ours, query resolution was formulated as a sequence generation task [179, 209]. Combining the strengths of both formulations of the query resolution task could result in developing more powerful models.

Specialized rankers in low resource settings In Chapter 5, we focused on query resolution for conversational search and used existing rankers for both the initial retrieval and reranking steps. State-of-the-art neural ranking models rely on large-scale annotated ranking datasets that are not yet available in conversational search [46, 203]. Therefore, future work could develop specialized rankers for conversational search in low resource settings, possibly by learning to perform query resolution and ranking in a joint manner.

7.2.3 Supporting knowledge exploration for narrative creation

Incorporate structured knowledge about events In Chapter 6, we found that the vocabulary gap for the retrieval task we studied is large, and that the retrieval methods we considered are not able to effectively account for that. One possible direction for future work is to incorporate structured knowledge about events (and the entities involved in them) in the retrieval methods [67, 154]. Such knowledge includes relationships between entities (which we studied in Chapters 2, 3 and 4), or sub-event relations [10, 68].

Temporal aspect In Chapter 6, we found that a simple combination of lexical and semantic rankers with a ranker that ranks articles by reverse chronological order improves performance when the relevant article is recent but harms performance otherwise, as expected. In future work, we aim to incorporate the temporal aspect in the ranking function in a more robust way. A possible way to achieve that is to identify temporal phenomena such as trending terms or entities in the underlying news article collection [90] or in external sources such as social media [44].

Dataset construction In Chapter 6, we proposed a dataset construction procedure that can produce reliable datasets for this task. The main limitation of this procedure is that only one article is relevant for each query, even though more than one article may be relevant. In future work, we aim to ask experts (journalists) to qualitatively assess the output of different rankers to enrich the automatically constructed datasets with more relevant articles per query [113, 114]. Another limitation of our dataset construction procedure is that in some cases the query (usually the narrative context) misses crucial information for the connection of the complete narrative and the relevant article that is contained in the link sentence. In future work we aim to ask experts to manually add such missing information to the narrative context. Since this task can be cumbersome, we will try to semi-automate this procedure by casting this as a prediction task [87].

Specialized rankers per discourse function Previous work has studied how to categorize parts of existing narratives according to their discourse function with respect to the main event of the narrative [37, 175]. We hypothesize that, in the narrative creation task we studied in Chapter 6, queries that serve a certain discourse function would have relevant news articles of specific characteristics. In future work, we aim to categorize queries to different discourse functions and perform manual analysis to validate this hypothesis. We will then design specialized rankers for each category to improve retrieval effectiveness.

Bibliography

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*. USENIX, 2016. (Cited on page 55.)

[2] N. Abdul-jaleel, J. Allan, W. B. Croft, O. Diaz, L. Larkey, X. Li, M. D. Smucker, and C. Wade. Umass at trec 2004: Novelty and hard. In *TREC*. NIST, 2004. (Cited on pages 68 and 76.)

[3] A. Agarwal, H. Raghavan, K. Subbian, P. Melville, R. D. Lawrence, D. C. Gondek, and J. Fan. Learning to rank for robust question answering. In *CIKM*. ACM, 2012. (Cited on page 13.)

[4] B. Aleman-Meza, C. Halaschek-Weiner, I. B. Arpinar, C. Ramakrishnan, and A. P. Sheth. Ranking complex relationships on the semantic web. *IEEE Internet Computing*, 9, 2005. (Cited on page 59.)

[5] E. Alfonseca, K. Filippova, J.-Y. Delort, and G. Garrido. Pattern learning for relation extraction with a hierarchical topic model. In *ACL*. ACL, 2012. (Cited on page 59.)

[6] M. Aliannejadi, H. Zamani, F. Crestani, and W. B. Croft. Asking Clarifying Questions in Open-Domain Information-Seeking Conversations. In *SIGIR*. ACM, 2019. (Cited on page 1.)

[7] J. Allan, J. Aslam, N. Belkin, C. Buckley, J. Callan, B. Croft, S. Dumais, N. Fuhr, D. Harman, D. J. Harper, D. Hiemstra, T. Hofmann, E. Hovy, W. Kraaij, J. Lafferty, V. Lavrenko, D. Lewis, L. Liddy, R. Manmatha, A. McCallum, J. Ponte, J. Prager, D. Radev, P. Resnik, S. Robertson, R. Rosenfeld, S. Roukos, M. Sanderson, R. Schwartz, A. Singhal, A. Smeaton, H. Turtle, E. Voorhees, R. Weischedel, J. Xu, and C. Zhai. Challenges in information retrieval and language modeling: report of a workshop held at the center for intelligent information retrieval, University of Massachusetts Amherst, September 2002. *ACM SIGIR Forum*, 37(1), 2003. (Cited on page 1.)

[8] J. Allan, C. Wade, and A. Bolivar. Retrieval and novelty detection at the sentence level. In *SIGIR*. ACM, 2003. (Cited on page 13.)

[9] T. Althoff, X. L. Dong, K. Murphy, S. Alai, V. Dang, and W. Zhang. Timemachine: Timeline generation for knowledge-base entities. In *KDD*. ACM, 2015. (Cited on page 28.)

[10] J. Araki, L. Mulaffer, A. Pandian, Y. Yamakawa, K. Oflazer, and T. Mitamura. Interoperable Annotation of Events and Event Relations across Domains. In *isa-14: The 14th Joint ACL - ISO Workshop on Interoperable Semantic Annotation*. ACL, 2018. (Cited on page 113.)

[11] R. Baeza-Yates, B. Ribeiro-Neto, et al. *Modern information retrieval*. ACM, 1999. (Cited on page 13.)

[12] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015. (Cited on pages 48 and 112.)

[13] P. Bailey, N. Craswell, R. W. White, L. Chen, A. Satyanarayana, and S. Tahaghoghi. Evaluating whole-page relevance. In *SIGIR*. ACM, 2010. (Cited on page 1.)

[14] P. Bajaj, D. Campos, N. Craswell, L. Deng, J. Gao, X. Liu, R. Majumder, A. McNamara, B. Mitra, T. Nguyen, M. Rosenberg, X. Song, A. Stoica, S. Tiwary, and T. Wang. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. *arXiv preprint arXiv:1611.09268*, 2018. (Cited on page 78.)

[15] H. Bast, B. Björn, and E. Haussmann. Semantic search on text and knowledge bases. *FnTIR*, 10(2-3), 2016. (Cited on page 41.)

[16] N. J. Belkin. Anomalous states of knowledge as a basis for information retrieval. *CJIS*, 5(1), 1980. (Cited on pages 2 and 67.)

[17] N. J. Belkin, C. Cool, A. Stein, and U. Thiel. Cases, scripts, and information-seeking strategies: On the design of interactive information retrieval systems. *Expert Systems with Applications*, 9(3), 1995. (Cited on page 67.)

[18] P. N. Bennett, R. W. White, W. Chu, S. T. Dumais, P. Bailey, F. Borisyuk, and X. Cui. Modeling the impact of short- and long-term behavior on search personalization. In *SIGIR*. ACM, 2012. (Cited on page 1.)

[19] C. Bhagavatula, S. Feldman, R. Power, and W. Ammar. Content-Based Citation Recommendation. In *NAACL-HLT*. ACL, 2018. (Cited on page 2.)

[20] R. Blanco and H. Zaragoza. Finding support sentences for entities. In *SIGIR*. ACM, 2010. (Cited on pages 13, 15, and 16.)

[21] R. Blanco, B. B. Cambazoglu, P. Mika, and N. Torzec. Entity recommendations in web search. In *ISWC*. Springer, 2013. (Cited on pages 1, 11, 28, and 41.)

[22] R. Blanco, G. Ottaviano, and E. Meij. Fast and space-efficient entity linking for queries. In *WSDM*. ACM, 2015. (Cited on pages 27 and 41.)

[23] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*. ACM, 2008. (Cited on pages 30, 36, 43,

and 51.)

[24] A. Borisov, P. Serdyukov, and M. de Rijke. Using metafeatures to increase the effectiveness of latent semantic models in web search. In *WWW*. ACM, 2016. (Cited on page 49.)

[25] H. Bota, K. Zhou, and J. M. Jose. Playing your cards right: The effect of entity cards on search behaviour and workload. In *CHIIR*. ACM, 2016. (Cited on pages 1 and 41.)

[26] L. Breiman. Random forests. *Mach. Learn.*, 45(1), 2001. (Cited on page 19.)

[27] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. In *WWW*. Elsevier, 1998. (Cited on page 2.)

[28] C. J. Burges, K. M. Svore, P. N. Bennett, A. Pastusiak, and Q. Wu. Learning to rank using an ensemble of lambda-gradient models. In *Yahoo! Learning to Rank Challenge*, 2011. (Cited on page 13.)

[29] B. Carterette, E. Kanoulas, M. Hall, and P. Clough. Overview of the trec 2014 session track. In *TREC*. NIST, 2014. (Cited on pages 66 and 67.)

[30] B. Carterette, P. Clough, M. Hall, E. Kanoulas, and M. Sanderson. Evaluating retrieval over sessions: The trec session track 2011-2014. In *SIGIR*. ACM, 2016. (Cited on page 66.)

[31] D. Caswell and K. Dörr. Automated Journalism 2.0: Event-driven narratives. *Journalism Practice*, 12 (4), 2018. (Cited on pages 90 and 91.)

[32] A. Chaney, H. Wallach, M. Connelly, and D. Blei. Detecting and Characterizing Events. In *EMNLP*. ACL, 2016. (Cited on pages 91 and 106.)

[33] O. Chapelle, D. Metzler, Y. Zhang, and P. Grinspan. Expected reciprocal rank for graded relevance. In *CIKM*. ACM, 2009. (Cited on page 20.)

[34] M. Cho, K. Alahari, and J. Ponce. Learning graphs to match. In *ICCV*. IEEE, 2013. (Cited on page 59.)

[35] E. Choi, H. He, M. Iyyer, M. Yatskar, W.-t. Yih, Y. Choi, P. Liang, and L. Zettlemoyer. QuAC: Question Answering in Context. In *EMNLP*. ACL, 2018. (Cited on pages 65, 67, and 75.)

[36] P. K. Choubey, K. Raju, and R. Huang. Identifying the Most Dominant Event in a News Article by Mining Event Coreference Relations. In *NAACL-HLT*. ACL, 2018. (Cited on page 106.)

[37] P. K. Choubey, A. Lee, R. Huang, and L. Wang. Discourse as a Function of Event: Profiling Discourse Structure in News Articles around the Main Event. In *ACL*. ACL, 2020. (Cited on pages 89, 91, 106, and 114.)

[38] A. Chuklin, I. Markov, and M. de Rijke. *Click Models for Web Search*. Morgan & Claypool Publishers, August 2015. (Cited on page 1.)

[39] E. Clark, A. S. Ross, C. Tan, Y. Ji, and N. A. Smith. Creative Writing with a Machine in the Loop: Case Studies on Slogans and Stories. In *IUI*. ACM, 2018. (Cited on pages 2 and 4.)

[40] J. Cohen. Multiple regression as a general data-analytic system. *Psychological Bulletin*, 70(6, Pt.1), 1968. (Cited on page 94.)

[41] G. V. Cormack, C. L. A. Clarke, and S. Buettcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *SIGIR*. ACM, 2009. (Cited on pages 70, 78, and 95.)

[42] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to construct knowledge bases from the world wide web. *Artificial Intelligence*, 118(1–2), 2000. (Cited on page 11.)

[43] W. B. Croft and R. H. Thompson. I3r: A new approach to the design of document retrieval systems. *JASIST*, 38(6), 1987. (Cited on pages 2 and 67.)

[44] A. Cucchiarelli, C. Morbidoni, G. Stilo, and P. Velardi. A topic recommender for journalists. *IRJ*, 22 (1), 2019. (Cited on pages 2, 4, 89, 104, and 114.)

[45] J. S. Culpepper, F. Diaz, and M. D. Smucker. Research Frontiers in Information Retrieval: Report from the Third Strategic Workshop on Information Retrieval in Lorne (SWIRL 2018). *SIGIR Forum*, 52(1), 2018. (Cited on pages 1, 4, and 65.)

[46] J. Dalton, C. Xiong, and J. Callan. Cast 2019: The conversational assistance track overview. In *TREC 2019*. NIST, 2019. (Cited on pages 1, 4, 65, 67, 73, 75, 77, and 113.)

[47] N. Dalvi, R. Kumar, B. Pang, R. Ramakrishnan, A. Tomkins, P. Bohannon, S. Keerthi, and S. Merugu. A web of concepts. In *ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, 2009. (Cited on page 1.)

[48] R. Das, A. Neelakantan, D. Belanger, and A. McCallum. Chains of reasoning over entities, relations, and text using recurrent neural networks. In *EACL*. ACL, 2017. (Cited on pages 47 and 48.)

[49] M. Dehghani, H. Zamani, A. Severyn, J. Kamps, and W. B. Croft. Neural ranking models with weak supervision. In *SIGIR*. ACM, 2017. (Cited on pages 46, 73, 81, 85, and 113.)

[50] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*. ACL, 2019. (Cited on pages 66, 70, 71, 72, and 95.)

[51] N. Diakopoulos. *Automating the News: How Algorithms Are Rewriting the Media*. Harvard University Press, 2019. (Cited on pages 2, 4, and 89.)

[52] N. Diakopoulos, M. De Choudhury, and M. Naaman. Finding and assessing social media information sources in the context of journalism. In *CHI*. ACM, 2012. (Cited on pages 4, 89, and 104.)

[53] L. Dietz, M. Verma, F. Radlinski, and N. Craswell. Trec complex answer retrieval overview. In *TREC*. NIST, 2017. (Cited on page 75.)

[54] A. Doko, M. Štula, and D. Stipaničev. A recursive TF-ISF based sentence retrieval method with local context. *IJMLC*, 3(2), 2013. (Cited on pages 13, 17, and 18.)

[55] A. Dong, Y. Chang, Z. Zheng, G. Mishne, J. Bai, R. Zhang, K. Buchner, C. Liao, and F. Diaz. Towards recency ranking in web search. In *WSDM*. ACM, 2010. (Cited on page 103.)

[56] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *KDD*. ACM, 2014. (Cited on page 11.)

[57] T. Ebesu and Y. Fang. Neural Citation Network for Context-Aware Citation Recommendation. In *SIGIR*. ACM, 2017. (Cited on page 104.)

[58] A. Elgohary, D. Peskov, and J. Boyd-Graber. Can You Unpack That? Learning to Rewrite Questions-in-Context. In *EMNLP*. ACL, 2019. (Cited on pages 66, 67, 71, 73, 75, 77, and 78.)

[59] M. Fadaee, A. Bisazza, and C. Monz. Data Augmentation for Low-Resource Neural Machine Translation. In *ACL*. ACL, 2017. (Cited on page 71.)

[60] J. Fan, R. Hoffman, A. Kalyanpur, S. Riedel, F. Suchanek, and P. P. Talukdar. *AKBC-WEKEX: the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*. ACL, 2012. (Cited on page 11.)

[61] L. Fang, A. D. Sarma, C. Yu, and P. Bohannon. REX: explaining relationships between entity pairs. *VLDB Endowment*, 5(3), 2011. (Cited on pages 13, 27, and 58.)

[62] R. T. Fernández, D. E. Losada, and L. Azzopardi. Extending the language modeling framework for sentence retrieval to include local context. *Information Retrieval*, 14(4), 2011. (Cited on pages 13 and 20.)

[63] K. Ganesan, C. Zhai, and J. Han. Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions. In *COLING*. ACL, 2010. (Cited on page 33.)

[64] J. Gao, M. Galley, and L. Li. Neural Approaches to Conversational AI. In *ACL*. ACL, 2018. (Cited on pages 1, 4, and 65.)

[65] A. Gatt and E. Krahmer. Survey of the State of the Art in Natural Language Generation: Core tasks, applications and evaluation. *JAIR*, 61, 2018. (Cited on page 112.)

[66] D. Gkatzia, O. Lemon, and V. Rieser. Natural language generation enhances human decision-making with uncertain information. In *ACL*, 2016. (Cited on pages 3 and 27.)

[67] S. Gottschalk and E. Demidova. EventKG: A Multilingual Event-Centric Temporal Knowledge Graph. In A. Gangemi, R. Navigli, M.-E. Vidal, P. Hitzler, R. Troncy, L. Hollink, A. Tordai, and M. Alam, editors, *ESWC 2018*. Springer, 2018. (Cited on pages 99, 106, and 113.)

[68] S. Gottschalk and E. Demidova. HapPenIng: Happen, Predict, Infer—Event Series Completion in a Knowledge Graph. In *ISWC*. Springer, 2019. (Cited on page 113.)

[69] D. Guan, H. Yang, and N. Goharian. Effective structured query formulation for session search. In *TREC*. NIST, 2012. (Cited on pages 66, 67, and 77.)

[70] J. Guo, G. Xu, X. Cheng, and H. Li. Named entity recognition in query. In *SIGIR*. ACM, 2009. (Cited on page 1.)

[71] K. Guu, J. Miller, and P. Liang. Traversing knowledge graphs in vector space. In *EMNLP*. ACL, 2015. (Cited on pages 47 and 48.)

[72] K. Guu, T. B. Hashimoto, Y. Oren, and P. Liang. Generating Sentences by Editing Prototypes. *TACL*, 6, 2018. (Cited on page 112.)

[73] D. Hacker and N. Sommers. *A Writer's Reference with Writing in the Disciplines*. Macmillan, 2011. (Cited on page 92.)

[74] M. Hagen, M. Potthast, M. Völske, J. Gomoll, and B. Stein. How Writers Search: Analyzing the Search and Writing Logs of Non-fictional Essays. In *CHIIR*. ACM Press, 2016. (Cited on page 2.)

[75] F. Hasibi, K. Balog, and S. E. Bratsberg. Dynamic factual summaries for entity cards. In *SIGIR*. ACM, 2017. (Cited on pages 1, 41, 59, and 112.)

[76] Q. He, J. Pei, D. Kifer, P. Mitra, and L. Giles. Context-aware citation recommendation. In *WWW*. ACM, 2010. (Cited on pages 2 and 104.)

[77] L. Hirschman and R. Gaizauskas. Natural language question answering: the view from here. *NLE*, 7 (04), 2001. (Cited on page 13.)

[78] N. Höchstötter and D. Lewandowski. What Users See - Structures in Search Engine Results Pages. *Inf. Sci.*, 2009. (Cited on page 1.)

[79] K. Hofmann, S. Whiteson, and M. de Rijke. Balancing exploration and exploitation in listwise and pairwise online learning to rank for information retrieval. *Information Retrieval*, 16(1), 2013. ISSN 1573-7659. (Cited on page 1.)

[80] K. Hofmann, L. Li, and F. Radlinski. Online Evaluation for Information Retrieval. *FnTIR*, 10(1), 2016. (Cited on page 1.)

[81] J. Huang, W. Zhang, S. Zhao, S. Ding, and H. Wang. Learning to explain entity relationships by pairwise ranking with convolutional neural networks. In *IJCAI*. AAAI, 2017. (Cited on pages 58 and 112.)

[82] W. Huang, Z. Wu, C. Liang, P. Mitra, and C. L. Giles. A neural probabilistic model for context based citation recommendation. In *AAAI*. AAAI, 2015. (Cited on page 104.)

[83] B. Huurnink, L. Hollink, W. v. d. Heuvel, and M. d. Rijke. Search behavior of media professionals at an audiovisual archive: A transaction log analysis. *JASIST*, 61(6), 2010. ISSN 1532-2890. (Cited on pages 4 and 89.)

[84] R. Jagerman, H. Oosterhuis, and M. de Rijke. To Model or to Intervene: A Comparison of Counterfactual and Online Learning to Rank from User Interactions. In *SIGIR*. ACM, 2019. (Cited on page 1.)

[85] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *TOIS*, 20(4), 2002. (Cited on page 20.)

[86] C. Jeong, S. Jang, H. Shin, E. Park, and S. Choi. A Context-Aware Citation Recommendation Model with BERT and Graph Convolutional Networks. *arXiv:1903.06464 [cs]*, 2019. arXiv: 1903.06464. (Cited on page 104.)

[87] R. Jha, A.-A. Jbara, V. Qazvinian, and D. R. Radev. NLP-driven citation analysis for scientometrics. *NLE*, 23(1), 2017. (Cited on pages 104 and 114.)

[88] T. Joachims. Optimizing search engines using clickthrough data. In *KDD*. ACM, 2002. (Cited on pages 1, 66, 73, and 113.)

[89] M. Joshi, O. Levy, D. S. Weld, and L. Zettlemoyer. BERT for Coreference Resolution: Baselines and Analysis. In *EMNLP-IJCNLP*. ACL, 2019. (Cited on page 71.)

[90] N. Kanhabua, R. Blanco, and K. Nørvåg. Temporal Information Retrieval. *FnTIR*, 9(2), 2015. (Cited on pages 103, 106, and 114.)

[91] Y. Kim, J. Seo, and W. B. Croft. Automatic boolean query suggestion for professional search. In *SIGIR*. ACM, 2011. (Cited on pages 2 and 96.)

[92] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2014. (Cited on page 46.)

[93] K. Kirkpatrick. Putting the Data Science into Journalism. *Commun. ACM*, 58(5), 2015. (Cited on pages 2, 4, and 89.)

[94] I. Konstas and M. Lapata. A global model for concept-to-text generation. *JAIR*, 48, 2013. (Cited on page 36.)

[95] A. M. Krasakis, M. Aliannejadi, N. Voskarides, and E. Kanoulas. Analysing the effect of clarifying questions on document ranking in conversational search. In *ICTIR*. ACM, 2020. (Cited on page 8.)

[96] V. Kumar and S. Joshi. Incomplete Follow-up Question Resolution Using Retrieval Based Sequence to Sequence Learning. In *SIGIR*. ACM, 2017. (Cited on pages 66, 67, and 71.)

[97] A. Lavie and A. Agarwal. METEOR: An automatic metric for MT evaluation with high levels of correlation with human judgments. In *WMT*. ACL, 2007. (Cited on page 36.)

[98] V. Lavrenko and W. B. Croft. Relevance based language models. In *SIGIR*. ACM, 2001. (Cited on pages 68 and 113.)

[99] R. Lebret, D. Grangier, and M. Auli. Neural text generation from structured data with application to the biography domain. In *EMNLP*. ACL, 2016. (Cited on pages 29 and 41.)

[100] G. G. Lee, J. Seo, S. Lee, H. Jung, B.-H. Cho, C. Lee, B.-K. Kwak, J. Cha, D. Kim, J. An, et al. SiteQ: Engineering high performance QA system using lexico-semantic pattern matching and shallow NLP. In *TREC*. NIST, 2001. (Cited on page 16.)

[101] H. Lee, Y. Peirsman, A. Chang, N. Chambers, M. Surdeanu, and D. Jurafsky. Stanford's multi-pass sieve coreference resolution system at the CoNLL-2011 shared task. In *CoNLL*. ACL, 2011. (Cited on page 14.)

[102] C. Li, M. Bendersky, V. Garg, and S. Ravi. Related Event Discovery. In *WSDM*. ACM, 2017. (Cited on pages 104 and 106.)

[103] H. Li and J. Xu. *Semantic Matching in Search*. Now Publishers, 2014. (Cited on page 98.)

[104] J. Li, M. Galley, C. Brockett, J. Gao, and B. Dolan. A Diversity-Promoting Objective Function for Neural Conversation Models. In *NAACL-HLT*. ACL, 2016. (Cited on page 65.)

[105] C.-Y. Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*. ACL, 2004. (Cited on page 36.)

[106] T. Lin, P. Pantel, M. Gamon, A. Kannan, and A. Fuxman. Active objects: Actions for entity-centric search. In *WWW*. ACM, 2012. (Cited on pages 1, 27, and 28.)

[107] Y. Lin, Z. Liu, and M. Sun. Knowledge representation learning with entities, attributes and relations. In *IJCAI*. AAAI, 2016. (Cited on page 35.)

[108] Y. Lin, Y. C. Tan, and R. Frank. Open sesame: Getting inside bert's linguistic knowledge. *arXiv preprint arXiv:1906.01698*, 2019. (Cited on page 71.)

[109] X. Ling and D. S. Weld. Fine-grained entity recognition. In *AAAI*. AAAI Press, 2012. (Cited on page 59.)

[110] D. E. Losada. A study of statistical query expansion strategies for sentence retrieval. In *SIGIR*. ACM, 2008. (Cited on page 13.)

[111] S. MacAvaney. OpenNIR: A Complete Neural Ad-Hoc Ranking Pipeline. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. ACM, 2020. (Cited on pages 95 and 96.)

[112] S. MacAvaney, A. Yates, A. Cohan, and N. Goharian. CEDR: Contextualized Embeddings for Document Ranking. In *SIGIR*. ACM, 2019. (Cited on pages 70 and 78.)

[113] A. MacLaughlin, T. Chen, B. K. Ayan, and D. Roth. Context-Based Quotation Recommendation. In *ICWSM*. ACM, 2021. (Cited on pages 4, 89, 95, 104, 107, and 114.)

[114] N. Maiden and K. Zachos. INJECT: Algorithms to Discover Creative Angles on News. 2020. Meeting Name: Computation + Journalism 2020. (Cited on pages 89, 104, 107, and 114.)

[115] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*, volume 1. Cambridge university press, 2008. (Cited on pages 1 and 103.)

[116] J. Maynez, S. Narayan, B. Bohnet, and R. McDonald. On Faithfulness and Factuality in Abstractive Summarization. In *ACL*. ACL, 2020. (Cited on page 112.)

[117] O. A. McBryan. GENVL and WWWW: Tools for Taming the Web. In *WWW*, 1994. (Cited on page 2.)

[118] E. Meij, W. Weerkamp, and M. de Rijke. Adding semantics to microblog posts. In *WSDM*. ACM, 2012. (Cited on pages 15 and 31.)

[119] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, 2013. (Cited on page 17.)

[120] I. Miliaraki, R. Blanco, and M. Lalmas. From "selena gomez" to "marlon brando": Understanding explorative entity search. In *WWW*. ACM, 2015. (Cited on pages 1 and 41.)

[121] D. Milne and I. H. Witten. Learning to link with Wikipedia. In *CIKM*. ACM, 2008. (Cited on page 15.)

[122] M. Mintz, S. Bills, R. Snow, and D. Jurafsky. Distant supervision for relation extraction without labeled data. In *ACL*. ACL, 2009. (Cited on pages 16, 51, 52, 59, and 68.)

[123] M. Mitra, A. Singhal, and C. Buckley. Improving automatic query expansion. In *SIGIR*. ACM, 1998. (Cited on page 66.)

[124] A. Mohan, Z. Chen, and K. Q. Weinberger. Web-search ranking with initialized gradient boosted regression trees. In *Yahoo! Learning to Rank Challenge*, 2011. (Cited on page 2.)

[125] V. Murdock and W. B. Croft. A translation model for sentence retrieval. In *EMNLP*. ACL, 2005. (Cited on page 17.)

[126] V. G. Murdock. *Aspects of Sentence Retrieval*. PhD thesis, University of Massachusetts Amherst, 2006. (Cited on page 13.)

[127] T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, and L. Deng. MS MARCO: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*, 2016. (Cited on page 75.)

[128] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich. A review of relational machine learning for knowledge graphs: From multi-relational link prediction to automated knowledge graph construction. *Proc. of the IEEE*, 104(1), 2016. (Cited on pages 1, 30, and 43.)

[129] V. Niculae, C. Suen, J. Zhang, C. Danescu-Niculescu-Mizil, and J. Leskovec. QUOTUS: The Structure of Political Media Coverage as Revealed by Quoting Patterns. In *WWW*. ACM, 2015. (Cited on pages 93 and 99.)

[130] R. Nogueira and K. Cho. Task-Oriented Query Reformulation with Reinforcement Learning. In *EMNLP*. ACL, 2017. (Cited on pages 68 and 76.)

[131] R. N. Oddy. Information retrieval through man-machine dialogue. *Journal of Documentation*, 33(1), 1977. (Cited on page 67.)

[132] K. D. Onal, Y. Zhang, I. S. Altingovde, M. M. Rahman, P. Karagoz, A. Braylan, B. Dang, H.-L. Chang, H. Kim, Q. McNamara, A. Angert, E. Banner, V. Khetan, T. McDonnell, A. T. Nguyen, D. Xu, B. C. Wallace, M. de Rijke, and M. Lease. Neural information retrieval: At the end of the early years. *IRJ*, 21(2–3), 2018. (Cited on page 70.)

[133] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. BLEU: A method for automatic evaluation of machine translation. In *ACL*. ACL, 2002. (Cited on page 36.)

[134] T. Pellissier Tanon, D. Vrandečić, S. Schaffert, T. Steiner, and L. Pintscher. From freebase to wikidata: The great migration. In *WWW*. ACM, 2016. (Cited on page 44.)

[135] B. Peng, X. Li, J. Gao, J. Liu, and K.-F. Wong. Deep Dyna-Q: Integrating Planning for Task-Completion Dialogue Policy Learning. In *ACL*. ACL, 2018. (Cited on page 65.)

[136] H. Peng, J. Liu, and C.-Y. Lin. News Citation Recommendation with Implicit and Explicit Semantics. In *ACL*. ACL, 2016. (Cited on page 96.)

[137] F. Petroni, P. Lewis, A. Piktus, T. Rocktäschel, Y. Wu, A. H. Miller, and S. Riedel. How context affects language models' factual predictions. In *AKBC*, 2020. (Cited on page 106.)

[138] D. Pighin, M. Cornolti, E. Alfonseca, and K. Filippova. Modelling events through memory-based, open-ie patterns for abstractive summarization. In *ACL*. ACL, 2014. (Cited on page 28.)

[139] G. Pirrò. Explaining and suggesting relatedness in knowledge graphs. In *ISWC*, 2015. (Cited on pages 49, 50, 54, and 58.)

[140] O. Popescu and C. Strapparava, editors. *Proceedings of the 2017 EMNLP Workshop: Natural Language Processing meets Journalism*. ACL, 2017. (Cited on page 4.)

[141] Y. Qiao, C. Xiong, Z. Liu, and Z. Liu. Understanding the Behaviors of BERT in Ranking. *arXiv preprint arXiv:1904.07531*, 2019. (Cited on pages 70 and 95.)

[142] C. Qu, L. Yang, M. Qiu, Y. Zhang, C. Chen, W. B. Croft, and M. Iyyer. Attentive History Selection for Conversational Question Answering. In *CIKM*. ACM, 2019. (Cited on page 65.)

[143] F. Radlinski and N. Craswell. A theoretical framework for conversational search. In *CHIIR*. ACM, 2017. (Cited on page 67.)

[144] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*, 21(140), 2020. (Cited on page 112.)

[145] D. Raghu, S. Indurthi, J. Ajmera, and S. Joshi. A statistical approach for Non-Sentential Utterance Resolution for Interactive QA System. In *SIGDIAL*. ACL, 2015. (Cited on pages 66 and 67.)

[146] S. Reddy, D. Chen, and C. D. Manning. CoQA: A Conversational Question Answering Challenge. *TACL*, 7, 2019. (Cited on page 65.)

[147] R. Reinanda, E. Meij, and M. de Rijke. Knowledge graphs: An information retrieval perspective. *FnTIR*, 2020. (Cited on page 1.)

[148] E. Reiter, R. Dale, and Z. Feng. *Building Natural Language Generation Systems*. MIT Press, 2000. (Cited on page 29.)

[149] P. Ren, Z. Chen, C. Monz, J. Ma, and M. de Rijke. Thinking globally, acting locally: Distantly supervised global-to-local knowledge selection for background based conversation. In *AAAI*, 2020. (Cited on page 68.)

[150] X. Ren, A. El-Kishky, C. Wang, F. Tao, C. R. Voss, and J. Han. Clustype: Effective entity recognition and typing by relation phrase-based clustering. In *KDD*. ACM, 2015. (Cited on page 59.)

[151] S. Riedel, L. Yao, and A. McCallum. Modeling relations and their mentions without labeled text. In *ECML-PKDD*. Springer-Verlag, 2010. (Cited on page 59.)

[152] A. Ritter, S. Clark, Mausam, and O. Etzioni. Named Entity Recognition in Tweets: An Experimental Study. In *EMNLP*. ACL, 2011. (Cited on page 68.)

[153] S. Robertson and H. Zaragoza. The Probabilistic Relevance Framework: BM25 and Beyond. *FnTIR*, 3 (4), 2009. ISSN 1554-0669, 1554-0677. (Cited on page 95.)

[154] M. Rospocher, M. van Erp, P. Vossen, A. Fokkens, I. Aldabe, G. Rigau, A. Soroa, T. Ploeger, and T. Bogaard. Building event-centric knowledge graphs from news. *JoWS*, 37-38, 2016. (Cited on pages 99, 103, and 113.)

[155] C. Rosset, C. Xiong, X. Song, D. Campos, N. Craswell, S. Tiwary, and P. Bennett. Leading Conversational Search by Suggesting Useful Questions. In *WWW*. ACM, 2020. (Cited on page 1.)

[156] B. Roth, T. Barth, M. Wiegand, and D. Klakow. A survey of noise reduction methods for distant supervision. In *AKBC*. ACM, 2013. (Cited on page 113.)

[157] T. Russell-Rose, J. Chamberlain, and L. Azzopardi. Information retrieval in the workplace: a comparison of professional search practices. *IPM*, 54(6), 2018. (Cited on page 2.)

[158] T. Saier and M. Färber. Semantic Modelling of Citation Contexts for Context-Aware Citation Recom-

mendation. In J. M. Jose, E. Yilmaz, J. Magalhães, P. Castells, N. Ferro, M. J. Silva, and F. Martins, editors, *ECIR*. Springer, 2020. (Cited on page 104.)

[159] G. Saldanha, O. Biran, K. McKeown, and A. Gliozzo. An entity-focused approach to generating company descriptions. In *ACL*. ACL, 2016. (Cited on pages 29 and 36.)

[160] F. Sarvi, N. Voskarides, L. Mooiman, S. Schelter, and M. de Rijke. A comparison of supervised learning to match methods for product search. In *eCOM 2020: The 2020 SIGIR Workshop on eCommerce*. ACM, 2020. (Cited on page 8.)

[161] S. Sauer. Audiovisual Narrative Creation and Creative Retrieval: How Searching for a Story Shapes the Story. *JSTA*, 9(2), 2017. (Cited on page 2.)

[162] A. See, P. J. Liu, and C. D. Manning. Get To The Point: Summarization with Pointer-Generator Networks. In *ACL*. ACL, 2017. (Cited on page 77.)

[163] V. Setty and K. Hose. Event2Vec: Neural Embeddings for News Events. In *SIGIR*. ACM, 2018. (Cited on page 106.)

[164] S. Seufert, K. Berberich, S. J. Bedathur, S. K. Kondreddi, P. Ernst, and G. Weikum. Espresso: Explaining relationships between entity sets. In *CIKM*. ACM, 2016. (Cited on page 58.)

[165] X. Shen, B. Tan, and C. Zhai. Context-sensitive information retrieval using implicit feedback. In *SIGIR*. ACM, 2005. (Cited on page 1.)

[166] G. Sidiropoulos, N. Voskarides, and E. Kanoulas. Knowledge graph simple question answering for unseen domains. In *AKBC*, 2020. (Cited on page 8.)

[167] M. Sloan, H. Yang, and J. Wang. A term-based methodology for query reformulation understanding. *Information Retrieval*, 18(2), 2015. (Cited on page 68.)

[168] C. L. Smith and S. Y. Rieh. Knowledge-Context in Search Systems: Toward Information-Literate Actions. In *CHIIR*. ACM, 2019. (Cited on page 1.)

[169] I. Soboroff, S. Huang, and D. Harman. Trec 2018 news track overview. In *TREC*. NIST, 2018. (Cited on page 92.)

[170] D. Sorokin and I. Gurevych. Context-aware representations for knowledge base relation extraction. In *EMNLP*. ACL, 2017. (Cited on page 52.)

[171] M. Surdeanu, M. Ciaramita, and H. Zaragoza. Learning to rank answers to non-factoid questions from web collections. *Computational Linguistics*, 37(2), 2011. (Cited on page 13.)

[172] M. Surdeanu, J. Tibshirani, R. Nallapati, and C. D. Manning. Multi-instance multi-label learning for relation extraction. In *EMNLP-CoNLL*. ACL, 2012. (Cited on page 59.)

[173] R. Sylvester and W.-l. Greenidge. Digital Storytelling: Extending the Potential for Struggling Writers. *The Reading Teacher*, 63(4), 2009. (Cited on page 2.)

[174] S. Teufel, J. Carletta, and M. Moens. An annotation scheme for discourse-level argumentation in research articles. In *EACL*. ACL, 1999. (Cited on page 106.)

[175] S. Teufel, A. Siddharthan, and D. Tidhar. Automatic classification of citation function. In *EMNLP*. ACL, 2006. (Cited on pages 106 and 114.)

[176] A. Tombros and M. Sanderson. Advantages of query biased summaries in information retrieval. In *SIGIR*. ACM, 1998. (Cited on pages 27 and 28.)

[177] M. Tsagkias, M. de Rijke, and W. Weerkamp. Linking online news and social media. In *WSDM 2011*. ACM, 2011. (Cited on page 15.)

[178] A. Turpin, Y. Tsegay, D. Hawking, and H. E. Williams. Fast generation of result snippets in web search. In *SIGIR*. ACM, 2007. (Cited on page 28.)

[179] S. Vakulenko, S. Longpre, Z. Tu, and R. Anantha. Question Rewriting for Conversational Question Answering. *arXiv:2004.14652 [cs]*, 2020. arXiv: 2004.14652. (Cited on page 113.)

[180] T. A. van Dijk. *News as discourse*. University of Groningen, 1988. (Cited on pages 89 and 91.)

[181] C. Van Gysel, E. Kanoulas, and M. de Rijke. Lexical query modeling in session search. In *ICTIR*. ACM, 2016. (Cited on pages 66, 68, and 78.)

[182] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is All you Need. In *NIPS*. 2017. (Cited on pages 66 and 71.)

[183] S. Verberne, J. He, U. Kruschwitz, G. Wiggers, B. Larsen, T. Russell-Rose, and A. P. de Vries. First International Workshop on Professional Search. *SIGIR Forum*, 52(2), 2019. (Cited on page 2.)

[184] N. Voskarides, D. Odijk, M. Tsagkias, W. Weerkamp, and M. de Rijke. Query-dependent contextualization of streaming data. In *ECIR*. Springer, 2014. (Cited on page 8.)

[185] N. Voskarides, E. Meij, M. Tsagkias, M. de Rijke, and W. Weerkamp. Learning to explain entity relationships in knowledge graphs. In *ACL-IJCNLP*. ACL, 2015. (Cited on pages 7, 11, 28, 31, and 58.)

[186] N. Voskarides, E. Meij, and M. de Rijke. Generating descriptions of entity relationships. In *ECIR*.

Springer, 2017. (Cited on pages 7, 27, 42, 52, and 58.)

[187] N. Voskarides, E. Meij, R. Reinanda, A. Khaitan, M. Osborne, G. Stefanoni, K. Prabhanjan, and M. de Rijke. Weakly-supervised contextualization of knowledge graph facts. In *SIGIR*. ACM, 2018. (Cited on pages 7, 41, 68, and 81.)

[188] N. Voskarides, D. Li, A. Panteli, and P. Ren. ILPS at TREC 2019 Conversational Assistant Track. TREC, NIST, 2019. (Cited on page 8.)

[189] N. Voskarides, D. Li, P. Ren, E. Kanoulas, and M. de Rijke. Query resolution for conversational search with limited supervision. In *SIGIR*. ACM, 2020. (Cited on pages 8 and 65.)

[190] N. Voskarides, E. Meij, S. Sauer, and M. de Rijke. News article retrieval in context for event-centric narrative creation. In *Under submission*, 2020. (Cited on pages 8 and 89.)

[191] A. Vtyurina, D. Savenkov, E. Agichtein, and C. L. A. Clarke. Exploring conversational search with humans, assistants, and wizards. In *CHI*. ACM, 2017. (Cited on page 67.)

[192] L. Wang, J. Lin, and D. Metzler. A cascade ranking model for efficient ranked retrieval. In *SIGIR*. ACM, 2011. (Cited on pages 69 and 94.)

[193] J. Weston, A. Bordes, O. Yakhnenko, and N. Usunier. Connecting language and knowledge bases with embedding models for relation extraction. In *EMNLP*. ACL, 2013. (Cited on page 11.)

[194] R. W. White, P. Bailey, and L. Chen. Predicting user interests from contextual information. In *SIGIR*. ACM, 2009. (Cited on page 1.)

[195] S. Wiseman, S. Shieber, and A. Rush. Learning Neural Templates for Text Generation. In *EMNLP*. ACL, 2018. (Cited on page 112.)

[196] I. Witten and D. Milne. An effective, low-cost measure of semantic relatedness obtained from wikipedia links. In *AAAI Workshop on Wikipedia and Artificial Intelligence: an Evolving Synergy*. AAAI, 2008. (Cited on pages 16 and 17.)

[197] F. Wu and D. S. Weld. Open information extraction using wikipedia. In *ACL*. ACL, 2010. (Cited on page 14.)

[198] F. Wu, Y. Qiao, J.-H. Chen, C. Wu, T. Qi, J. Lian, D. Liu, X. Xie, J. Gao, W. Wu, and M. Zhou. MIND: A Large-scale Dataset for News Recommendation. In *ACL*. ACL, 2020. (Cited on page 95.)

[199] Q. Wu, C. J. Burges, K. M. Svore, and J. Gao. Ranking, boosting, and model adaptation. Technical report, Microsoft Research, 2008. (Cited on page 36.)

[200] X. Xu, O. Dušek, J. Li, V. Rieser, and I. Konstas. Fact-based Content Weighting for Evaluating Abstractive Summarisation. In *ACL*. ACL, 2020. (Cited on page 112.)

[201] H. Yang, D. Guan, and S. Zhang. The query change model: Modeling session search as a markov decision process. *TOIS*, 33(4), 2015. (Cited on pages 66, 68, and 77.)

[202] P. Yang, H. Fang, and J. Lin. Anserini: Enabling the Use of Lucene for Information Retrieval Research. In *SIGIR*. ACM, 2017. (Cited on pages 95 and 96.)

[203] W. Yang, H. Zhang, and J. Lin. Simple applications of BERT for ad hoc document retrieval. *arXiv preprint arXiv:1903.10972*, 2019. (Cited on pages 70, 71, 95, and 113.)

[204] Y. Yang, W. Chen, Z. Li, Z. He, and M. Zhang. Distantly Supervised NER with Partial Annotation Learning and Reinforcement Learning. In *COLING*. ACL, 2018. (Cited on page 68.)

[205] X. Yao, B. Van Durme, and P. Clark. Automatic coupling of answer extraction and information retrieval. In *ACL*. ACL, 2013. (Cited on page 21.)

[206] W. V. Yarlott, C. Cornelio, T. Gao, and M. Finlayson. Identifying the Discourse Function of News Article Paragraphs. In *Workshop on Events and Stories in the News 2018*. ACL, 2018. (Cited on page 89.)

[207] M. Yatskar. A Qualitative Comparison of CoQA, SQuAD 2.0 and QuAC. In *NAACL-HLT*. ACL, 2019. (Cited on page 66.)

[208] W.-t. Yih, M.-W. Chang, X. He, and J. Gao. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *ACL*. ACL, 2015. (Cited on pages 1, 30, and 41.)

[209] S. Yu, J. Liu, J. Yang, C. Xiong, P. Bennett, J. Gao, and Z. Liu. Few-Shot Generative Conversational Query Rewriting. In *SIGIR*. ACM, 2020. (Cited on page 113.)

[210] H. Zamani, S. Dumais, N. Craswell, P. Bennett, and G. Lueck. Generating Clarifying Questions for Information Retrieval. In *WWW*. ACM, 2020. (Cited on page 1.)

[211] H.-J. Zeng, Q.-C. He, Z. Chen, W.-Y. Ma, and J. Ma. Learning to cluster web search results. In *SIGIR*. ACM, 2004. (Cited on page 2.)

[212] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *SIGIR*. ACM, 2001. (Cited on page 69.)

[213] A. Zubiaga, H. Ji, and K. Knight. Curating and contextualizing Twitter stories to assist with social newsgathering. In *IUI*. ACM, 2013. (Cited on pages 4, 89, and 104.)

Summary

Search engines leverage knowledge to improve information access. Such knowledge comes in different forms: unstructured knowledge (e.g., textual documents) and structured knowledge (e.g., relationships between real-world objects and topics). In order to effectively leverage knowledge, search engines should account for context, i.e., additional information about the user and the query. In this thesis, we aim to support search engines in leveraging knowledge while accounting for different types of context.

In the first part of this thesis, we study how to make structured knowledge more accessible to the user when the search engine proactively provides such knowledge as context to enrich search results. We focus on knowledge graphs (KGs), which store world knowledge in the form of facts, i.e., relationships between entities (e.g., persons, locations, organizations). Since KG facts are stored in a formal form, they are not suitable for presentation to end users. As a first task, we study how to retrieve natural language descriptions of KG facts from a text corpus. We propose a method that successfully extracts and then ranks descriptions of KG facts. The method breaks down when a description for a certain KG fact does not exist. This leads us to our second task, where we study how to automatically generate KG fact descriptions. We propose a method that first creates sentence templates and then fills them with relevant information from the KG. KG fact descriptions often contain mentions to other related facts that can increase the understanding of the fact as a whole. As a third task, we study how to contextualize KG facts, that is, automatically find facts related to a query fact. We propose a method that enumerates KG facts in the neighborhood of the query fact and then ranks them with respect to their relevance to the query fact.

In the second part of this thesis, we move to a different research theme and study how to improve interactive knowledge gathering. We focus on conversational search, where the user interacts with the search engine to gather knowledge over large unstructured knowledge repositories. Here, the search engine should account for context that stems from interactions between the user and the search engine in a conversational search session. We focus on multi-turn passage retrieval as an instance of conversational search. A prominent challenge is that the current turn query may be underspecified. Thus, we need to perform query resolution, that is, add missing context from the conversation history to the current turn. We propose to model query resolution as a term classification task and propose a method to address it.

In the third and final part of this thesis, we focus on a specific type of search engine users, professional writers in the news domain. We study how to support such writers create event-narratives by exploring knowledge from a corpus of news articles. We focus on a scenario where the writer has already generated an incomplete narrative that consists of a main event and a context, and aims to retrieve news articles that discuss relevant events from the past. We formally define the task of news article retrieval in context for event-centric narrative creation. We propose a retrieval dataset construction procedure for this task that relies on existing news articles to simulate incomplete narratives and relevant articles. We study the performance of multiple rankers, lexical and semantic, and perform an in-depth quantitative and qualitative analysis to acquire insights into the characteristics of this task.

Samenvatting

Zoekmachines maken gebruik van kennis om de toegang tot informatie te verbeteren. Dergelijke kennis komt in verschillende vormen voor: ongestructureerde kennis (bijv. tekstdocumenten) en gestructureerde kennis (bijv. relaties tussen objecten in de echte wereld en onderwerpen). Om kennis effectief te benutten, moeten zoekmachines rekening houden met de context, in dit geval, aanvullende informatie over de gebruiker en de zoekopdracht. In dit proefschrift willen we zoekmachines ondersteunen bij het benutten van kennis en tegelijkertijd rekening houden met verschillende soorten context.

In het eerste deel van dit proefschrift bestuderen we hoe gestructureerde kennis toegankelijker voor de gebruiker kan worden gemaakt wanneer de zoekmachine proactief kennis zoals context verschaft om zoekresultaten te verrijken. We richten ons op kennisgrafen (KG's), die wereldkennis opslaan in de vorm van feiten, d.w.z. relaties tussen entiteiten (bijv. personen, locaties, organisaties). Omdat KG-feiten in een formele vorm worden opgeslagen, zijn ze niet geschikt voor presentatie aan eindgebruikers. Als eerste taak bestuderen we hoe we beschrijvingen van KG-feiten in natuurlijke taal uit een tekstcorpus kunnen halen. We stellen een methode voor die met succes de beschrijving van KG-feiten extraheert en rangschikt. De methode werkt echter niet als er geen beschrijving voor een bepaald KG-feit bestaat. Dit leidt ons naar onze tweede taak, waar we bestuderen hoe we automatisch KG-feitbeschrijvingen kunnen genereren. We stellen een methode voor die eerst zinssjablonen maakt en deze vervolgens vult met relevante informatie uit de KG. KG-feitbeschrijvingen bevatten vaak vermeldingen van andere gerelateerde feiten die het begrip van het feit als geheel kunnen vergroten. Als derde taak bestuderen we hoe we KG-feiten kunnen contextualiseren, dat wil zeggen, automatisch feiten vinden die verband houden met een vraagfeit. We stellen een methode voor die KG-feiten opsomt in de buurt van het vraagfeit en deze vervolgens rangschikt met betrekking tot hun relevantie voor het vraagfeit.

In het tweede deel van dit proefschrift stappen we over op een ander onderzoeksthema en bestuderen we hoe we interactieve kennisvergaring kunnen verbeteren. We richten ons op conversational search, waarbij de gebruiker interactie heeft met de zoekmachine om kennis te vergaren over grote ongestructureerde kennisverzamelingen. Hier moet de zoekmachine rekening houden met de context die voortkomt uit interacties tussen de gebruiker en de zoekmachine in een conversatiezoeksessie. We richten ons op het ophalen van passages in meerdere beurten als een voorbeeld van *conversational search*. Een prominente uitdaging is dat de vraag voor een enkele beurt mogelijk te weinig is gespecificeerd. We moeten dus een zoekopdrachtresolutie uitvoeren, dat wil zeggen: ontbrekende context uit de gespreksgeschiedenis toevoegen aan de huidige beurt. We stellen voor om zoekopdrachtresolutie te modelleren als een termclassificatietaak en stellen een methode voor om deze aan te pakken.

In het derde en laatste deel van dit proefschrift richten we ons op een specifiek type gebruikers van zoekmachines, namelijk professionele schrijvers in het nieuwsdomein. We bestuderen hoe we dergelijke schrijvers kunnen ondersteunen bij het creëren van verhalen over gebeurtenissen door kennis uit een corpus van nieuwsartikelen te verkennen. We richten ons op een scenario waarin de schrijver al een onvolledig verhaal heeft geschreven dat bestaat uit een hoofdgebeurtenis en een context, en beoogt nieuwsartikelen op te halen die relevante gebeurtenissen uit het verleden bespreken. We definiëren de taak van het ophalen van nieuwsartikelen formeel in de context voor het creëren van op gebeurtenissen gerichte verhalen. We stellen voor deze taak een procedure voor het construeren van een dataset voor, die gebaseerd is op bestaande nieuwsartikelen om onvolledige verhalen en relevante artikelen te simuleren. We bestuderen de prestaties van meerdere rangschikmodellen, lexicaal en semantisch, en voeren een diepgaande kwantitatieve en kwalitatieve analyse uit om inzicht te verwerven in de kenmerken van deze taak.