Improvements of a local bibliographic citation recommendation model based on hyperparameter and query optimization

Medić, Zoran

University of Zagreb

FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Zoran Medić

IMPROVEMENTS OF A LOCAL BIBLIOGRAPHIC CITATION RECOMMENDATION MODEL BASED ON HYPERPARAMETER AND QUERY OPTIMIZATION

DOCTORAL THESIS

Zagreb, 2023

University of Zagreb

FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Zoran Medić

IMPROVEMENTS OF A LOCAL BIBLIOGRAPHIC CITATION RECOMMENDATION MODEL BASED ON HYPERPARAMETER AND QUERY OPTIMIZATION

DOCTORAL THESIS

Supervisor: Professor Jan Šnajder, PhD

Zagreb, 2023

Sveučilište u Zagrebu

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zoran Medić

POBOLJŠANJA MODELA ZA LOKALNO PREPORUČIVANJE BIBLIOGRAFSKIH IZVORA TEMELJENA NA OPTIMIZACIJI HIPERPARAMETARA I UPITA

DOKTORSKI RAD

Mentor: prof. dr. sc. Jan Šnajder

Zagreb, 2023.

The doctoral thesis has been made at the University of Zagreb, Faculty of Electrical Engineering and Computing, at the Department of Electronics, Microelectronics, Computer and Intelligent Systems, as part of the Text Analysis and Knowledge Engineering Laboratory (TakeLab).

Supervisor: Professor Jan Šnajder, PhD

The doctoral thesis consists of: 88 pages

Doctoral thesis num.:

About the Supervisor

Jan Šnajder has received his BSc, MSc, and PhD degrees in Computer Science from the University of Zagreb, Faculty of Electrical Engineering and Computing (FER), Zagreb, Croatia, in 2002, 2006, and 2010, respectively. From September 2002 he was working as a research assistant, from 2011 as Assistant Professor, from 2016 as Associate Professor, and from 2021 as Full Professor at the Department of Electronics, Microelectronics, Computer and Intelligent Systems at FER. He was a visiting researcher at the Institute for Computational Linguistics at the University of Heidelberg, the Institute for Natural Language Processing at the University of Stuttgart, the National Institute of Information and Communications Technology in Kyoto, and the University of Melbourne. He participated in a number of research and industry projects in the field of natural language processing and machine learning. He has been the principal investigator on projects funded by the CSF and HAMAG-BICRO, and work package leader on ESF, UKF, and HORIZON projects. He has (co-) authored more than 120 papers in journals and conferences in natural language processing and information retrieval, and has been reviewing for major journals and conferences in the field. He is the lecturer in charge for six courses at FER and has supervised and co-supervised more than 100 BA and MA theses. He is a member of IEEE, ACM, ACL, the secretary of the Croatian Language Technologies Society, the co-founder and secretary of the Special Interest Group for Slavic NLP of the Association for Computational Linguistics (ACL SIGSLAV). He is a member of the Centre of Research Excellence for Data Science and Advanced Cooperative Systems and the associate editor of the Journal of Computing and Information Technology. He has been awarded the Silver Plaque "Josip Lončar" in 2010, the Croatian Science Foundation fellowship in 2012, the fellowship of the Japanese Society for the Promotion of Science in 2014, and the Endeavour Fellowship of the Australian Government in 2015.

O mentoru

Jan Šnajder diplomirao je, magistrirao i doktorirao u polju računarstva na Sveučilištu u Zagrebu Fakultetu elektrotehnike i računarstva (FER), 2002., 2006. odnosno 2010. godine. Od 2002. godine radio je kao znanstveni novak, od 2011. godine kao docent, od 2016. godine kao izvanredni profesor, a od 2021. kao redoviti profesor na Zavodu za elektroniku, mikroelektroniku, računalne i inteligentne sustave FER-a. Usavršavao se na Institutu za računalnu lingvistiku Sveučilišta u Heidelbergu, Institutu za obradu prirodnog jezika Sveučilišta u Stuttgartu, Nacionalnome institutu za informacijske i komunikacijske tehnologije u Kyotu te Sveučilištu u Melbourneu. Sudjelovao je na nizu znanstvenih i stručnih projekata iz područja obrade prirodnog jezika i strojnog učenja. Bio je voditelj projekata financiranih of strane HRZZ-a,

HAMAG-BICRO-a, ESF-a, UKF-a, te HORIZON projekata. Autor je ili suautor više od 120 znanstvenih radova u časopisima i zbornicima međunarodnih konferencija u području obrade prirodnog jezika i pretraživanja informacija te je bio recenzentom za veći broj časopisa i konferencija iz tog područja. Nositelj je šest predmeta na FER-u te je bio mentorom ili sumentorom studentima na više od 100 preddiplomskih i diplomskih radova. Član je stručnih udruga IEEE, ACM, ACL, tajnik Hrvatskoga društva za jezične tehnologije te suosnivač i tajnik posebne interesne skupine za obradu prirodnog jezika za slavenske jezike pri udruzi za računalnu lingvistiku (ACL SIGSLAV). Član je Znanstvenog centra izvrsnosti za znanost o podacima i kooperativne sustave te je pridruženi urednik časopisa Journal of Computing and Information Technology (CIT). Dobitnik je Srebrne plakete "Josip Lončar" 2010. godine, stipendije Hrvatske zaklade za znanost 2012. godine, stipendije Japanskog društva za promicanje znanosti 2014. godine te stipendije australske vlade Endeavour 2015. godine.

"*I knew exactly what to do. But in a much more real sense, I had no idea what to do.*" – Michael G. Scott, The Office

With the thesis written, submitted, and successfully defended, I am grateful to everyone who helped me figure out what to do along the way and to conclude this life chapter successfully.

To my friends and colleagues, thank you for all the moments of laughter and joy that helped me clear my mind and made this journey very enjoyable.

To my mom, dad, and sister, thank you for your unconditional love and support throughout my life. This accomplishment is yours as much as mine.

To my mentor, the best I could have wished for, thank you for many lessons in NLP, as well as in life. I am forever grateful for knowing what to do when it comes to choosing a PhD mentor.

Abstract

The thesis focuses on analyzing improvements of a natural language processing and deep learning-based (DL) model for local citation recommendation (LCR) based on optimizing design choices when building LCR systems and enhancing LCR queries by combining LCR with global citation recommendation (GCR) queries. Improving LCR systems on their own could help scientists find related work more easily in the process of writing scientific articles or when searching for relevant articles for any idea they might have. While the research on LCR presented various successful models for the task, many influential design choices have been overlooked in training such models. Moreover, previous research has not dealt with tasks that fit between the general recommendations obtained by GCR and the very specific ones obtained by LCR, which are arguably more practically relevant.

The thesis consists of three main parts. In the first part, we analyze how enhancing citation context used as a query for the LCR model can be used to account for the possible lack of information in the context. We propose including the title and abstract of an article from which the context was extracted as global information and using it to obtain a richer context representation in a DL-based model. The second part of the thesis focuses on analyzing how optimizing hyperparameters of a prefiltering model in a two-stage pipeline for LCR affects the system's performance and whether different training regimes for the training of a reranking model in such a pipeline can improve performance. We additionally evaluate negative sampling strategies for training a triplet-based reranking model for dense retrieval. The last part of the thesis focuses on the proposal of a new task called paragraph-level citation recommendation (PCR), which strikes a balance between GCR and LCR in the specificity of recommendations obtained by the two. We propose a task in which a model takes a topic sentence from a paragraph as input and outputs the recommendations for citing in the rest of the paragraph. We also present a model for the task that outperforms the baseline models, bringing insights into the task difficulty and potential further improvements. Both the experiments with the LCR and PCR models demonstrate the difficulty and the potential of further work on improving CR systems that could make them ready for integration and widespread use in the scientific community.

Keywords: citation recommendation, local citation recommendation, triplet-based reranking models, paragraph-level citation recommendation, natural language processing, deep learning

Prošireni sažetak

Poboljšanja modela za lokalno preporučivanje bibliografskih izvora temeljena na optimizaciji hiperparametera i upita

Broj objavljenih znanstvenih članaka raste sve većom brzinom u posljednjim godinama, što znanstvenicima i drugim zainteresiranima otežava pristup informacijama objavljenima u člancima, budući da postaje nemoguće u kratkom roku obraditi toliku količinu teksta. Kako bi se olakšao pristup informacijama iz znanstvenih članaka, oblikovani su brojni sustavi za pretraživanje znanstvenih članaka, koji u pozadini koriste modele strojnog učenja temeljene na metodama obrade prirodnoga jezika (OPJ, engl., *natural language processing*) i pretraživanja informacija (PI, engl., *information retrieval*) pomoću kojih analiziraju tekstove znanstvenih članaka. Dodatno, kombiniranje metoda iz OBJ-a i PI-a s metodama iz područja dubokog učenja (DU), u kojemu se koriste modeli temeljeni na dubokim neuronskim mrežama, dovodi do znatnog poboljšanja performanci modela, što ujedno i širi područje primjene tih metoda u različitim domenama.

Potreba za poboljšanjem performanci i proširenjem mogućnosti takvih sustava dovela je do razvoja novog područja u domeni OPJ-a i PI-a koje je usmjereno na zadatke obrade znanstvenih dokumenata. Među brojnim zadacima u tom području, koji uključuju provjeru činjenica, ekstrakciju imenovanih entiteta te klasifikaciju tema članaka, zadatak koji je možda od najveće koristi pri pisanju znanstvenih članaka je zadatak preporučivanja bibliografskih izvora (PBI) u tekstovima znanstvenih članaka. Budući da su u tekstovim članaka autori dužni usporediti svoj rad s postojećim radovima na tu temu te takve bibliografske izvore referencirati u svome tekstu, sustavi koji su sposobni uspješno riješiti PBI mogli bi biti od pomoći znanstvenicima u današnjem informacijama prezasićenom svijetu.

Ovaj rad ispituje načine na koji se mogu poboljšati performance modela strojnog učenja, i to modela temeljenog na dubokom učenju, za lokalno preporučivanje bibliografskih izvora (LPBI) u tekstovima znanstvenih članaka. Zadatak LPBI definiran je kao zadatak u kojemu model na ulazu prima tekst koji predstavlja dio teksta iz samoga znanstvenog članka, tj. kontekst, za koji se traži preporuka bibliografskog izvora za citiranje, dok na izlazu generira listu znanstvenih članaka koji su poredani prema mjeri relevantnosti za citiranje u danome kontekstu. Za razliku od zadatka LPBI, druga varijanta zadatka PBI je globalno preporučivanje bibliografskih izvora (GPBI), u kojem model na ulazu uobičajeno dobiva naslov i sažetak znanstvenog članka, dok na izlazu također generira listu preporučenih članaka. U dosadašnjim radovima na temu zadatka LPBI, nije se previše pažnje posvećivalo pitanju ima li kontekst koji je na ulazu modela dovoljno informacija na temelju kojih se može donijeti odluka o relevantnom članku za citiranje. I u zadatku LPBI, kao i u zadatku GPBI, modeli na raspolaganju imaju vrlo velik broj članaka iz kojih ekstrahiraju preporuke. Tipično se sustavi za zadatak PBI konstruiraju u dvije faze, gdje

model iz prve faze obavlja zadatak prethodnog filtriranja te generira manji skup relevantnih članaka koji onda model iz druge faze rerangira i generira konačnu listu preporuka. Premda su sustavi za zadatak PBI često konstruiraju na taj način, neki od izbora oblikovanja pri izgradnji takvih sustava često su bili zanemarivani. Primjerice, mnogi radovi iz literature o PBI-u koriste isti skup podataka za treniranje modela u obje faze sustava, iako se u stvarnosti pogreške modela iz prve faze propagiraju u model iz druge faze, što dovodi do razlike pri treniranju i evaluaciji modela, koja može štetiti performancama modela. Mnogi drugi parametri pri oblikovanju i izgradnji takvih sustava također su često zanemareni.

Sustavi za LPBI generiraju preporuke bibliografskih izvora za citiranje koje odgovaraju danom kontekstu, pri čemu se očekuje da je kontekst vrlo detaljan i u dovoljnoj mjeri opisuje referencirani bibliografski izvor, budući da upravo takvi konteksti sačinjavaju i skup za treniranje, koji je dobiven obradom tekstova dostupnih znanstvenih članaka i njihovih bibliografskih izvora. S druge strane, sustavi za GPBI generiraju preporuke bibliografskih izvora koje su često općenite i raznolike, budući da se odnose na širu temu članka koji je modelu predočen na ulazu. S obzirom na tu raznolikost među preporukama dobivenim različitim modelima za PBI, postojeći zadatci za PBI od upitne su koristi za cjelokupan postupak pisanja znanstvenih članaka, pogotovo za faze u kojima autori žele pronaći bibliografske izvore za kontekste koji nisu dovoljno specifični. Primjerice, u fazi pisanja poglavlja o srodnim radovima iz literature, autori mogu napisati samo tematsku rečenicu za odlomak koji trenutno pišu, a da ne znaju dovoljno o izvorima koji su relevantni za tu rečenicu. U takvim slučajevima, ni sustavi za LPBI, a ni sustavi za GPBI ne bi bili prikladni za korištenje, budući da su trenirani na drugačijim ulaznim podacima.

Ovaj se rad bavi načinima poboljšanja performanci modela za LPBI pomoću proširenja informacija u kontekstu s informacijama tipično korištenima u zadatku GPBI te analizama utjecaja različitih izbora oblikovanja sustava za LPBI izgrađenih u dvije faze. Pomoću eksprimenata na skupovima podataka znanstvenih članaka analizirano je jesu li neke odluke u oblikovanju takvih sustava bile nepravedno zanemarene u prethodnim radovima te može li pažljiv odabir tih parametara oblikovanja dovesti do poboljšanja performanci sustava za LPBI. Također, u radu je predložen nov zadatak za PBI: preporučivanje bibliografskih izvora na razini odlomka (OPBI), koje je prikladnije za središnje faze pisanja znanstvenih radova od postojećih zadataka PBI-a. Za zadatak OPBI, predstavljen je skup podataka te model strojnog učenja koji pri vrednovanju ostvaruje bolje rezultate od referentnih modela.

1. Uvod

U prvome poglavlju ("*Introduction*") predstavljena je motivacija te ciljevi doktorskoga istraživanja. Opisan je zadatak PBI te varijante zadatka GPBI i LPBI. Nakon toga navedeni su nedostatci postojećih prevladavajućih pristupa zadatku LPBI-a te motivacija za uvođenje novog

zadatka za preporuku bibliografskih izvora na razini odlomka. Konačno, opisan je izvorni znanstveni doprinos ovog rada te je dan kratak pregled poglavlja od kojih se rad sastoji.

2. Rangiranje teksta

U drugome poglavlju ("*Text Ranking*") opisan je zadatak rangiranja teksta, budući da se zadatku PBI najčešće pristupa upravo na taj način. Dana je definicija zadatka rangiranja teksta te su u nastavku navedeni uobičajeno korišteni pristupi temeljeni na leksičkim i neuronskim modelima. Kod leksičkih modela opisan je model BM25 za zadatak rangiranja teksta, dok je kod neuronskih modela dan pregled razvoja neuronskih modela za rangiranje teksta kroz povijest, pri čemu je naglašena razlika između modela koji generiraju statične i kontekstualizirane reprezentacije teksta. Opisani su modeli temeljeni na povratnim neuronskim mrežama, kao i modeli temeljeni na arhitekturi transformera. Na kraju su opisane najčešće korištene mjere za vrednovanje rangiranja teksta, od kojih su neke korištene i u vrednovanju u ostalim poglavljima doktorskoga rada.

3. Preporučivanje bibliografskih izvora

U trećemu poglavlju ("*Citation Recommendation*") definiran je zadatak PBI kao zadatak rangiranja teksta u kojem se za dani upit traže preporuke bibliografskih izvora za citiranje. Predstavljena je razlika između zadataka LPBI i GPBI te je objašnjena razlika u izgradnji upita u obje varijante. Nakon toga navedeni su i opisani skupovi podataka korišteni u radovima na temu PBI-a. Konačno, u posebnim potpoglavljima opisani su radovi iz literature na temu GPBI-a te na temu LPBI-a.

4. Poboljšanje konteksta citiranja bibliografskog izvora za zadatak LPBI

U četvrtome poglavlju ("*Citation Context Enhancement for Local Citation Recommendation*") predstavljen je model za zadatak LPBI u kojemu je kontekst citiranja bibliografskog izvora proširen infromacijama iz naslova i sažetka članka iz kojeg je kontekst ekstrahiran (citirajući članak). U početku poglavlja, motivirana je potreba za proširenjem informacija iz konteksta, nakon čega je opisan neuronski model sastavljen od dva modula temeljena na dvosmjernoj povratnoj neuronskoj mreži i konvolucijskoj neuronskoj mreži. Modul temeljen na povratnoj neuronskoj mreži koristi mehanizam pozornosti kako bi kombinirao informacije iz konteksta citiranja bibliografskog izvora s informacijama iz naslova i sažetka citirajućeg članka te pomoću te kombinacije generirao vektor koji predstavlja neuronsku reprezentaciju upita. Isti modul, koristeći isti mehanizam pozornosti, generira i vektorsku reprezentaciju članka koji je kandidat za citiranje u danome kontekstu. Za izračun relevantnosti kandidata za citiranje u tekstu konteksta

korištena je mjera kosinusne sličnosti između dva vektora dobivena na izlazu modula. Modul temeljen na konvolucijskoj neuronskoj mreži obrađuje informacije o autorima članka kandidata za citiranje te informacije o broju citata koje je članak prikupio, kako bi modelirao mjeru bibliografske relevantnosti članka u znanstvenoj zajednici. Kombiniranjem obiju mjera dobiva se konačna mjere relevantnosti članka kandidata za citiranje u danome kontekstu.

Vrednovanje modela provedeno na dva često korištena skupa podataka u literaturi na temu zadatka LPBI, koji sadrže tekstove znanstvenih članaka na engleskome jeziku, pokazalo je da proširenje informacija u kontekstu naslovom i sažetkom citirajućeg članka povećava performance modela u skupovima podataka koji sadrže kratke kontekste. Takav rezultat upućuje na to da bi budući radovi na temu LPBI-a trebali uzeti u obzir veličinu konteksta te razmotriti i druge načine proširenja informacija u kontekstu, ako je riječ o kontekstu male veličine.

5. Izbori oblikovanja u ulančanom sustavu za zadatak LPBI

U petome poglavlju ("*Design Choices in the Local Citation Recommendation Pipeline System*") analizirana su tri odabrana izbora oblikovanja u izgradnji ulančanih sustava za LPBI sastavljena od dvije faze, pri čemu je model u prvoj fazi, tj. model prethodnog filtriranja, leksički model BM25, dok je model u drugoj fazi, tj. model rerangiranja, neuronski model iz poglavlja 4.

Prvi izbor oblikovanja koji je analiziran je utjecaj optimizacije hiperparametara modela iz prve faze na performance sustava. Naime, u prethodnim radovima optimizacija hiperparametara često je zanemarivana te su modeli korišteni s njihovim pretpostavljenim vrijednostima. Eksperimenti iz ovog poglavlja pokazali su da pažljiva optimizacija hiperparametara modela za prethodno filtriranje iz prve faze može značajno poboljšati performance modela za LPBI.

Drugi izbor oblikovanja koji je analiziran je odabir režima treniranja modela za rerangiranje. Dok su modeli iz literature dominatno koristili standardan režim treniranja u kojemu je skup upita za treniranje modela za rerangiranje identičan skupu upita za treniranje modela za prethodno filtriranje, u radu evaluiramo i tzv. "strogi" režim, u kojem se skup upita za treniranje modela za rerangiranje sastoji samo od upita za koje je model za prethodno filtriranje generirao skup kandidata u kojem se nalazi članak koji je i referenciran u tom upitu. Budući da se u standardnom režimu treniranja članak koji je referenciran u kontekstu naknadno uključuje u skup kandidata ako model nije uvrstio taj članak u isti skup, može doći do posmaka u podacima između podataka korištenih za treniranje i vrednovanje modela, budući da pri vrednovanju model neće imati na raspologanju umetnuti članak ako ga model za prethodno filtriranje nije uvrstio u skup. Vrednovanje ova dva režima pokazalo je da strogi režim, premda računalno zahtjevniji, dovodi do boljih rezultata modela na skupu za vrednovanje, što povezujemo s izbjegavanjem posmaka u podacima do kojeg dolazi sa standardnim režimom.

Konačno, treći izbor oblikovanja je odabir negativnih primjera pri treniranju modela za rerangiranje članaka temeljenih na gubitku trojki. Naime, modeli za rerangiranje tipično su

trenirani tako da između trojki sastavljenih od upita, relevantnog (pozitivnog) primjera i nerelevantnog (negativnog) primjera maksimiziraju sličnost između vektorskih reprezentacija upita i pozitivnog primjera u odnosu na udaljenost između upita i negativnog primjera. Prethodni radovi na temu razvoja i treniranja takvih modela ukazali su na velik utjecaj odabira negativnih primjera pri izgradnji skupa trojki korištenih za treniranje modela za rerangiranje. U eksperimentima vezanim uz ovaj izbor oblikovanja, vrednovane su različite strategije odabira negativnih primjera za treniranje modela u drugoj fazi ulančanog sustava za LPBI na dva različita skupa podataka. Vrednovanje je pokazalo da i ovaj parametar izgradnje sustava može imati ključnu ulogu u podizanju performanci modela, dok je odabir same strategije odabira negativnih primjera ovisan o skupu podataka.

6. Preporučivanje bibliografskih izvora na razini odlomka

U šestome poglavlju ("*Paragraph-level Citation Recommendation*") predložen je nov zadatak u području preporučivanja bibliografskih izvora u kojem model na temelju tematske rečenice odlomka treba preporučiti članke za citiranje u nastavku odlomka (OPBI). Poglavlje započinje s definicijom novog zadatka i opisom razlike između zadatka OPBI i druga dva standardna zadatka PBI. Nakon toga slijedi pregled skupa podataka koji je izgrađen za potrebe izgradnje modela za OPBI, a dobiven je iz postojećeg skupa podataka u kojem su rečenice iz poglavlja znanstvenih članaka koja opisuju povezana područja označene svojim diskursnim ulogama. Korištenjem dostupnog skupa podataka i klasifikatora za detekciju diskursnih uloga rečenica, izgrađen je prikladan skup za treniranje i vrednovanje modela za PPBI. U nastavku poglavlja, predstavljen je model temeljen na arhitekturi transformera koji za dani upit, kao i za dani članak, generira vektorsku reprezentaciju, a treniran je pomoću gubitka četvorki, gubitka sličnom gubitku trojki iz poglavlja 5. Vrednovanje modela pokazalo je da model ostvaruje bolje rezultate od referentnih modela, a dodatna analiza testnog skupa rezultirala je saznanjima o težini zadatka te o specifičnostima u području zadatka OPBI, poput onog da stariji članci češće završavaju na dnu liste preporuka generiranih modelima za OPBI. U istome su poglavlju dani i karakteristični primjeri iz skupa podataka u kojima su primijećene takve specifičnosti.

7. Rasprava i budući rad

U sedmome poglavlju ("*Discussion and Outlook*") raspravljeni su dobiveni rezultati te su dane smjernice za daljnji rad na temu izgradnje praktično iskoristivih modela za PBI. Među predloženim idejama za daljnji rad istaknuti su novi zadaci za predtreniranje modela za generiranje vektorskih reprezentacija upita i članaka, korištenje višezadaćnog učenja za treniranje modela za PBI te korištenje generativnih modela za generiranje teksta članka temeljenih na preporučenim člancima za citiranje.

8. Zaključak

U osmome poglavlju ("*Conclusion*") dan je pregled i sažetak provedenog istraživanja i dobivenih rezultata. Zaključeno je da su rezultati iz poglavlja u kojima se analiziraju utjecaj informacija u kontekstu zadatka LPBI te različitih izbora oblikovanja pri izgradnji sustava za LPBI potvrdili važnost iscrpne pretrage i pažljivog odabira hiperparametara modela za poboljšanje performanci sustava u cjelini. Rezultati iz područja novopredloženog zadatka OPBI potvrdili su da je riječ o teškom problemu, ali i ponudili alternativu postojećim modelima za PBI, koja je prikladnija za uporabu u ranijim fazama pisanja znanstvenog članka.

Ključne riječi: preporučivanje bibliografskih izvora, lokalno preporučivanje bibliografskih izvora, modeli za rerangiranje temeljeni na gubitku trojki, preporučivanje bibliografskih izvora na razini odlomka, obrada prirodnoga jezika, duboko učenje

Contents

1. Introduction 1
- 1.1. Contributions 6
- 1.2. Thesis structure 6

2. Text Ranking 7
- 2.1. Lexical Models 7
- 2.2. Neural Models 9
  - 2.2.1. Categorization of Neural Ranking Models 9
  - 2.2.2. From Word2Vec to Transformers 13
- 2.3. Evaluation Metrics 16

3. Citation Recommendation 19
- 3.1. Task 19
- 3.2. Datasets 21
- 3.3. Global Citation Recommendation Models 23
- 3.4. Local Citation Recommendation Models 27

4. Citation Context Enhancement for Local Citation Recommmendation 29
- 4.1. Model with Enhanced Context Representation 30
- 4.2. Model Training and Evaluation 33

5. Design Choices in the Local Citation Recommendation Pipeline System 36
- 5.1. Optimization of the Prefiltering Model 37
  - 5.1.1. Results 38
- 5.2. Comparison of Different Training Regimes 40
  - 5.2.1. Results 41
- 5.3. Negative Sampling Strategies 42
  - 5.3.1. Results 44

6. Paragraph-level Citation Recommendation 48
6.1. Task 49
6.2. Dataset 49
6.3. Paragraph-level Citation Recommendation Model 51
6.4. Evaluation 53

7. Discussion and Outlook 64
7.1. Findings 64
7.2. Future Work 65

8. Conclusion 68

Bibliography 70

Biography 86

Životopis 88

Chapter 1

Introduction

Scientific articles are widely recognized as a primary source of research information today. They are invaluable tools for storing and disseminating knowledge across all scientific fields. While digitalization and investment in research worldwide have accelerated science and led to many breakthroughs in different fields, the growth in scientific publishing is accompanied by the struggles of scientists to keep up with all the published work. A 2012 study among scientists from U.S. and Australian universities, described in [1], found that the scientists who took part in the study read an average of 22 articles per month, which amounts to the same number as in a similar study conducted in 2005 [2]. It seems that scientists have already reached their peak regarding the number of articles they can process in a fixed time frame.

As a solution for the state of information overload in which scientists have found themselves, different technological solutions have been proposed to ease access to information from scientific articles, not just for scientists but for any interested parties. For example, specialized search engines that index large databases of scientific articles, such as Google Scholar,[1] Microsoft Academic,[2], and Semantic Scholar,[3] have been proposed. These engines analyze text and metadata information from scientific articles, such as author names, publication venues, or citations, and enable users to search the published articles and keep track of the research they are interested in. Under the hood, these engines utilize various techniques from artificial intelligence and machine learning, in particular *natural language processing* (NLP) and *information retrieval* (IR). While NLP is a field in computer science focused on designing algorithms for processing and understanding data in the form of natural language, IR is a field that deals with the processing of data (often in textual form) that can be easily retrieved as relevant for a user's query. In combination with deep learning (DL) methods that led to improvements in many ML tasks in recent years, techniques from NLP and IR can result in powerful features that can enhance the existing search engines. For example, Semantic Scholar uses an NLP model that

[1] `https://scholar.google.com/`
[2] `https://academic.microsoft.com`
[3] `https://www.semanticscholar.org/`

analyses citations between the articles in its collection and detects which articles were influential for the work that cited them, thus potentially enabling users to retrieve the relevant work easily. As the interest in automated analysis of scientific literature grew, many tasks were introduced in the field of *scholarly document processing* (SDP), such as extraction of keyphrases or key aspects from articles [3, 4], argumentation mining over article's sentences [5, 6, 7], classification of citations into different categories [8, 9, 10], article summarization [11, 12], and citation recommendation [13, 14, 15].

One of the arguably more exciting tasks in SDP is *citation recommendation* (CR), the task of recommending an article from a pool of scientific articles that should be cited in a given text query posed by a user. Approaches to CR can be categorized into two types, depending on how the query in the task is constructed. In *global* CR (GCR) [13, 16, 17], a query is typically constructed using an article's title and abstract, and recommendations represent articles that should be cited anywhere in the article's text. In contrast, *local* CR (LCR) [14, 18, 19] assumes a query represents a text snippet from a manuscript, i.e., a *citation context*, and the recommendations represent articles that should be cited in that context. Defined in such a way, GCR's recommendations are considered more general and diverse, while LCR ones are specific for the context in the query. In both cases, obtaining the datasets for training the models for the tasks is done by parsing the collections of publicly available articles and the citations found in them, which enables a simple construction of pairs of queries and relevant articles.

Both GCR and LCR tasks are useful not only as complements to the scientific writing process in which authors need to cite the relevant work but also as proxy tasks that can be used to train models that recommend articles relevant to the user's query so that the user can learn more about specific topics. As scientific articles describe the cited articles in their corresponding citation contexts in the body of an article, they could be adapted and used to fine-tune the search engines that operate over large article databases. Moreover, with a growing interest in writing assistants, not just in science but also other domains, LCR models can be integrated into such assistants and provide both the citation contexts and the links to the cited articles. Undoubtedly, many other avenues for adaptation and usage of CR models exist today.

To make the LCR models ready for integration with scientific research assistants or proxy models for general search over scientific articles, it is important to ensure that recommendations for given contexts can be obtained with the information given to the model. Despite the increasing recent interest in developing LCR systems, not many works focused on analyzing whether the context's information is enough for the model to provide recommendations or if it could be enhanced with additional information to bridge the gap between the context's and the cited article's content. As citations in full texts of the articles serve different purposes, it is naive to expect each citation context to contain enough details for a model to confidently choose an article as a recommendation for citing.

Another overlooked phenomenon when developing CR systems relates to specific design choices when building such systems. As CR models operate over a large pool of articles from which they select the recommendations, the systems for CR are typically constructed as pipelines consisting of multiple stages, where each stage consists of a single model and processes the output from the previous stage. Having a system consist of multiple stages instead of a single one enables more efficient processing of a large pool of articles since the models in each stage can strike a balance between the efficiency and efficacy [20, 21]. For example, in a two-stage pipeline system, the first stage model does the prefiltering of the pool of articles that results in a set of candidate articles that is smaller than the pool, while the second stage model reranks the prefiltered articles and produces the final relevance scores. However, when building such a system, previous work did not account for some design choices that were shown to improve the performance of similar IR pipelines. Specifically, the optimization of the parameters of the first-stage model was often overlooked, although it might lead to an improvement in performance across the whole pipeline. In addition, as the pipeline models are separate, the question arises as to whether the training of the second model should also be independent of the first one or not. Furthermore, the choice of irrelevant documents used to train the model in the second stage has been found to have a lot of impact on the performance of the pipeline as a whole. Despite that, not much research was done in the domain of analyzing the impact of the choice of irrelevant documents for the training of LCR models.

While the specific design choices and the research on improving the context representation in LCR can improve the overall performance of these models, it still remains unclear what the practical benefits of using such models could be. Let's consider the process of writing a scientific article, where CR systems by design should be appropriate for use. Scientists typically start writing a manuscript with a rough draft and then proceed to expand their main thoughts and ideas into detailed paragraphs and sentences. Looking at the existing CR tasks, models for GCR seem the most appropriate for the initial stage of the writing process, where authors might look for general recommendations for the overall manuscript. In contrast, LCR recommendations are more suitable for the final stage of the writing process, where authors have already written detailed sentences describing the relevant articles and might only look for citations to insert in the text. However, the scientific writing process consists of many intermediate stages which differ in the specificity of the ideas and thoughts described in each stage. As the writing and the research process progresses, authors learn more about specific topics and the existing work, but before they do, they might only write more general sentences in the article that describe the existing work on a broad level. Given this difference in both the knowledge the authors have in the different stages of the writing process and the level of detail in the text in each stage, it is not clear whether existing CR tasks are sufficient to build writing assistants or CR models that are of practical use throughout the whole scientific writing process.

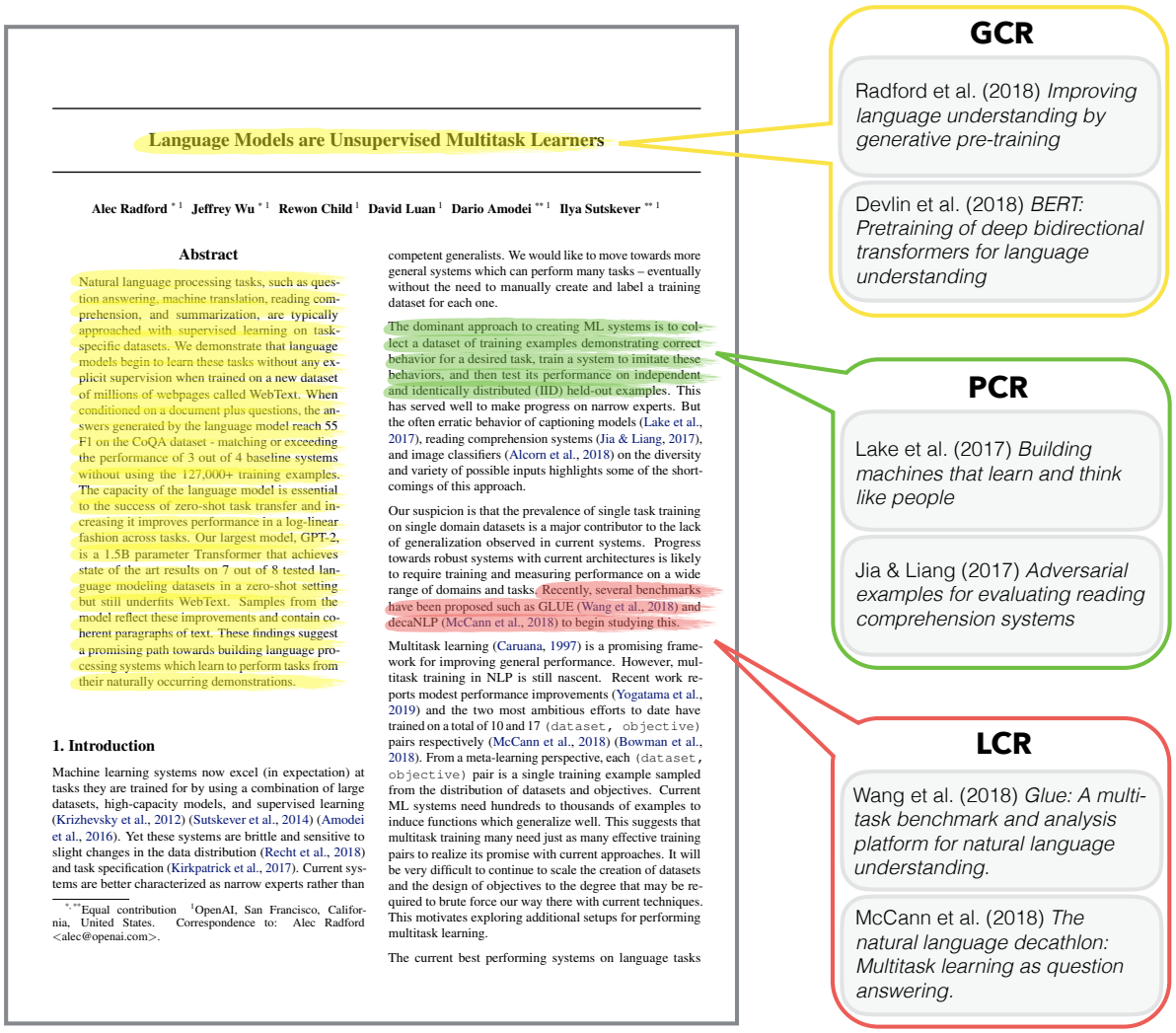


Figure 1.1: Illustration of the differences between variants of CR tasks. On the left side of the figure is the first page of the article "*Language Models are Unsupervised Multitask Learners*" by Radford et al. When a user is either writing the article or only looking at it, a CR system could provide the citation recommendation based on the article's content. For example, if a user is looking for general recommendations for the article at hand, a GCR model takes as input the article's title and abstract (in yellow) and might recommend the articles in the yellow box as suitable for citing. In another scenario, a user might highlight only a sentence or two from the article's text (in red), and an LCR model would recommend articles in the red box for citing. Finally, for the task we propose in the thesis, PCR, a corresponding model takes the paragraph's topic sentence as input (in green) and outputs a list of recommended articles as citations in the rest of the paragraph.

In this thesis, we aim to fill the aforementioned research gaps in the CR task by investigating how an LCR model's performance can be further improved and by introducing a new task that aims to expand CR models' applicability in practice. We start by analyzing a DL-based LCR model's performance with a citation context representation enhanced with the title and abstract of the article from which a context was extracted. We find that this additional information improves the model's performance in cases when the citation context's size is small. We then proceed to investigate the impact of the specific design choices on the performance of an LCR pipeline consisting of a prefiltering model in the first and the reranking model in the second stage. We find that, although often overlooked, these design choices can improve the system's performance upon the baseline models designed with default values for these design choices. Finally, we present a new task of paragraph-level citation recommendation (PCR) that takes a topic sentence from the beginning of a paragraph, builds a query based on that sentence, and aims to generate a list of articles to cite in the rest of the paragraph. Such a task might suit an intermediate scientific writing stage more than GCR and LCR. In Figure 1.1, we show how the new task we propose in the thesis fits between the two existing CR tasks when looking from a user's point of view.

To put it more concretely, this thesis addresses the following five research questions (RQs):

- **RQ1**: Can enhancing citation context with global information in a DL-based model for LCR improve the performance of the model?
- **RQ2**: Does optimization of prefiltering model's hyperparameters affect the performance of an LCR system?
- **RQ3**: Does using different training regimes for training the reranking model in the second stage of an LCR pipeline system affect the system's performance?
- **RQ4**: How does the performance of the LCR system change depending on the negative sampling strategy for the construction of training triplets in the training of a triplet-based reranking model in an LCR pipeline?
- **RQ5**: Can the existing datasets and models be used to design a new CR task and model better suited for the intermediate stages of the scientific article writing process?

The answers to the first four research questions can be used to improve the performance of LCR systems by incorporating the design choices that were found useful in our experiments. The last research question aims to improve the practicality of the CR task and investigates the potential of a new CR task. In the following chapters, we turn to each of these questions, conduct experiments and provide answers to each.

1.1 Contributions

This doctoral thesis's research objective revolves around improving a local bibliographic citation recommendation system based on deep learning. The thesis presents methods for improvement of the systems for local bibliographic citation recommendation through an exhaustive analysis of the impact that different hyperparameters of the system have on its accuracy and proposes a new task in the field of bibliographic citation recommendation that aims at improving the usability of such systems in practice. The original scientific contribution of this thesis consists of the following:

1. A deep learning model for local bibliographic citation recommendation based on enhanced context representation aiming to improve the system's prediction accuracy;
2. An analysis of the effect of the relevant hyperparameters of a deep learning-based local bibliographic citation recommendation system on the system's prediction accuracy;
3. A deep learning model for a new task of bibliographic citation recommendation with the optimization of queries based on topic sentences in paragraphs, with the goal of improving functionality of bibliographic citation recommendation systems.

1.2 Thesis structure

The thesis consists of eight chapters, including this one, which motivated and introduced the topic of the thesis and outlined its scientific contribution. In Chapter 2, we introduce the task of text ranking since the task of CR is typically framed as such and give an overview of the lexical and neural models used for text ranking alongside the evaluation metrics commonly used for evaluating text ranking models. Chapter 3 introduces the task of CR, i.e., its local and global variants, together with the datasets that exist and are used in work on CR. We conclude the chapter with a short review of the models introduced in the previous work on LCR and GCR. Chapter 4 starts with a description of the proposed model that uses an enhanced representation of the citation context in the task of LCR. We continue with a presentation of the evaluation results of the proposed model on two commonly used datasets in LCR. In Chapter 5, we further attempt to improve the performance of an LCR system by analyzing the effect that different hyperparameters have on the system's performance. Chapter 6 introduces a new task in CR, paragraph-level citation recommendation, and describes how the dataset for this new task was constructed. The chapter ends with the presentation of the model for the new task and the results of the evaluation of the model and comparison with baseline models. In Chapter 7, we discuss potential avenues for future work based on the results from the previous section, while in Chapter 8 we give an overview of the thesis highlighting the main conclusions.

Chapter 2

Text Ranking

The task of CR is typically approached as a text ranking problem. In text ranking, the goal is to generate a list of texts retrieved from a pool of texts and ordered by relevance for a given query [22]. The relevance scores of ordered texts represent how relevant these texts are for a given query. In CR, the queries represent articles or contexts for which citations are sought, while the pool of texts is actually a pool of the available scientific articles. The relevance scores produced in the end are interpreted as the values indicating how relevant an article is for citing in a given query.

This chapter will introduce the models typically used in text ranking tasks by following a more detailed overview of text ranking approaches from [22]. We start with lexical models based on the exact matching of words appearing in query and pool texts. Then we proceed to deep learning-based models that use different neural architectures to encode the text into vector representations and use these representations for calculating relevance scores. The chapter ends with an overview of the most common evaluation metrics used to evaluate the performance of text ranking models.

2.1 Lexical Models

Using computers to improve access to information stored in vast documents originated in [23], who proposed indexing the words in the pool's documents with weights that resemble probabilities that a user searching for that document would use that word in a query. This approach resembles traditional libraries, where librarians group the books and documents in the library using specific words occurring in the texts. This idea of automatic indexing of documents based on the words appearing in them was built upon by [24], who proposed a vector space model in which queries and documents are each represented as sparse "bag of words" vectors. In such a vector, each dimension corresponds to a word in the vocabulary, and the value of the dimension, in its basic form, represents whether a word appears in the text. The relevance score between

a query and a document can then be calculated as, for example, cosine similarity between the two vector representations. A significant contribution of [24] was the proposal of a weighting scheme called "term frequency – inverse document frequency" (*TF-IDF*), in which each term (a word or other token in text) is assigned a weight proportional to how many times the term appears in a query (TF) and inversely proportional to the number of documents containing the term (IDF). Using TF-IDF weights in vector representations of queries and documents ensured that these representations contained information about the relevance of each term in a document for a given query.

Many extensions of the TF-IDF weighting scheme were later proposed in the literature. However, arguably the most successful one for the task of text ranking was presented in [25], who described BM25 ("Best Match 25"), a scoring function based on term and document frequencies, that even today performs competitively with the state-of-the-art text ranking models. The BM25 ranking function is defined as follows. Given a query Q, containing terms $[q_1, ..., q_n]$, and a document D, BM25 calculates relevance score s as:

$$\mathrm{s}(Q,D) = \sum_{i=1}^{n} \mathrm{IDF}(q_i) \cdot \frac{f(q_i,D) \cdot (k_1+1)}{f(q_i,D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\mathrm{avgdl}}\right)} \tag{2.1}$$

where $f(q_i,D)$ is the frequency of q_i in document D, $|D|$ is the length of D in words, avgdl is the average document length in a pool of documents, and k_1 and b are parameters that can be tuned for a specific document collection. $\mathrm{IDF}(q_i)$ is the inverse document frequency for q_i, and is typically calculated as:

$$\mathrm{IDF}(q_i) = \ln\left(\frac{N - n(q_i) + 0.5}{n(q_i) + 0.5} + 1\right) \tag{2.2}$$

where N is the total number of documents in the collection and $n(q_i)$ is the number of documents containing the term q_i. Intuitively, the BM25 scoring function outputs higher scores for documents that contain many terms from Q that also do not appear often in other documents. Parameter b controls to what degree the length of a document will affect the final score [26], while k_1 controls to what degree an additional occurrence of a term affects the final score [27].

BM25 has been extensively used since its introduction, and it obtained state-of-the-art results in many IR tasks. Nowadays, many efficient algorithm implementations exist, making it easy to use in different scenarios. However, while providing a strong performance, the model lacks generalization capabilities, as it relies on the exact matching of the terms, which can sometimes lead to a suboptimal scoring of the texts. Neural models, which we describe next, try to overcome this limitation.

2.2 Neural Models

The success of BM25 can be attributed mostly to its simplicity and small complexity in space and time, typically accomplished using an inverted index structure [28]. However, reliance on the exact matching of the terms from the queries and the documents can hurt the performance. For example, if a user poses a query with a term that does not appear in any of the documents in the pool but the term's synonym does, a lexical model would miss out on retrieving those relevant documents. Such situations are examples of the *vocabulary mismatch problem* [29], one of the fundamental problems in IR.

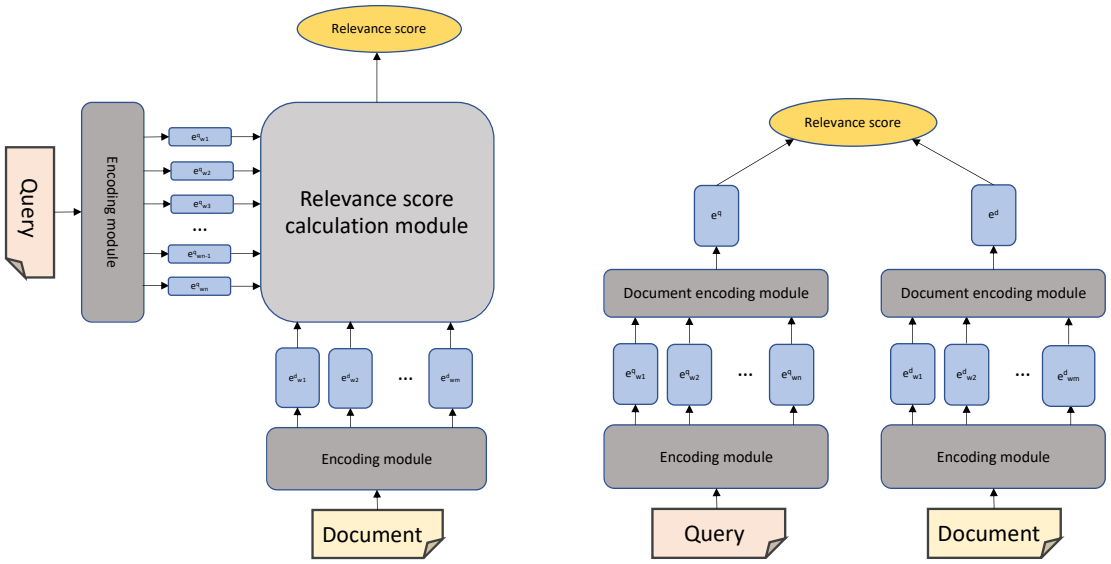
Neural models that encode words or sequences of words into continuous vector representations can tackle the vocabulary mismatch problem by producing similar vector representations for semantically similar words or word sequences. These vector representations can then be used to calculate relevance scores in text ranking setups and help alleviate the vocabulary mismatch problem present in lexical models.

This section will give an overview of the development of neural models for text ranking. We will start with categorizing neural models based on the level of interaction between the vector representations of queries and documents. After that, we will briefly outline the most prominent approaches to neural text ranking.

2.2.1 Categorization of Neural Ranking Models

Neural ranking models are based on neural modules that encode given text sequences into vector representations. These representations are typically called *embeddings*, and the models that produce them are typically called *encoders*. Encoders can output an embedding for each term appearing in a text sequence but can also output an embedding for the text's subsequences (such as sentences) or for a complete text sequence, even longer ones such as documents. Using the embeddings of queries and documents on the term or full-text level, neural ranking models produce recommendation scores for ranking the relevant documents.

Depending on whether the relevance scores for a query-document pair are calculated using the embeddings of the terms in the query and the document or using the embeddings of the query and the document's full texts, neural ranking models are categorized differently. More formally, let Q represent a query consisting of $[w_1^q, ..., w_n^q]$ terms. Let D represent a document consisting of $[w_1^d, ..., w_m^d]$ terms, and let $\mathbf{e}$ represent an embedding of a given text input x. An encoder can produce a matrix $\mathbf{E}_Q = [\mathbf{e}_{w_1}^q, ..., \mathbf{e}_{w_n}^q]$, where a row in a matrix represents a term embedding for each term appearing in Q, as well as a matrix $\mathbf{E}_D = [\mathbf{e}_{w_1}^d, ..., \mathbf{e}_{w_m}^d]$ for each term appearing in D. However, it can also output embeddings $\mathbf{e}_Q$ and $\mathbf{e}_D$ that encode the full texts of Q and D, respectively. Depending on whether the relevance score between a query Q and a document D is calculated based on a pair ($\mathbf{E}_Q$, $\mathbf{E}_D$) or only using a pair ($\mathbf{e}_Q$, $\mathbf{e}_D$), neural ranking models are



(a) Cross-encoder (b) Bi-encoder

Figure 2.1: An illustration of the differences between cross-encoder and bi-encoder design in text ranking. In the cross-encoder, shown in Subfigure 2.1a, terms from the query and the document are typically independently encoded until a certain point in the model, when they start to be combined in order to produce the final recommendation score. Here the combination of the term encodings from the query and the document happens inside the "Relevance score calculation module". On the other hand, in the bi-encoder design, shown in Subfigure 2.1b, the query and the document are independently encoded into two encodings, which are then processed to produce a single relevance score. The processing typically means calculating the cosine similarity or L2 distance between the two embeddings.

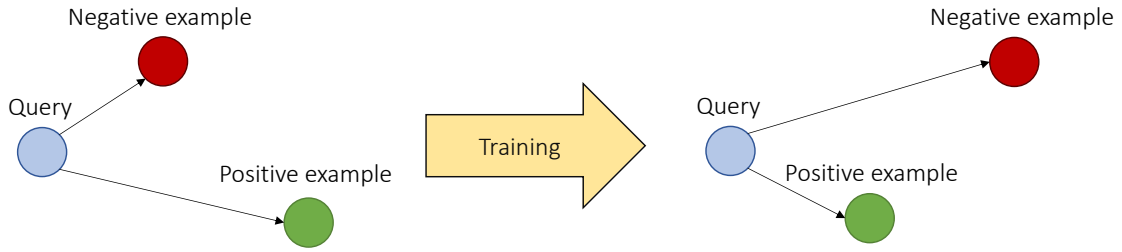


Figure 2.2: An illustration of a change in the arrangement of the query, positive, and a negative example before and after the training with the triplet loss. Before the training step, a negative example is closer to the query than the positive one, resulting in a non-zero triplet loss value when calculated on such a triplet. After training and updating the parameters, the model "pushes" the positive example closer to the query and a negative example further away.

categorized into interaction- and representation-based models, respectively [22]. Interaction-based models are typically called *cross-encoders*, while representation-based models are called *bi-encoders*. Figure 2.1 illustrates the difference between these two approaches.

In cross-encoders, the relevance score between Q and D is calculated using a custom function that takes all the term embeddings as input and outputs a single score. This custom function is typically implemented as a neural network module whose complexity depends on the module's building blocks. On the other hand, bi-encoders typically use simple functions for relevance score calculation, such as cosine similarity, dot product, or L2 distance between the full-text embeddings of Q and D, which boils down to the nearest neighbor search in a d-dimensional embedding space. The simplicity of the relevance score function makes the bi-encoders more convenient to use than cross-encoders, especially when dealing with a large number of query and document pairs. Paired with GPU-based methods for approximate nearest neighbor search, such as [30], the difference in time complexity when using bi-encoders compared to cross-encoders can be reduced even more.

Using DL-based bi-encoders to retrieve relevant documents in text ranking setups is typically called *dense retrieval*, as the retrieval is done over a set of continuous vector representations of documents. On the other hand, lexical models that create vector representations typically of the size of the vocabulary are categorized into *sparse retrieval* methods, as the vector representations are dominantly sparse, i.e., contain non-zero values only in the dimensions of the terms that appear in the queries and documents.

Both cross- and bi-encoders require a training set consisting of a set of queries $\{Q^{(i)}\}_{i=1}^{N}$, a pool of documents $\{D^{(i)}\}_{i=1}^{M}$, and a set of N_i relevant documents $D_R^{(i)} = \{D^{(i)}\}_{i=1}^{N_i} \subset \{D^{(i)}\}_{i=1}^{M}$ for each query $Q^{(i)}$. Additionally, depending on the task, each relevant document might be annotated with a relevance score, indicating that some are more relevant for $Q^{(i)}$ than others. Given such a dataset, cross-encoders are typically trained to minimize the cross-entropy loss

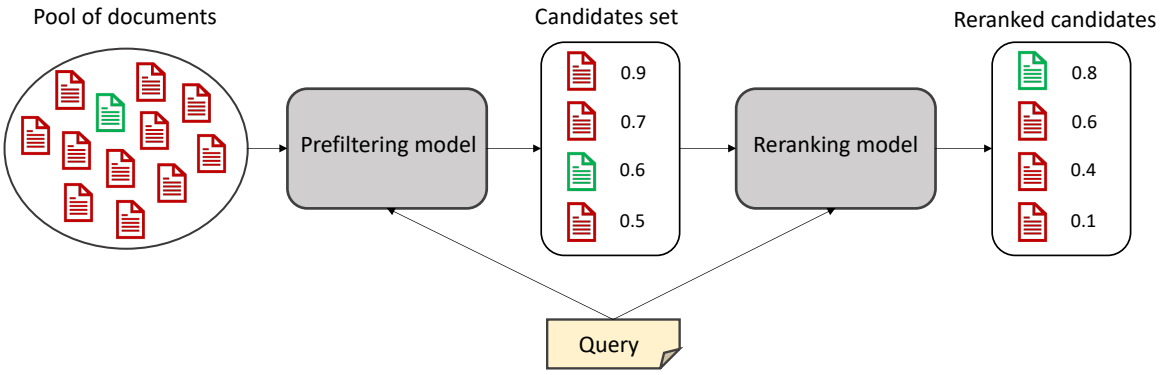


Figure 2.3: An illustration of a two-stage pipeline model containing a prefiltering model in the first stage and the reranking model in the second stage. Icons of documents in red represent irrelevant documents for a given query, while the green icon represents the relevant document. Both models in the pipeline output a list of documents sorted by the relevance scores obtained by each model. However, the pools of the documents each model operates over differ. While the prefiltering model operates over a complete pool of articles, the reranking model produces only those articles produced in the output of the prefiltering model, i.e., a subset of the complete pool of articles.

over a set of pairs of queries and relevant or irrelevant documents. Bi-encoders, on the other hand, are typically trained to minimize some form of contrastive loss. Contrastive losses are designed to penalize the model if the embedding of a query is more similar to the embeddings of the irrelevant documents than those of the relevant ones. An example of a contrastive loss is the triplet loss [31], often used in various text ranking setups. In the triplet loss, the loss is calculated across a set of triplets of the form $(Q^{(i)}, D_{+}^{(i)}, D_{-}^{(i)})$, where $D_{+}^{(i)}$ and $D_{-}^{(i)}$ are the relevant (positive example) and the irrelevant document (negative example), respectively, for the query $Q^{(i)}$. If the function of choice for calculating the similarity between the two embeddings is L2 distance, the loss for a given triplet is calculated as:

$$\mathscr{L}_t = \max(0, \|\mathbf{e}_{Q^{(i)}} - \mathbf{e}_{D_{+}^{(i)}}\| - \|\mathbf{e}_{Q^{(i)}} - \mathbf{e}_{D_{-}^{(i)}}\| + m) \tag{2.3}$$

where $\|\cdot\|$ is the L2 norm, and m is a margin between the two norms that defines how distant the irrelevant documents should be from the query. Figure 2.2 illustrates the process of training an encoder using triplet loss. To make the model learn better embeddings of queries and texts, it is important to choose informative negative items in the triplets, i.e., those similar to the positive items but still not relevant to the query.

While bi-encoders present a faster alternative to cross-encoders, the complexity of custom relevance score functions in cross-encoders leads to a better performance in providing more relevant results in the ranking lists compared to bi-encoders. However, as using cross-encoders for producing a ranking list over a whole pool of documents is timely, they are typically paired with other ranking models that perform the initial ranking over the whole pool of documents

and produce a smaller pool of documents for the cross-encoder to rank. Such systems that consist of multiple ranking models in different stages are called *pipelines*. Typically, multi-stage pipelines consist of the *prefiltering models*, i.e., fast but not very precise ranking models in the initial stages, and *reranking models*, i.e., slower but more precise ranking models in the final stages. For example, a two-stage pipeline might consist of a fast bi-encoder in the first stage and a slower cross-encoder in the second stage. Figure 2.3 shows an example of such a pipeline. Combining lexical with neural models in different pipeline stages is also possible. For example, instead of a bi-encoder in the first pipeline stage, BM25 could be used as a fast prefiltering model and paired with a cross-encoder in the second pipeline stage.

2.2.2 From Word2Vec to Transformers

After introducing the categorization of neural ranking models, we will describe the development of the most influential techniques and models used as building blocks of both bi- and cross-encoder models.

After the advent of DL in the 2000s, a major breakthrough in NLP was the introduction of word2vec [32], an algorithm that produced word embeddings that were close to each other in a vector space if the words appeared in a similar context in the training corpus. The algorithm was based on the distributional hypothesis [33], which posits that "*a word is characterized by the company it keeps*". This idea was formalized in an algorithm in which a neural model was trained to produce word embeddings such that the embeddings of the words cooccurring together have a larger dot product than the words that do not appear near each other in a corpus. The so-produced embeddings can then be used as features in various machine learning algorithms. When it comes to the neural ranking models, word2vec embeddings of terms appearing in the queries and documents can be used as input to a cross-encoder ranking model that outputs a relevance score for a given pair. Embeddings of words in a text sequence can also be summed or averaged to obtain an embedding of a whole sequence. Summing or averaging embeddings of terms in queries and documents can then be used to produce embeddings as input to a bi-encoder, which would then output the relevance scores for a query and a document based on the similarity between the embeddings. Although a simple operation, averaging embeddings of words in a sequence to produce an embedding of a whole sequence can result in a very strong baseline that is less complex than the state-of-the-art encoders.

Algorithms like word2vec produce a single embedding for each word in a vocabulary. This means that if a word has multiple meanings, the embedding representing that word will be the same regardless of its meaning or the context in which it is used. For this reason, such embeddings are typically called *static*. On the other hand, embeddings that consider the context in which a word is used and change the word's embedding based on that are called *contextualized* embeddings. While static embeddings were a breakthrough and improved performance

in many NLP tasks, contextualized embeddings have further advanced the performance of NLP models. An example of a model that produces contextualized embeddings is Long-short Term Memory (LSTM) network [34], a type of recurrent neural network (RNN) that processes words in a text sequentially and updates its hidden state based on the context in which a word appears. Updating the state means that the hidden state of an LSTM cell can be used as a contextualized embedding of the words in the sequence. In the same way that static word embeddings can be used as input to cross-encoders, contextualized embeddings could also be used. On the other hand, updating a hidden state in LSTM based on the previously processed words makes it suitable for producing contextualized word embeddings and a contextualized embedding of a complete text sequence. Such an embedding that would represent a complete text sequence could be used in a bi-encoder model, i.e., LSTMs could be used as bi-encoders that produce embeddings for given text sequences. As LSTMs process a text sequence in a single direction, [34] proposed a bidirectional LSTM variant (Bi-LSTM), which processes given text from both directions and combines the final hidden state from each direction into a sequence embedding.

Recurrent neural networks, such as LSTM, have been shown to suffer from a drop in performance when processing longer text inputs. A potential reason for this drop in performance lies in a fixed-length vector in RNN's hidden state that compresses all information from the sequence in a single vector. In addition, when updating the hidden state, RNN focuses only on the information from the currently processed word to decide how much to change the values in the hidden state. To remedy these limitations, *attention* mechanism [35] has been proposed to allow the model to select previously processed words it should focus on when updating the hidden state. Let s represent a text sequence consisting of $[w_1, ..., w_n]$ tokens and let $[\mathbf{e_{w_1}}, ..., \mathbf{e_{w_n}}]$ represent the contextualized embeddings of the s's tokens encoded with an RNN model. Using the additive attention scoring function (as defined in [35]), attention scores a_i for tokens $[w_i, ..., w_n]$ when calculating the final embedding of a token w_j are calculated as:

$$a_i = \mathbf{v} \cdot \tanh(\mathbf{W} \cdot [\mathbf{e}_{w_j}; \mathbf{e}_{w_i}]) \tag{2.4}$$

where $[\mathbf{x}; \mathbf{z}]$ represents a concatenation of the vectors $\mathbf{x}$ and $\mathbf{z}$, and parameters $\mathbf{v}$ and $\mathbf{W}$ are learnable parameters of the model. Normalizing the a_i scores, applying them to their corresponding hidden states of the RNN, and summing them together leads to a final embedding of a token w_j produced using an attention mechanism. The attention mechanism enabled RNN encoders to produce more informative embeddings of longer sequences by enabling the model to efficiently extract information from previously processed words, which would not be possible in the basic RNN model. It also added another layer of customization in the encoders through different definitions for calculating the weights for each word in sequence, potentially improving the model's performance if properly selected.

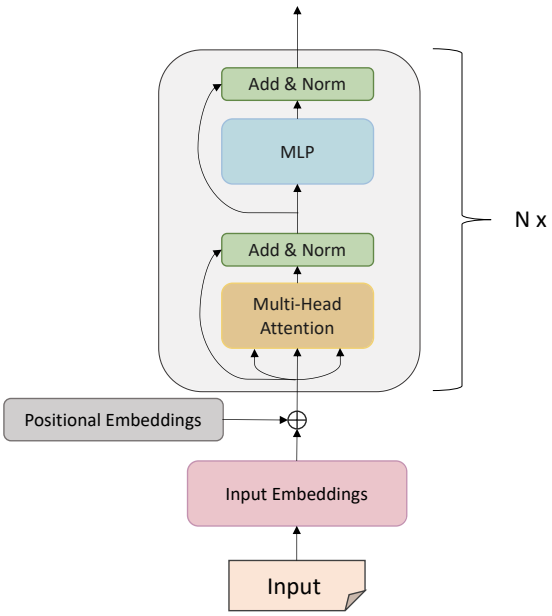


Figure 2.4: An illustration of a transformer encoder module described in [36]. The input text is first tokenized and encoded with the tokens' embeddings, which are combined with positional embeddings and, as such, passed to the layer's first submodule, i.e., a multi-head attention layer. The output of the multi-head attention layer is summed with its input (residual connection) and passed through a layer normalization module to produce the output of the first submodule. This output is passed to the multi-layer perceptron module, which is again followed by a residual connection and another layer normalization module. A transformer encoder consists of N such layers.

Successful integration of the attention mechanism with the RNNs led to further research into the possibility of using the attention mechanism to detect relevant information in text sequences. Along with exploring the attention mechanism's capabilities, research in NLP was also focused on trying to replace the sequential nature of RNNs with an architecture that could support the parallel processing of words in a sequence and, through it, reduce the time complexity of RNNs. This line of work culminated with the *transformer* architecture [36], which introduced the attention mechanism as a central building block. Contrary to RNNs, transformer models process words from the input sequence in parallel while using the attention mechanism by default to construct contextualized representations in each layer. A transformer encoder model consists of N layers, all parameterized with a number of modules relying on the attention mechanism, so-called *attention heads*. Through the many attention calculations that produce contextualized representations of the input text, combined with the positional embeddings indicating the position of a token in a sequence, transformer encoders provide rich semantic representations that led to state-of-the-art results in many NLP tasks. An illustration of a transformer encoder is given in Figure 2.4.

Transformers have been successfully adapted to both bi-encoder and cross-encoder designs regarding text ranking setups. In the bi-encoder design, the input sequence is typically prepended with a special token "`[CLS]`", and the token embedding from the last layer of the

transformer encoder is taken as the sequence embedding. When using transformer models in the cross-encoder design, it is convenient to use the attention mechanism already present in the transformer's architecture to combine the information from the query's and the document's tokens. This is done by concatenating a query and a document into a single text sequence and then passing this new sequence as input to the transformer model. As the attention mechanism in the attention heads is applied across all the words in a given input sequence, the query's and document's terms interaction happens in all of the transformer's layers using the operations already implemented in them. A relevance score in such case is typically calculated by again using a "`[CLS]`" token final embedding and passing it through a linear projection layer that produces a single score, which is then optimized during training.

2.3 Evaluation Metrics

Evaluating a text ranking model involves verifying whether the query's relevant documents are ranked higher than the irrelevant ones in the ranked list of documents produced at the model's output. Such evaluation can be quantified using different metrics, which we discuss below. These metrics apply to other ranking scenarios that do not necessarily involve document ranking (e.g., images). However, here we describe them in the context of text ranking.

Precision, recall, and F1 at k. Precision at k ($P@k$) and recall at k ($R@k$) are calculated over the top k documents in a ranked list of documents, as follows:

$$P@k = \frac{TP@k}{k} \qquad R@k = \frac{TP@k}{TP@k + FN@k} \tag{2.5}$$

where $TP@k$ (i.e., *true positives* at k) represents the number of relevant documents found in the top k documents produced at the model's output and $FN@k$ (i.e., *false negatives* at k) is the number of relevant documents that are not included in the top k documents. In general, precision at k tells the percentage of relevant documents in the top k recommended documents, while recall at k captures the percentage of overall relevant documents included in the top k documents. The F1 at k, denoted $F1@k$, is calculated as the harmonic mean of the corresponding $P@k$ and $R@k$ scores and is used to balance the precision and recall values. All three metrics take values between 0 and 1, with higher values indicating better performance.

R-precision Not all queries in a dataset typically contain the same number of relevant documents. To account for this difference, R-precision is used and calculated as follows:

$$R\text{-}prec = \frac{\sum_{q=1}^{|Q|} R@k_i}{|Q|} \tag{2.6}$$

where k_i represents the number of relevant documents for a query Q_i. As the metric is an average over $R@k$ values, it also takes values from 0 to 1, with higher values meaning better performance.

Mean reciprocal rank. Although precision, recall, and $F1@k$ provide a useful measure of the model's performance, these metrics are rank-insensitive, i.e., treat all relevant documents equally regardless of whether they are ranked first or k-th. On the other hand, reciprocal rank is a statistical measure commonly used in information retrieval that calculates the rank of the first relevant document in the ranked list of documents. When multiple queries are used in the evaluation, mean reciprocal rank (MRR) is calculated as the average of reciprocal ranks across all queries, as follows:

$$MRR = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{rank_i} \tag{2.7}$$

where Q is a set of queries and $rank_i$ is the rank of the first relevant document in the list of documents for the i-th query. Similar to recall at k, MRR can take values from 0 to 1, with higher values indicating better performance. Large MRR values mean that relevant documents are often positioned at the top of the list.

Mean average precision. Different than MRR, which focuses only on the first relevant document in a ranked list produced by a model, average precision is calculated by taking into account all the relevant documents in the ranked list:

$$AvgP(q) = \frac{\sum_{k=1}^{n}(P@k \times rel(k))}{Rel_q} \tag{2.8}$$

where n is the size of the ranked list, precision at k, denoted as $P@k$ is calculated as (2.5), $rel(k)$ stands for an indicator function that is equal to 1 if the document at rank k is relevant to the query and 0 otherwise and Rel_q represents the total number of relevant documents for the input query q. Mean average precision (MAP) is calculated as the mean value of the average precisions over a set of queries:

$$MAP = \frac{\sum_{q=1}^{|Q|}AvgP(q)}{|Q|} \tag{2.9}$$

The range of values for MAP is between 0 and 1, with a higher score indicating better performance. As MAP considers the positions of all the relevant documents in the ranked list, it is more suitable than MRR if a user wants to evaluate the performance across all the relevant documents instead of only the most highly ranked one.

Normalized discounted cumulative gain. In a text ranking scenario, often not all the relevant documents are equally relevant for a given query. In such cases, when multiple documents are appropriate for recommending, but the relevance for a given query varies among them, previously described metrics would not be appropriate to use, as they do not account for the difference in relevance. Instead, to compare the ordering in the ranked lists with the ground truth relevance scores, discounted cumulative gain [37] is used. DCG is a metric that estimates the gain of the ranked documents according to their position the ranked list, where the documents are sorted by their relevance scores. DCG is calculated as follows:

$$DCG_p = \sum_{i=1}^{p} \frac{rel_i}{\log_2(i+1)} \tag{2.10}$$

where p represents the rank position and rel_i represents the graded relevance of a document i (when relevance is binary, it is either zero or one). Scaling the relevance scores inversely with the rank's logarithm penalizes relevant documents ranked lower in the list. Given that queries differ per the number of relevant documents for each, *DCG* for each query is typically normalized with the ideal *DCG*. The ideal *DCG* (*IDCG*) is defined as follows:

$$IDCG_p = \sum_{i=1}^{|REL_p|} \frac{rel_i}{\log_2(i+1)} \tag{2.11}$$

where REL_p represents the list of relevant documents ordered by their relevance up to the position p. Normalized discounted cumulative gain (*NDCG*) for a query q at position p is then computed as:

$$NDCG_p = \frac{DCG_p}{IDCG_p} \tag{2.12}$$

Averaging values obtained by (2.12) across all the queries gives the final metric value for a set of queries. As *NDCG* considers both the variability in the number of relevant documents per query and the graded relevance scores of the relevant documents, it is most appropriate in such text ranking setups.

Chapter 3

Citation Recommendation

Having introduced the task of text ranking, in this chapter, we turn to the task of citation recommendation (CR), which is, in most cases, framed as a text ranking task. We start with a formal definition of global and local CR task variants. Then, we introduce the datasets used in research on CR. We then give an overview of different models previously used in global and local CR.

3.1 Task

Like other text ranking tasks, CR takes a query at the input and outputs a list of texts at the output, ordered by their relevance score for a given query. In the case of CR, texts ordered at the output are scientific articles, and the relevance scores represent how appropriate these articles are for citing in a given query.

Depending on how the query in CR is defined, two approaches exist: global and local. In global CR (GCR), a query represents an entire article, i.e., a manuscript in writing. In an ideal scenario, a query would be constructed using a draft of a project idea since this is typically a situation where scientists are looking for work relevant to their research topic. However, as such datasets are difficult to construct, proxy queries in which manuscripts are represented using an article's title and abstract are typically used. The article used to construct a query, i.e., the one for which recommendations are sought, is called a *citing* article. The task in GCR is to produce a list of articles as recommendations for citing in the article represented with a given query. The goal in training GCR models is to rank highly the articles referenced in the citing article (i.e., *cited* articles).

In contrast, in local CR (LCR), a query is composed of a text snippet extracted from a citing article's full text, and the task is to output recommendations for citing in a given snippet. This text snippet is called *citation context* and can span several characters or tokens surrounding a *citation marker*, i.e., a reference to a cited article. As an example, consider the following text

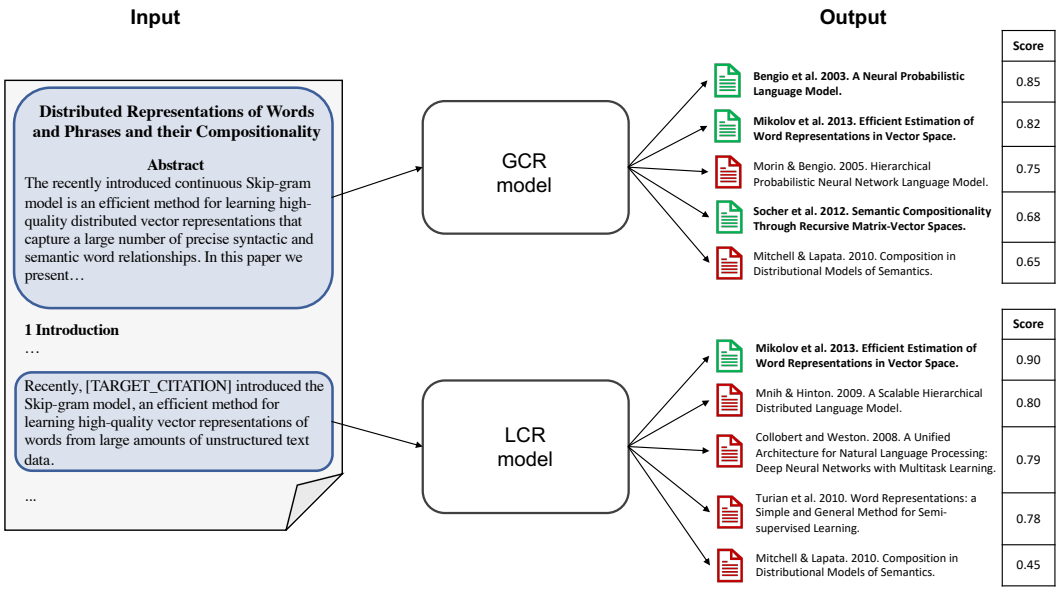


Figure 3.1: An illustration of the difference between GCR and LCR tasks. While the input to GCR is typically an article's title and abstract, the input to LCR in most cases consists of only the citation context, i.e., a text snippet from the article's body, in which a citation marker is masked with a placeholder (in this case, "`[TARGET_CITATION]`"). Models for both tasks produce a list of articles sorted by the relevance scores obtained by the model, where the goal during the training of these models is to rank the articles cited in the query (in green) higher than others (in red). The article titles printed in **bold** represent those that were indeed cited in the given article or context.

snippet from an article featuring a citation:[1]

> *We obtain the word embedding by training an unsupervised word2vec* `[CITATION]` *model on the training and validation splits and then use the word embedding to initialize* W_e.

A successful LCR model should recommend the correct citation for the "`[CITATION]`" placeholder in the given example. In this case, it is the article "*Distributed representations of words and phrases and their compositionality*" by Mikolov et al. (2013). Figure 3.1 illustrates these two approaches.

The difference in query construction leads to a difference in the applicability of the recommendations obtained by each model. As GCR's query represents a general overview of an article, the recommendations produced at the output can be diverse, with each focusing on some aspects of the query article's topic. On the other hand, citation contexts in LCR queries focus on the specific aspects of the articles cited in the contexts, making the LCR models' recommendations more specific than those of GCR.

In both GCR and LCR task variants, the datasets for training and evaluating the models that solve these tasks are obtained from the publicly available databases of scientific articles. These

[1]The excerpt is taken from the article "Hierarchical Attention Networks for Document Classification" by Yang et al. (2016)

datasets contain not only the content of the articles (title, abstract, full-text if available) but also metadata information such as author names, venues, and, most importantly, citation links between the articles. Additional parsing of the articles' full texts can produce citation contexts and links between the citation contexts and the articles cited in them, which can be used to train LCR models.

Although CR models focus primarily on the text in the queries to provide recommendations, they also use the metadata available in the datasets to model the recommended articles' authors, venues, citation counts, etc. Metadata information enables modeling authors' preferences regarding reasons for citing beyond the text. Although the use of metadata information could lead the models to being biased toward specific variables, such as specific authors, institutions, or publication venues, it is necessary as CR models are evaluated on datasets that already contain such biases [38]. However, relying too much on the metadata might cause the models to miss out on semantically more relevant articles. Debiasing and diversifying recommendations in CR models is thus an ongoing research [39, 40].

3.2 Datasets

In this section, we describe the datasets most commonly used in CR tasks. A number of various datasets composed of scientific articles have been introduced and used in CR tasks. Most of the datasets are for the English language, although a few exist for languages other than English, e.g., Chinese [41]. Typically, datasets used for CR consist of two parts: a collection of scientific articles that include both text and metadata and information about how the articles are cited in relation to one another. The citation information across the articles effectively creates a *citation graph*.

Datasets for CR differ with respect to the scientific fields they cover and the granularity of the information available for each article in the dataset. Some datasets contain full article texts, while others only contain titles and abstracts of articles, typically due to limited open access to the published articles. Furthermore, those datasets that contain full article texts further differ in the amount of citation information available for each article. While some datasets only contain lists of articles cited in each article, others provide citation contexts for each citation in an article, where citation context sizes might also vary across the datasets. These differences make some datasets suitable for GCR only, while others are appropriate for both GCR and LCR.

GCR and LCR datasets are constructed using parsers that process the citing article and output a list of all the cited articles (used for GCR), together with a list of citation contexts extracted from the text if the citing article's full text is available (used for LCR). When extracting citation contexts, the parsers typically generate a single citation context for each citation marker in the text. This means that although a number of articles can be cited in the same sentence (e.g.,

listed one after another), the citation contexts corresponding to each citation marker will be different, as parsers typically consider a number of characters or tokens surrounding the marker. To alleviate this problem, researchers mostly use some string matching techniques that attempt to unify multiple citation contexts into a single one, for cases when several articles are cited together. However, most of the work on LCR disregards this dataset deficiency and assumes only one article is appropriate for citing for a given citation context corresponding to a citation marker.

When evaluating CR models, it is common to use time-based data splits, where the training, validation, and test sets are made up of articles published in different years. This is considered more realistic than using random data splits since a CR system can only recommend articles published before the citing article in real-world situations.

ACL-ARC.[2] This is a dataset of articles from conferences and journals in the field of computational linguistics and natural language processing, published by the *Association for Computation Linguistics* (ACL).[3] ACL-ARC (*Association for Computation Linguistics – Anthology Reference Corpus*) was first released in 2008 by [42] and has been updated a number of times since then. All dataset versions contain full article texts and metadata such as author names, publication venues, and citation links between the articles. The version from 2016 contains citation contexts from articles' full texts extracted using ParsCit [43], where each citation context spans 600 characters before and after a citation marker. Since the dataset contains extracted citation contexts, it suits both LCR and GCR tasks.

RefSeer.[4] A dataset compiled using the collection of articles from the CiteSeer digital library [44]. The dataset contains information about the title, abstract, author, and venue information for each article included in the dataset. In total, there are over 800K articles in it, covering various scientific fields, with the majority being from computer science. The dataset also contains over 4.5M citation contexts between pairs of articles, each containing 200 characters before and after a citation marker.

DBLP.[5] DBLP dataset [45] is a dataset of articles from the field of computer science. Like in RefSeer, each article in DBLP is represented with its title, abstract, author, and venue information, without citation contexts. In total, the latest version of DBLP contains over 50K articles.

[2] `https://acl-arc.comp.nus.edu.sg/`
[3] `https://www.aclweb.org/`
[4] `https://citeseerx.ist.psu.edu`
[5] `https://dblp.dagstuhl.de/`

PubMed.[6] The PubMed dataset covers articles from the biomedicine field. As RefSeer and DBLP, PubMed also provides information about the title, abstract, author, and venue for each article in the dataset. While multiple versions of the dataset exist, most of the work on CR uses a version of the dataset that contains over 45K articles with an average of 17 citations per article [16].

OpenCorpus.[7] The OpenCorpus dataset [16] consists of around 7M articles that are predominantly from computer science and neuroscience domains. Each article in the dataset contains information about the article's title, abstract, authors, publication year, venue, keyphrases, and citations across the articles. The dataset does not contain extracted citation contexts, so it is suitable only for the GCR task.

S2ORC.[8] The most recently introduced dataset of scientific articles and the accompanying metadata is S2ORC (The Semantic Scholar Open Research Corpus) [46]. It is the largest of the datasets covered in this section, with around 81.1M articles in its collection covering various scientific fields. Of these articles, 8.1M articles are represented with their parsed full texts, which provide citation contexts linked to the articles cited in them and other parsed article content such as tables and figures. Since it has large coverage and detailed information available in it, S2ORC has been used in many tasks in SDP other than CR, such as scientific fact verification [47] and scientific question answering [48].

3.3 Global Citation Recommendation Models

Most of the early work on GCR models used keyword search across a pool of articles and combined it with the processing of metadata, such as citation graphs and citation counts [49, 50, 51]. The queries in those models comprised an article's full text or a few keywords representing the article's topic.

A more realistic approach was described in [13], who used the title and abstract of a citing article as a proxy for a research idea while treating all the articles cited in the citing article as relevant recommendations. Although more realistic than using the complete article's text as input, [13] is still far from an ideal approach, as abstracts typically mention the results of the conducted experiments, therefore deviating from a description of a research idea that lacks the results. Another deficiency of using titles and abstracts for query construction is that these, by definition, present a summary of work described in the article's full text. Therefore, one cannot expect a GCR model to provide all the references cited in a given article if only a summary is

[6] https://www.ncbi.nlm.nih.gov/pubmed
[7] https://api.semanticscholar.org/corpus/
[8] https://github.com/allenai/s2orc/

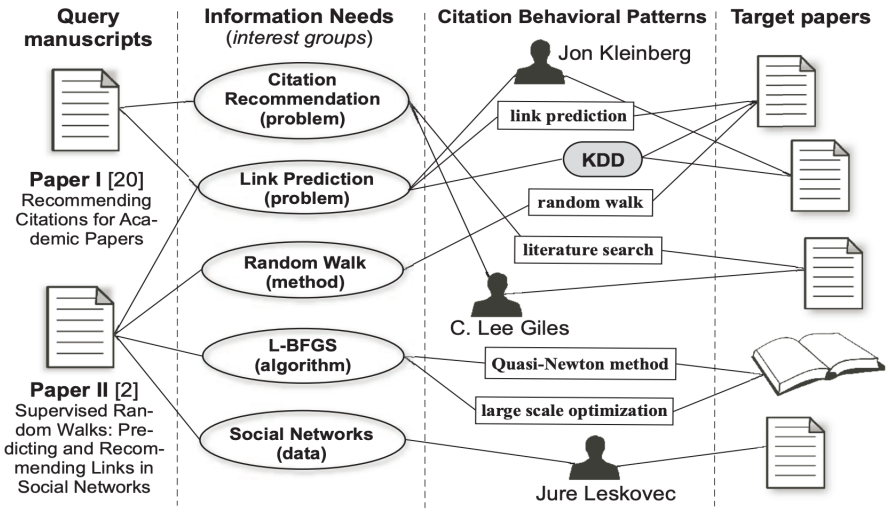


Figure 3.2: An overview of the steps in the recommendation process when using ClusCite. First, each query manuscript is matched with an interest group, while in the second step, articles from the selected interest groups are scored for relevance for a given query based on the metadata features. Figure taken from [53].

given as input. The model built by [13] was based on textual and metadata features. The textual features they used were TF-IDF scores of terms appearing in the title and abstract of the query, terms appearing in the texts of the pool articles, and topical embeddings of articles. Among the metadata features, they used citation counts, the article's PageRank score [52], the difference in years between the citing and the pool article, and others. To improve the representation of the pool articles, they also collected all the citation contexts in which each article from the pool was cited and vectorized these contexts using TF-IDF. The model was then trained to learn the weights for each feature in an iterative process to produce a final recommendation score as a weighted sum of the features. Different models for learning the feature weights were evaluated on the ACL-ARC dataset, and the best-performing one was SVM, with a MAP score of 28.7 on the test set.

The potential of CR models on single-domain datasets such as ACL-ARC was successfully demonstrated in [13]. As an extension to multiple domains, [53] proposed a CR model that considers the information need or the *intent* behind a citation. The system they proposed, called ClusCite, uses *interest groups*, i.e., soft clusters of articles that group articles based on the article's content, authors, and venue. Recommendation scores in ClusCite are then calculated in two steps. Given an input manuscript, first, a score representing the relevance of each interest group to the given input is calculated. The second step calculates the relative importance of articles within each interest group. An example of processing two query manuscripts with ClusCite is given in Figure 3.2. ClusCite also uses graph-extracted metadata features representing articles, venues, and interest groups. The model is evaluated on two datasets, DBLP and PubMed, where it obtained an MRR score of over 0.5 in both datasets. However, the downside

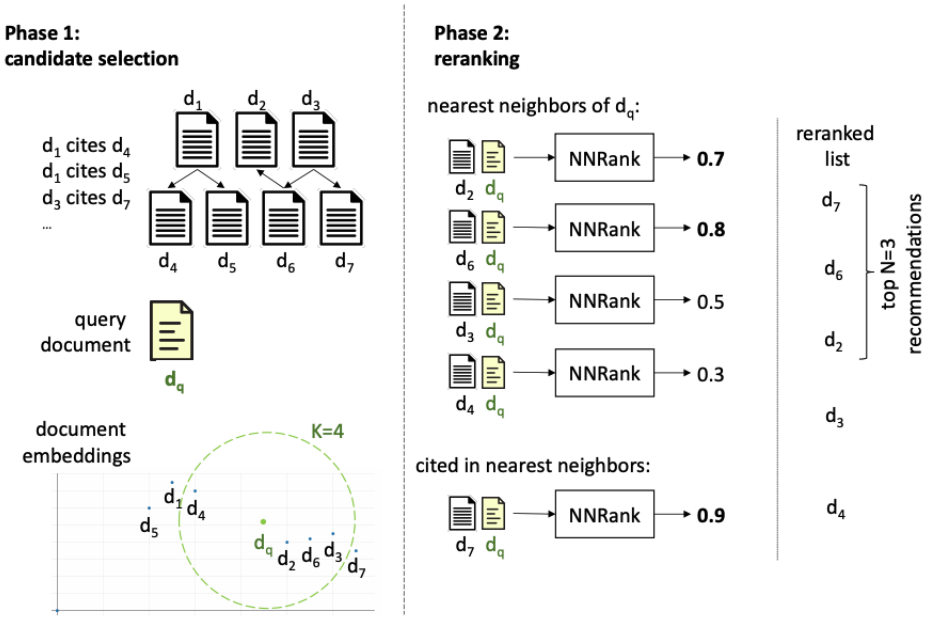


Figure 3.3: An overview of a GCR model from [16]. In the NNSelect module, embeddings for all seven documents d are in a shared document embedding space. The K nearest neighbors for a query are detected and passed as input to the NNRank module. NNRank reranks these K articles using a reranker model that produces a recommendation score for each article. Figure taken from [16].

of the approach is its large time complexity, as the time it takes to train the model increases as the number of links in the dataset increases.

Advancements in DL have led to the developing of new models for GCR based on DL. In DL, the similarity between items is usually determined by creating a vector representation of each item (an *embedding*) and then measuring the similarity between the vectors using a method such as cosine similarity or L2 distance. One such model in which articles are represented with embeddings in a shared embedding space was presented in [16]. Their model creates article embeddings focusing mostly on the article's text, which makes it robust to the potential lack of metadata for some articles. The proposed model comprises candidate selection (NNSelect) and reranker module (NNRank). An overview of both modules is shown in Figure 3.3. In the NNSelect module, all articles from the dataset are transformed into the same embedding space using a neural network trained to create a vector representation of each article using their corresponding titles and abstracts. The module is trained using the triplet loss (cf. Section 2.2) with negative examples in the triplets selected in three ways: (1) randomly, (2) from a set of the nearest neighbors in the embedding space that were not cited in the query, or (3) from the set of articles cited in the citations of the query article but not cited in the query. The NNRank module reranks the top k closest neighbors of the query article obtained from the NNSelect module. The recommendation scores are produced by a neural network that is given cosine similarities between different article's fields (such as title, abstract, authors), cosine similarities between the documents obtained from the embeddings used in NNSelect, and the number of

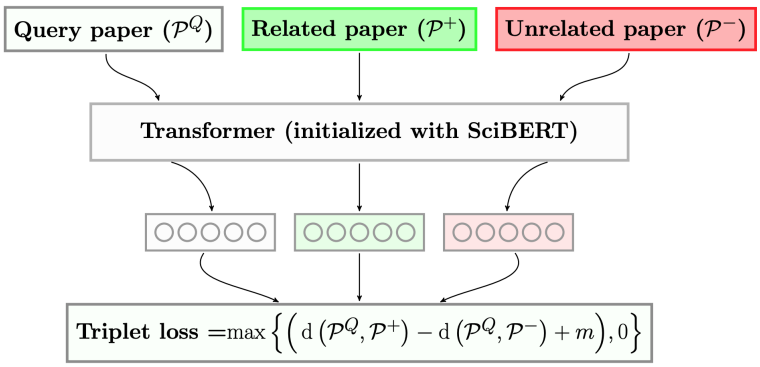


Figure 3.4: An overview of SPECTER's training process [54]. The three articles (query, related, and unrelated) are passed as input to the same transformer model, which outputs an embedding for each article. The loss is calculated as triplet loss using L2 distance from both the related and unrelated articles to the query. Figure taken from [54].

citations of an article that is being ranked. The model was evaluated with F1@20 and MRR on DBLP, PubMed, and OpenCorpus datasets, where it outperformed both ClusCite and BM25 in each dataset. This model is an example of a two-stage pipeline system (cf. Section 2.2) where models in both stages are neural-based.

Following the successful incorporation of article vector representations and triplet loss-based training of article encoders in [16], [54] introduced SPECTER, a transformer-based model that outputs an embedding for a given article, that can be used in many different SDP tasks. SPECTER builds on SciBERT [55] and fine-tunes it on triplets of citing, cited, and non-cited articles. The fine-tuning process maximizes the difference in the L2 distance between a citing and a cited article and that between a citing and a non-cited article. This leads to the model producing similar vector representations for articles that cite each other, therefore, those that are similar to each other in some aspect. Figure 3.4 shows an overview of the model's training process. Although [54] did not evaluate SPECTER on any of the CR datasets introduced in this chapter, they compared its performance with the model from [16] on the task of "click-recommendation" using the data from a scientific search engine. They showed that transformer-based embeddings from SPECTER outperform the model of [16] in *NDCG* and *P*@1.

Following the success of SPECTER, [56] proposed a model named SciNCL, which is of the same architecture but with better-produced representations as measured on scientific article representation benchmarks [57, 58]. They trained the model of the same size and with the same triplet loss but chose the negative examples that were informative enough for the model. First, they built graph embeddings using the citation graph structure of the S2ORC dataset, in which two articles end up with a similar embedding if they are not far apart in the citation graph. Using these embeddings, they selected the negative examples for the triplets in training SciNCL by choosing the articles close to the query in the graph embedding space but not cited in it. They demonstrated that this careful selection boosts the model's performance in various

tasks requiring encoding of an article, ranging from classification to recommendation.

3.4 Local Citation Recommendation Models

The first to introduce the task of LCR was [14]. They constructed a dataset for LCR by extracting a subset of over 400k articles from the CiteSeer digital library. Each article in the constructed dataset was represented with its title, abstract, authors, and all the citation contexts in which it was cited. These citation contexts were used as queries, but they were also used to enhance the representation of the pool articles. Specifically, when predicting the missing citation in a given citation context, the model calculated the similarity between the context and a pool article represented as a TF-IDF-weighted "bag-of-words" vector over the article's title, abstract, and the citation contexts in which it was cited. Although the proposed model showed potential as the first model for that task, it was not suitable for recommending articles that were previously not cited, as the article's representation relied on its citation contexts.

Soon after the first proposed model for LCR, the community moved to DL-based approaches. In [15], the authors proposed a neural probabilistic model for LCR that was trained to maximize the dot product between the word embeddings of the words that appeared in the same citation contexts, as well as between the context words and the cited article's embedding. At test time, the proposed model sorted the pool articles by the sums of the probabilities of the context words citing the pool articles, where the probabilities were obtained by passing the dot product of the embeddings through a sigmoid function. The model was evaluated on the RefSeer dataset and obtained an MRR score of 0.18 on the top 10 candidates. A major deficiency of the model is its inability to provide embeddings for the previously unseen articles, as it only stores embeddings for the articles seen at test time. In addition, as the model assumes independence between the words in a citation context, it is not able to model compositionality between them., i.e., to represent the meaning of a word sequence as a function of the meanings of the sequence parts, according to how these parts are combined in a sequence [59].

Following the DL-based approaches, [18] experimented with an encoder-decoder architecture that allowed the model to model compositionality between the input query's words. They presented a Neural Citation Network model (NCN) that follows an encoder-decoder architecture comprising two parts: an encoder that creates a representation of the input word sequence and a decoder that generates the final output using the representation created by the encoder. This type of architecture is often used in various sequence-to-sequence scenarios in NLP [60], where the model takes a word sequence at the input and outputs another (e.g., in machine translation or question-answering tasks). To adapt this architecture for the LCT task, [18] used a time delay neural network [61] to encode the citation context from the input and combined it with author embeddings to produce the representation that was passed to a decoder. The decoder

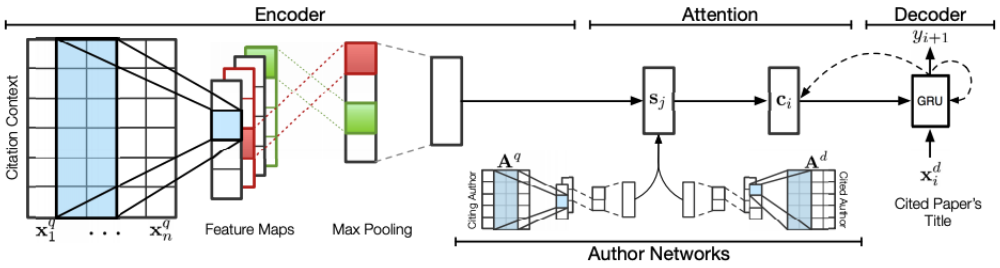


Figure 3.5: An overview of the NCN model proposed by [18]. Figure taken from [18].

was trained to produce the title of the article cited in a given context. Combining textual and metadata information in the encoder allows the model to relate one piece of information to the other and use it for prediction. For example, knowing the author of the citation context might make it easier for the model to predict which title should be cited in the context, as it might choose the type of work or the authors they usually cite. The proposed model was evaluated on the RefSeer dataset and obtained an MRR score of 0.27 on the top 10 recommendations.

The model of [18] improved the modeling of the input citation context over the previous work of [15]. However, it only used the title of the cited article in the process of training the model. Titles might not contain enough information for the model to decide whether an article should be cited in a given input context. Attempting to improve the representation of the cited article, [62] incorporated stacked denoising autoencoders [63] to encode either abstracts or full texts of the cited articles. Instead of a time delay neural network used in [18], they use a bi-directional LSTM to obtain a citation context embedding. Both Bi-LSTM and stacked denoising autoencoder used attention mechanisms to further select the information in the input sequence for producing final sequence embedding. Embeddings of citation context and cited article were concatenated and passed as input to a neural network that produces a recommendation score for a given pair. The proposed model was evaluated on ACL-ARC and DBLP datasets, outperforming NCN in both datasets.

In this chapter, we have formally described the tasks in CR (GCR and LCR) and introduced the existing datasets used to train CR models. We have also covered the related work on the topic of building both GCR and LCR models. In the next chapter, we turn to the first set of experiments in which we aim to improve the performance in the LCR task by leveraging information from the citing article's title and abstract to represent a citation context.

Chapter 4

Citation Context Enhancement for Local Citation Recommmendation

In this chapter, we focus on improving the performance of an LCR model by enhancing the citation context representation using additional textual information from outside the context, i.e., we address (**RQ1**), the first of the five research questions introduced in Chapter 1. We start with the motivation for introducing such context enhancement and then proceed to the model's description and present the evaluation results. The chapter builds on the model and results presented in our previous work [64].

Recall from the previous chapter that the datasets for LCR are constructed using parsers that process PDF documents of scientific articles and output citation contexts linked to the articles cited in them. The citation markers in these contexts are then masked with a placeholder, and the LCR model is trained to predict the masked article. An example of such citation context with a masked citation marker is:

> *"Other approaches include for example clustering-based algorithms, such as the one presented in* `CITATION`*, or techniques which rely on building a statistical language model to rank KPs"*

In this example, `CITATION` placeholder is inserted as a mask for the marker of the article that was cited in the context.[1]

In a typical LCR scenario, this citation context would be used as the only textual input to the model, which means that the model should rank the relevant articles based solely on the information from the context. However, looking at this example, it seems difficult even for humans to decide which article should be recommended for citing in a given context. One way to allow the context access to more information would be to increase the size of the citation

[1]Citation context is from *Evaluating anaphora and coreference resolution to improve automatic keyphrase extraction* of Basaldella et al. (2016). The article cited in the context is *Clustering to find exemplar terms for keyphrase extraction* of Liu et al. (2009).

context, i.e., to increase the number of characters before and after the citation marker. However, this is not always possible as already configured parsers produce the datasets. Even if it could be possible, it is still difficult to estimate how much the context should be expanded for each citation, as not all citations are described in the same number of words.

Another option that is better fitting for the adaptation with already created datasets is to combine the information from the citation context with the citing article's title and abstract, which we call citing article's *global information*. As citation contexts are extracted by parsing PDFs, the global information for each context can be easily found given the ID of the article from which the context was extracted. Enhancing context with global information ensures that all citation contexts are combined with the overall topic of the citing article, whereas simple expanding of the citation context might introduce noise in the context resulting from a description surrounding the context that is not relevant to the cited article. Thus, we propose a model that uses global information to supplement the citation context and describe it in the following section.

4.1 Model with Enhanced Context Representation

We propose a second-stage reranking model in a two-stage LCR pipeline system consisting of a BM25 model in the prefiltering stage. In the first stage of the pipeline, BM25 retrieves 1000 candidates from a pool of articles, which are then passed to the reranking model to provide final recommendation scores. Our proposed model produces a recommendation score for a pair of a citation context and a pool article based on five inputs: (1) citation context text, (2) citing article's title and abstract, (3) candidate pool article's title and abstract, (4) a list of candidate pool article's authors, and (5) the number of candidate pool article's citations in the last y years, together with the total number of citations since its publication. The first three inputs represent semantic information that can be found in the text of the citation context and the pool article, while the last two inputs represent metadata information. As authors often cite articles for reasons beyond text alone [38], combining textual information with metadata helps the model to simulate citation behavior in the existing datasets.

The model consists of text and bibliographic modules, depicted in Figures 4.1 and 4.2, respectively. The text module takes textual information as input, while the bibliographic module takes metadata as input. Each module produces its own recommendation score for a given pair of a citation context and a candidate article, and these scores are then combined into a final recommendation score via weighted sum. Weights in the sum are used to model the author's preferences when choosing an article to cite. While in some contexts, more popular articles are preferable and more weight should be put on the bibliographic recommendation score, in other cases, more semantically relevant articles should be preferred, with a larger text score weight.

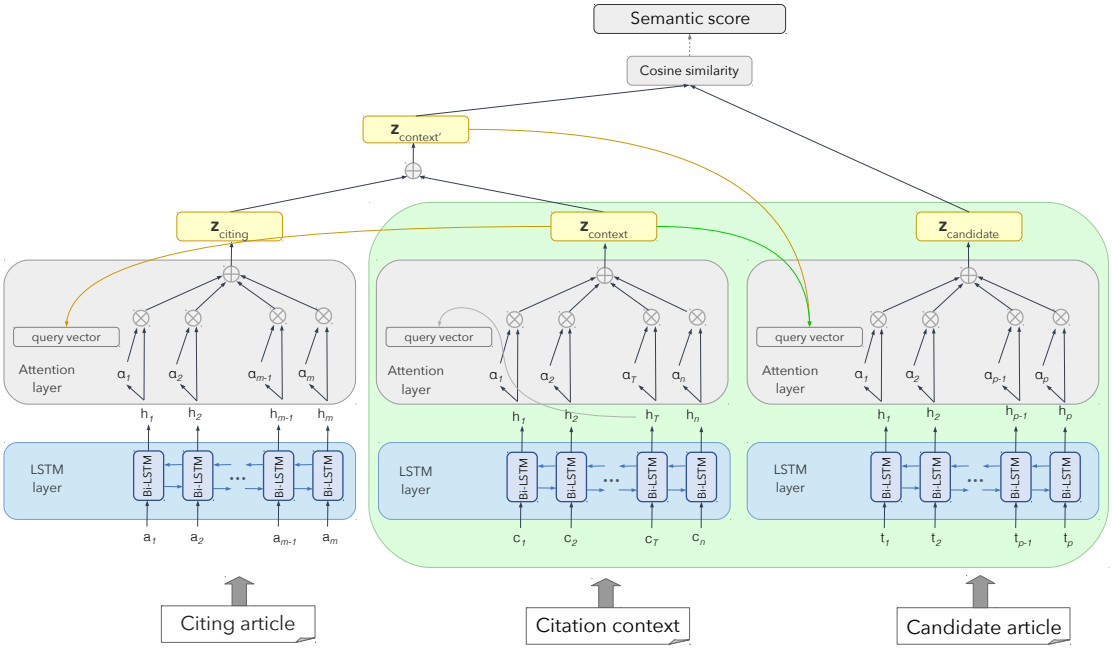


Figure 4.1: The text module of the proposed LCR model. All the text sequences are passed as input to the same layers. The difference in the processing of each input is in the definition of the attention query (arrows indicate which embedding is passed as a query in each input). The green oval corresponds to the model variant in which global information is not included in the citation context representation.

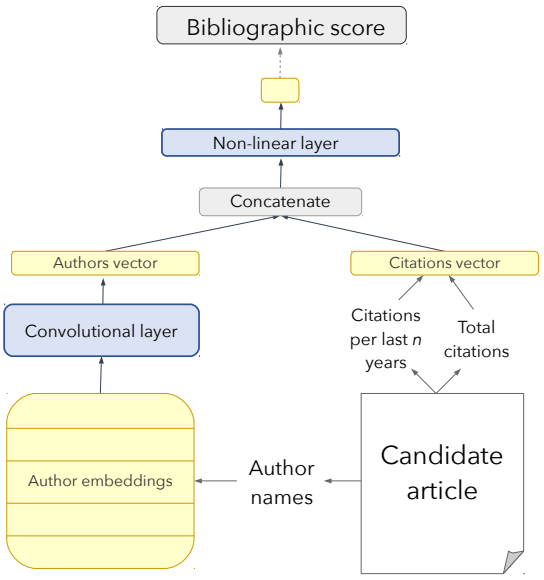


Figure 4.2: The architecture of the bibliographic module of the proposed LCR model. Author names are extracted from the candidate article and matched with their corresponding embeddings. These are then passed through a convolutional layer and concatenated with a citations vector, consisting of the number of candidate article's citations overall and throughout the last n years. The final score is produced by passing the concatenation through a non-linear layer.

Next, we describe both modules and how they are used to produce the final recommendation score in the model.

Text module. As the main neural network block for the text module, we use a Bi-LSTM cell [34] in combination with an attention layer [35]. Before passing each textual input to the model, we perform standard text preprocessing such as sentence segmentation and tokenization, but also include the masking of the citation marker in the citation context by replacing the marker with a special token "`TARGET_CITATION`".

Each text sequence is passed as a separate input to the module. Let s represent a textual sequence $[t_1, \ldots, t_n]$ consisting of n tokens. Each token t_i is first mapped to its embedding $\mathbf{x}_i$, resulting in a sequence of n embeddings $[\mathbf{x}_1, \ldots, \mathbf{x}_n]$. This sequence first passes through a Bi-LSTM layer that produces contextualized token embeddings $[\mathbf{h}_1, \ldots, \mathbf{h}_n]$ as concatenations of the forward and backward hidden states. Each embedding is then passed through an additive attention layer [35], which produces a final sequence embedding $\mathbf{z}_s$. Specifically, given an input query vector $\mathbf{q}$ and a hidden state vector $\mathbf{h}_i$, attention score for token t_i is calculated as:

$$a_i = \mathbf{v} \cdot \tanh(W \cdot [\mathbf{q}; \mathbf{h}_i]) \tag{4.1}$$

where $\mathbf{v}$ and W represent model parameters learned during the training. These scores are then normalized and applied to the corresponding token embeddings, which are summed to produce the final sequence embedding.

Inside the attention mechanism, we use a different query vector q for each of the three textual inputs (citation context, the citing article's title and abstract, and the candidate article's title and abstract). In general, query vectors in the attention mechanism are used to calculate the attention weights that determine how much information from each token's embedding will be included in the final embedding for a given query. With this in mind, when processing citation context, we use the hidden state of the citation's placeholder "`TARGET_CITATION`", ($\mathbf{h}_{TC}$), as $\mathbf{q}$ to allow the model to choose the information relevant to the citation at that specific place. When processing the citing article's title and abstract, we set $\mathbf{q}$ to the $\mathbf{z}_c$ where c is the citation context, i.e., $\mathbf{z}_c$ is the final embedding of the citation context obtained after the attention layer. The motivation for this is that, depending on the context, the model should focus on different parts of the title and abstract. Finally, when processing the candidate article's title and abstract, we use the sum of the citation context ($\mathbf{z}_c$) and the citing article embedding ($\mathbf{z}_{ca}$, where ca represents a sequence of citing article's title and abstract) as $\mathbf{q}$. This allows the model to create a different embedding for each candidate article, depending on the citation context. The interaction between the citation context (with its citing article) and the candidate article through the attention mechanism makes our model a cross-encoder by design.

For a citation context c and a candidate article p, the relevance score produced by the text

module $s_{\text{text}}(c,p)$ is calculated as the cosine similarity between the enhanced context embedding and candidate article's embedding.

Bibliographic module. The bibliographic module produces a bibliographic relevance score that indicates how relevant the pool article is to the citation context based on its authors and the number of previous citations, which models the popularity of an article. We represent each author with an embedding learned during the training of the model. Specifically, given a list of m authors of a pool article $a = (a_1, \ldots, a_m)$, we create a sequence of author embeddings $\mathbf{A}_e = (\mathbf{a}_1, ..., \mathbf{a}_m)$, which is passed as input to a convolutional layer with max-pooling and non-linear transformation at the output. The embedding produced at the output is concatenated with the number of citations in the last n years and a total number of citations of the pool article. The resulting vector is then passed through a non-linear layer that outputs a single score at the output $s_{\text{bib}}(p)$, i.e., a bibliographic score for the pool article p.

Final recommendation score. Once both text and bibliographic scores are calculated, we use a weighted sum of these scores to produce the final recommendation score for citing article p in the citation context c:

$$s_{\text{fin}}(c,p) = w_t \cdot s_{\text{text}}(c,p) + w_b \cdot s_{\text{bib}}(p) \tag{4.2}$$

where w_t and w_b are the weights that are obtained by passing the context embedding $\mathbf{z}_c$ through a non-linear layer with two values at the output. We refer to this model as TEXTBIBENH-WS – the full model that produces the final recommendation score using a weighted sum of text and bibliographic modules with an enhanced context representation.

4.2 Model Training and Evaluation

We trained the model using the standard triplet loss described in Section 2.2. However, instead of the L2 distance between the embeddings of the query and each example in the triplet, we used cosine similarity as a metric. This change of metric resulted in a slightly different formula than Eq. 2.3, since a high value in cosine similarity indicates larger similarity between the two embeddings, whereas high values in L2 distance used in Eq. 2.3 indicate a lower similarity. Specifically, we minimized the following loss:

$$\mathscr{L}_t = \max(0, \cos(c,p_-) - \cos(c,p_+) + m) \tag{4.3}$$

where $\cos(c,p_-)$ and $\cos(c,p_+)$ represent cosine similarities between the citation context and the article not cited and cited in the context, respectively. The negative examples for the triplets

Dataset	Train	Validation	Test	Articles
ACL-ARC	30,390	9,269	9,408	19,711
RefSeer	3,521,582	124,551	126,021	624,957

Table 4.1: Number of contexts and articles in each dataset's train, validation, and test splits, alongside the total number of articles in each dataset.

were sampled randomly from the whole pool of articles.

We evaluated our proposed model on ACL-ARC and RefSeer datasets. For both datasets, we used time-based splits, ensuring that the contexts in the train set are older than those in the validation and that validation ones are older than those in the test set. The datasets statistics are given in Table 4.1. Before processing a query-article pair with our model, we performed prefiltering of the pool articles using BM25, where BM25's parameters b and k_1 were optimized using grid search on the validation set. When training our model, we used all the citation contexts from the training set paired with the articles cited in them, regardless of whether BM25, a first-stage prefiltering model, would output these cited articles in its prefiltered candidate set. We did the same at test time, i.e., for the cases when BM25 did not include relevant articles in the candidate set, we inserted those manually and reranked the updated set of candidates using our model. Although this results in an unrealistic performance estimation, as it cannot be expected to insert articles manually in production, it aligns with previous work [18].

As mentioned in Section 3.2, these datasets differ in their size, the fields they cover, and the size of the citation contexts in each. Specifically, citation contexts in ACL-ARC contain 600 characters before and after a citation marker, while those in RefSeer contain 200. This makes them appropriate for analyzing the impact of including global information in the context representation on the performance of an LCR model given different citation context sizes. To account for the difference in the context sizes, we evaluated the models on ACL-ARC with its 600 characters in the context as in the default version (ACL-600) and with 200 characters in the context as in RefSeer (ACL-200).

We compared the performance of our model with BM25 and NCN [18]. Together with the full model (TEXTBIBENH-WS), which uses enhanced context representation, both textual and bibliographic modules, and a weighted score as the final recommendation score, we evaluated the ablated versions to analyze how specific design choices affect performance. Specifically, we compared the performance with the following variants: TEXTCON, a variant that uses only text score without context enhancement with the citing article's global information, TEXTENH, a variant that uses only the text score with enhanced context, TEXTBIBCON-WS and TEXTBIBCON-S, variants without the enhanced context representation that use both module scores with and without weighting them, and TEXTBIBENH-S, a variant the same as the full model, but without the weighting of the modules' scores.

	ACL-600		ACL-200		RefSeer	
Model	R@10	MRR	R@10	MRR	R@10	MRR
BM25	0.095	0.049	0.095	0.049	0.090	0.050
NCN	–	–	–	–	*0.291*	*0.267*
TEXTCON	0.568	0.306	0.537	0.291	0.340	0.166
TEXTENH	0.553	0.290	0.546	0.285	0.445	0.216
TEXTBIBCON-S	0.654	0.322	0.693	0.340	0.363	0.185
TEXTBIBCON-WS	0.689	**0.368****	0.647	0.335	0.406	0.206
TEXTBIBENH-S	0.662	0.315	**0.716**	0.341	0.437	0.230
TEXTBIBENH-WS	**0.699**	0.357	0.703*	**0.366***	**0.534***	**0.280***

Table 4.2: Results on the RefSeer and the two ACL-ARC dataset variants for baselines and variants of our proposed model. ** and * indicate a statistically significant difference between TEXTBIBENH-WS and TEXTBIBCON-WS for $p<0.05$ and $p<0.01$, respectively (two-sided t-test for MRR and two-proportion z-test for R@10).

The results of our evaluation, in terms of R@10 and MRR, are given in Table 4.2. The results show that all of our proposed model variants outperform BM25 and NCN in all cases of datasets and metrics. Furthermore, we obtain the best results with our full model (TEXTBIBENH-WS) in the case of RefSeer and in the case of ACL-200 in MRR, indicating that the inclusion of the global information and separate modeling of text and bibliographic scores are indeed helpful in case of citation contexts of smaller size. However, it is interesting to note that enhancing context representation with the global information did not help in the case of ACL-600, i.e., a dataset with a larger context size, where TEXTBIBCON-WS obtained better results in terms of MRR than TEXTBIBENH-WS. This suggests that contexts of a larger size might already contain enough information for the model to decide which articles should be cited in them. Even if the longer contexts do not entirely describe only the article masked with a placeholder, but the context's content overlaps with a description of another article and its citation, this information might be sufficient for the model to detect the domain and narrow down the search over a possible too large a pool.

Overall, in this chapter, we have provided an answer to the (**RQ1**), the first research question of the thesis, where we determined that global information can be effectively utilized in the LCR models and that it helps the model's performance if the citation contexts are smaller. In the next chapter, we analyze specific design choices in building LCR systems that might improve the performance of the LCR models.

Chapter 5

Design Choices in the Local Citation Recommendation Pipeline System

The previous chapter presented a model for LCR that can be used as a second-stage reranking model in an LCR pipeline system. The presented reranking model was trained on standard LCR datasets, using a triplet loss over triplets composed of citation contexts as queries, the cited articles as positive items, and randomly sampled non-cited articles as negative items. When training and evaluating the model, we did not account for the mistakes that could possibly be made by the pipeline's first-stage prefiltering model, BM25 in our case, and manually inserted cited articles in the set of prefiltered candidates if those were not there initially. However, we did try to improve the performance of the prefiltering model by optimizing b and k_1 parameters of BM25, which could reduce the possibility of not retrieving the relevant articles in the first pipeline stage. When training and evaluating a reranking model for LCR, this procedure is quite common in previous work [14, 18, 65, 66], where authors train two-stage pipeline systems focusing on reranking only and disregarding the errors potentially made by the retriever model. Such a training regime, which we call *standard*, might introduce a *dataset shift*, i.e., a difference in the data distribution at train and test time [67, 68, 69], which in turn could hurt the model's performance. A potentially better regime, to which we will refer as *strict*, is one in which the reranking model is trained with only the contexts for which the prefiltering model provided the correct recommendations in the candidate set, and the rest are discarded. Furthermore, some resort to using prefiltering models without optimizing their hyperparameters (e.g., parameters b and k_1 in BM25), eventually leading to a pessimistic performance estimation, as the model's capacity was not explored entirely.

Another issue with the model presented in the previous chapter is a possibly oversimplistic generation of the training set triplets. Research on encoders based on metric learning has shown that negative examples in the triplets need to be informative enough for the model to learn how to effectively project the given input into a shared embedding space and obtain the desired

relationships between the produced embeddings [56, 70, 71]. Regarding LCR encoder models, previous work has not used metric learning approaches, and the model from the previous chapter used random negative sampling, which tends to be suboptimal.

In [72], we analyzed how these previously overlooked design choices affect the performance of a two-stage pipeline system for LCR. This chapter presents these findings in more detail, focusing on the second (RQ2), third (RQ3), and fourth (RQ4) research question introduced in Chapter 1. We first look into how much the performance of a prefiltering model can be improved when optimizing its hyperparameters. Next, we compare the performance of a reranking model trained with a standard training regime with that trained with a strict regime, thus obtaining a more realistic performance estimation. Finally, we analyze how different negative sampling strategies for generating training triplets affect the performance of an LCR system. As our results demonstrate, all these typically ignored design choices can lead to an improvement in performance of LCR systems.

5.1 Optimization of the Prefiltering Model

To analyze how optimizing the prefiltering model's hyperparameters affects its performance (RQ1), we experimented with BM25, a lexical model, and SPECTER, a dense retrieval-based model. We conducted the experiments on ACL-ARC and RefSeer, the same two datasets used in the experiments from the previous chapter.

When conducting experiments with BM25, we built a collection of articles by representing them with their corresponding titles and abstracts concatenated into a single sequence. We then performed a grid search over different values for the parameters b and k_1 in BM25's scoring function (cf. Eq. 2.1). Specifically, we considered the values $[0.25, 0.5, 0.75]$ for b and values $[0.5, 1.5, 2.5]$ for k_1, and chose a pair that lead to the highest $R@1000$ score on the validation set of the dataset used.

SPECTER, a pretrained transformer-based article encoder, has no parameters that can explicitly be fine-tuned for its scoring function when used as a retrieval model. However, it demonstrated good performance on a benchmark for evaluating the quality of scientific article representations [54], which makes it a fitting prototype model for the dense retrieval of scientific articles. Since the model has no explicit parameters that can be fine-tuned to improve the retrieval performance as BM25 does, we chose to experiment with different input variants to SPECTER to evaluate its impact on retrieval performance and adapt it to the task of LCR. By design, SPECTER expects two text fields given at input: a title and an abstract of a scientific article. In our experiments, we always passed the title as input but experimented with four different combinations of information passed in the second field, i.e., the one where abstract is expected. Specifically, we considered the following four encoding variants: (1) context only, (2)

b	0.25			0.5			0.75			
k_1	0.5	1.5	2.5	0.5	1.5	2.5	0.5	1.2	1.5	2.5
ACL-ARC	0.665	0.696	0.702	0.714	0.764	0.780	0.746	<u>0.797</u>	0.807	**0.823**
RefSeer	0.565	0.578	0.575	0.581	0.600	0.600	0.592	<u>0.613</u>	0.615	**0.616**

Table 5.1: Performance of BM25 as a prefiltering model in terms of $R@1000$ on the validation sets of the ACL-ARC and RefSeer datasets for different b and k_1 parameter values. Values in **bold** indicate the best scores. <u>Underlined</u> values indicate the performance obtained when using the default parameter values.

	Input variants			
Dataset	context	abstract	context + abstract	abstract + context
ACL-ARC	0.677	0.684	**0.707**	0.691
RefSeer	0.512	0.450	**0.541**	0.476

Table 5.2: Performance of SPECTER as the prefiltering model in terms of $R@1000$ on the validation sets of the ACL-ARC and RefSeer datasets for different input variants. Values in **bold** indicate the performance for the input variant that obtained the best score.

abstract only, (3) context followed by abstract, and (4) abstract followed by context. The first variant corresponds to the standard LCR input with only the citation context given. In the second variant, we only passed the abstract of the citing article, which resorts to the standard input expected in SPECTER. This variant corresponds to the classic GCR input, where context is not given at input. In the third and the fourth variant, we passed both the context and the abstract but in different positions in the concatenation. As with BM25, we evaluated these four variants on the validation sets of the datasets used, again using $R@1000$ as the evaluation metric.

5.1.1 Results

We present the grid search results for BM25's parameters on the validation sets of the two datasets in Table 5.1. In the case of both datasets, the best results were obtained with values of $b = 0.75$ and $k_1 = 2.5$, with the scores in ACL-ARC being overall higher than in the case of RefSeer. Importantly, the results show that the difference in performance can vary depending on the chosen parameters, with the default parameter values for BM25 yielding subpar performance compared to the other combinations.

The results on the validation sets of the two datasets for different input variants in SPECTER are shown in Table 5.2. In the case of both datasets, the best results were obtained with the third input variant, i.e., the one where context is included before the abstract in the second input

	ACL-ARC			RefSeer		
Model	R@1000	R@10	MRR	R@1000	R@10	MRR
BM25	0.806	0.254	0.077	0.624	0.173	0.055
SPECTER	0.704	0.170	0.080	0.545	0.119	0.055

Table 5.3: Performance of BM25 and SPECTER in terms of $R@1000$, $R@10$, and MRR on the test sets of ACL-ARC and RefSeer datasets.

field of SPECTER. This is not surprising, as we evaluated the model on the LCR task, which means that context information is expected to help the model perform well. The performance across different variants varies, which again shows that it pays off to optimize the parameters or the hyperparameters of the prefiltering model, regardless of whether the model is lexical or neural-based.

After identifying the best parameters for each model, we evaluated them on the test sets of the datasets used. Table 5.3 shows these results, where we report $R@1000$, $R@10$, and MRR for each dataset. Overall, comparing the performance of the two models, we note that the results in terms of $R@1000$ and $R@10$ are better when using BM25 in both datasets, whereas the performance in terms of MRR is quite similar between BM25 and SPECTER in both datasets. However, we again note that SPECTER was not trained with LCR's input in mind, which means another dense retrieval model trained specifically for LCR might obtain even better results than BM25.

The main purpose of the prefiltering models in LCR multi-stage pipeline systems is to select k candidates from a pool of articles and pass them onto the next model in the pipeline for reranking. So far, we've only evaluated the model with $k = 10$ or $k = 1000$. As the k increases, $R@k$ is also expected to increase since, in the most extreme case, one can set k to equal the pool's size and obtain the perfect recall. However, as the number of prefiltered candidates increases, so does the time needed to rerank these candidates in the next step of a pipeline. Therefore, detecting the smallest k that provides a good enough trade-off between the recall and the time cost for reranking the candidates is important. To understand how the recall changes, we plot $R@k$ for different values of k for each combination of model and dataset in Figure 5.1. We notice that the rate of the increase gets slower as k grows. Although we did not test values of k larger than 1000, we note that the rate of increase is quite small in all four cases when approaching $k = 1000$, indicating that it might be a safe choice in this trade-off.

Overall, our experiments in this section answer (RQ2): the prefiltering models' parameters and hyperparameters could considerably impact the performance of the LCR systems. We thus argue that these parameters should be optimized in more detail in the future development of LCR systems.

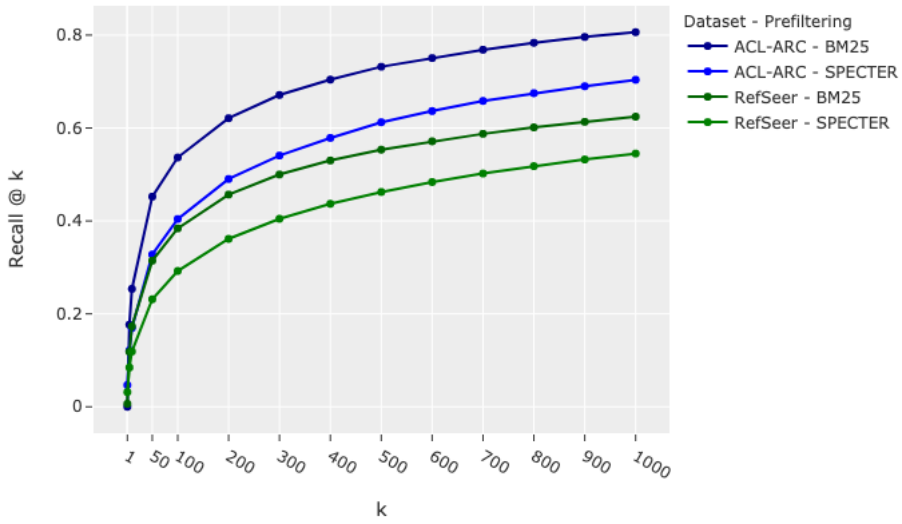


Figure 5.1: Recall @ k for different values of k for combinations of the two datasets and the two prefiltering models.

5.2 Comparison of Different Training Regimes

Having analyzed the impact of the prefiltering models' parameters on its performance, in this section we analyze how different training regimes when it comes to training the reranking model affect the performance of an LCR pipeline system (RQ3). As mentioned in the introduction to this chapter, we differentiate between a standard and a strict training regime. The former corresponds to the one used in most of the previous work, where all the citation contexts from the training set are used to construct the training triplets, regardless of whether the prefiltering model provided a candidate set with the correct recommendation for a given context. The latter regime uses only a subset of the citation contexts from the training set, namely, those for which the prefiltering model provided a correct recommendation in the generated candidate set. While the strict regime avoids the dataset shift between the train and test time that typically happens with the standard regime, it uses a smaller number of contexts for training, i.e., a smaller training set, with size depending on the performance of the prefiltering model. However, the benefit of the strict regime is that training the model with that regime leads to a more realistic performance estimation.

We evaluated these different training regimes on the ACL-200 and RefSeer datasets. In the standard regime, the number of contexts used for the training corresponds to the total number of contexts available in the train set of each dataset, as reported in Table 4.1. For the strict regime, we filtered the contexts based on the performance of the prefiltering model used. Specifically, for the reranking model trained on the ACL-ARC dataset combined with BM25 and SPECTER prefiltering, the number of training contexts was 25,1148 and 21,722, respectively, while for the

same prefiltering models and RefSeer dataset, the size was reduced to 2,126,407 and 1,808,104 contexts, respectively.

As the reranking model, we experimented with TEXTENH and TEXTBIBENH variants of the reranking model introduced in Chapter 4, where TEXTBIBENH corresponds to the version with the weighted sum score. For each combination of the reranking model, dataset, and training regime, we constructed a dataset by generating triplets as in Chapter 4, i.e., randomly sampling negative articles for a given pair of citation context and a cited article. We kept the batch size and the number of triplets per pair the same across all the combinations (batches of 300 triplets, with 10 triplets per pair). Each reranking model was trained for 100 epochs, where an epoch was defined as a sequence of 100 randomly sampled batches. This means the number of parameter updates was the same across all combinations, regardless of the dataset size. The rest of the hyperparameters were set to values reported in [64]. To avoid overfitting, we tracked the performance of the reranking model on a validation set after every five epochs and store the model that performed the best in terms of *MRR*. This model was then evaluated on the test set.

As the goal of a strict regime is to provide a more realistic estimation, we applied the same filtering of citation contexts as for the training set on the validation and test sets as well. This means that we did not rerank the candidates for the citation contexts for which the prefiltering model provided a candidate set without the correct recommendation in it. Subsequently, the values on the test sets we report for the evaluation in this section are lower than those reported in Chapter 4, as in that case, correct recommendations were manually inserted in the test set if the prefiltering model failed to retrieve them. We used the same test sets for standard and strict regimes to compare the two. The test sets for the ACL-ARC dataset when using BM25 and SPECTER as prefiltering models contained 7,568 and 6,621 contexts, respectively, while in the case of RefSeer, the number of contexts with the two prefiltering models was 78,686 and 68,683, respectively.

5.2.1 Results

The results of the comparison of the two regimes are given in Table 5.4. The values represent averages over three different training runs. Important to note is that only the results for the same dataset-prefiltering model combination are comparable, as the number of contexts in the test set depends on that.

Overall, the results show that training with the strict regime leads to better performance than the standard regime in all four dataset-prefiltering model combinations. Among the metrics, especially noticeable is the improvement in $R@10$ in the case of all four combinations, where strict regime yields, on average, a 12% and 3.3% improvement over the standard regime in the RefSeer and ACL-ARC datasets, respectively. Both TEXTENH and TEXTBIBENH-WS reranking models perform better when trained with the strict regime than the standard one. The

			TEXTENH		TEXTBIBENH-WS	
Dataset	Prefiltering	Regime	R@10	MRR	R@10	MRR
ACL-ARC	BM25	standard	0.513	0.259	0.692	0.375
		strict	**0.531**	**0.268**	**0.725**	**0.401**
	SPECTER	standard	0.534	0.275	0.708	0.387
		strict	**0.551**	**0.287**	**0.729**	**0.399**
RefSeer	BM25	standard	0.266	0.109	0.292	0.132
		strict	**0.300**	**0.116**	**0.324**	**0.147**
	SPECTER	standard	0.264	0.122	0.268	0.123
		strict	**0.297**	**0.139**	**0.301**	**0.137**

Table 5.4: Results in terms of *R*@10 and *MRR* of the reranking model variants TEXTENH and TEXTBIBENH-WS on the ACL-ARC and RefSeer datasets for the standard and strict training regimes. **Bold** values indicate the best-performing training regime for a reranking model variant and a combination of a dataset and a prefiltering model.

difference in the performance exists between the datasets, where the values are generally lower in the case of RefSeer than in ACL-ARC, which might result from a larger number of fields covered in RefSeer.

The obtained results demonstrate the benefit of training with a strict training regime compared to the standard one. The results answer our RQ3 from Chapter 1 and show that the training regime should be considered an important design choice when training LCR systems.

5.3 Negative Sampling Strategies

The previous two subsections have demonstrated that the performance of a pipeline can be further improved by optimizing the prefiltering model's parameters and using a strict regime when training the reranking model. Here we turn to another design choice that can further improve the performance: the choice of negative examples in triplets when training the reranking model (RQ4). The choice of negative examples was discussed and evaluated in our previous work [58], which we expand in more details below.

We analyzed this effect when training the same reranking model variants as in the previous section: TEXTENH and TEXTBIBENH-WS. The parameter that we modified compared to the baseline model presented in Chapter 4 is the sampling of negative articles in the triplets of the training set. Whereas in Chapter 4 we trained a model with randomly sampled negative items, here we experimented with different strategies that we list and describe below.

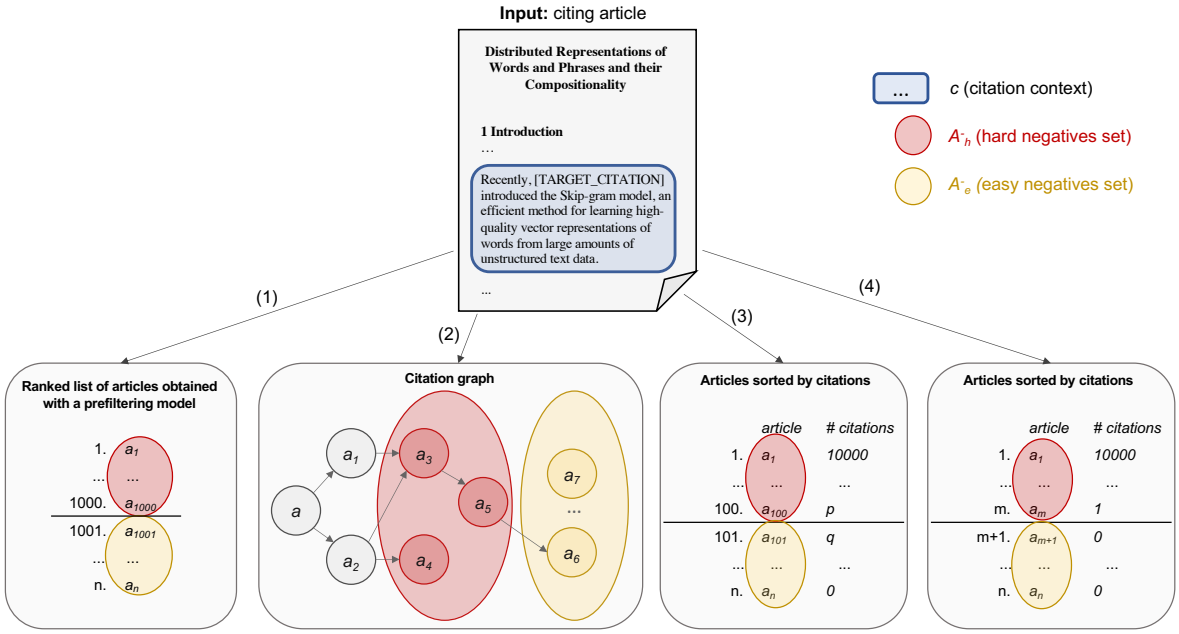


Figure 5.2: Construction of hard and easy negatives sets using four different negative sampling strategies. The input is a citing article with context c. The arrow labels indicate the sampling strategy: (1) prefiltered, (2) graph neighbors, (3) most citations, and (4) cited. Red ovals correspond to the hard negatives set (A_h^-), while yellow ovals correspond to the easy negatives set (A_e^-) for a given strategy.

Previous research on negative sampling demonstrated that combining hard and easy negatives improves the encoder model's performance over using only hard or only easy negatives. Each strategy we considered generates a set of hard and easy negatives from which we sampled negative items for triplets. Specifically, let c represent a citation context, a^+ an article cited in c (i.e., a positive item), and A^- a set of articles that were not cited in c (i.e., all possible negative items). Each strategy split A^- into two subsets: A_h^-, a set of hard negative items, and A_e^-, a set of easy negative items. We then sampled five negative items from each set when training our models. The difference in sampling strategies is basically in the criterium that is used to split A^- into the two subsets.

In total, we considered four different negative sampling strategies: (1) *prefiltered*, (2) *graph neighbors*, (3) *most citations*, and (4) *cited*. The construction of sets of hard and easy negatives for each strategy is illustrated in Figure 5.2.

Prefiltered. In this strategy, we construct the set A_h^- using the recommendation obtained by the LCR pipeline's prefiltering model, while all the other articles were added to the A_e^- set. The motivation behind the *prefiltered* is that the prefiltering model can be considered a proxy for determining a negative article's relevance for a given context – if an article is ranked highly in the recommendation of a prefiltering model but not cited in the context, it must be more relevant to it than a randomly chosen article.

Graph neighbors. This strategy uses a citation graph to construct A_h^- and A_e^-. A citation graph

is defined as a directed graph with articles as nodes and citation links between the articles as edges, with the direction of the edge being from the citing to the cited article. Given a citing article's node in such a graph, the first-order neighbors of the node are the articles that are one link away from it, i.e., the article cited in it. The second-order neighbors are then articles that are two links away, i.e., articles cited in the cited articles but not in the citing. An analog definition holds for third-, fourth-, and other n-th-order neighbors. Given such a graph structure, the *graph neighbors* strategy creates A_h^- from the second- and third-order neighbors, while A_e^- contains all other articles. Intuitively, nodes closer to the citing article in such a graph are more topically related to it than those farther away.

Most citations. The *most citations* strategy uses citation counts from the dataset to create the sets A_h^- and A_e^-. Specifically, articles cited most often are included in the set A_h^-, while all others are added to A_e^-. The motivation for such a choice is that the most often cited articles seem to fit into many citation contexts.

Cited. Similar to *most citations*, the *cited* strategy also uses citation counts to detect hard and easy negatives. The difference is that the *cited* strategy defines A_h^- as the set of all the articles with at least one citation, while all others are in A_e^-. Here, the motivation is that articles that do not fit in any context in our dataset are less likely to fit into any context from the training set. Therefore, such articles can be considered easy negatives.

Previous work on GCR used sampling strategies similar to *prefiltered* and *graph neighbors* [16]. To the best of our knowledge, our work [72] was the first to experiment with strategies such as *most citations* and *cited*. Regarding the complexity of the described strategies, we note that only the *prefiltered* strategy is the computationally expensive one, as it is necessary to process each training set context with a prefiltering model before training the reranking model. Other strategies can be used more efficiently by building a data structure (a graph or a table) that can be easily queried during the reranker's training.

5.3.1 Results

We evaluated all four strategies on the same train, validation, and test splits as in the experiments with training regimes and again report the average values over three runs. As a baseline, we used random sampling and report the results from Table 5.4. In addition, we compared the performance of strategies coupled with either a standard or a strict training regime. The results for the four different sampling strategies coupled with a standard training regime are given in Table 5.5, while the ones with the strict regime are in Table 5.6. We first focus on the results obtained using the standard training regime, as it shows how much improvement can be obtained by changing only the sampling strategy compared to the default training approach.

The results from Table 5.5 show that in all combinations of a dataset, prefiltering model, and a reranking model variant, a custom negative sampling strategy obtains better results than

		BM25				SPECTER			
		TEXTENH		TBENH-WS		TEXTENH		TBENH-WS	
Dataset	Negative sampling	R@10	MRR	R@10	MRR	R@10	MRR	R@10	MRR
ACL	random (standard reg.)	0.513	0.259	0.692	0.375	0.534	0.275	0.708	0.387
	random (strict reg.)	0.531	0.268	0.725	0.401	0.551	0.287	0.729	0.399
	prefiltered	0.509	0.265	0.679	0.369	0.535	0.289	0.703	0.388
	graph neighbors	0.498	0.248	0.659	0.342	0.538	0.280	0.676	0.363
	most citations	0.462	0.213	0.611	0.308	0.533	0.269	0.607	0.304
	cited	**0.538**	**0.274**	0.735	0.404	**0.569**	**0.300**	0.746	0.413
	cited (only)	0.440	0.199	**0.745**	**0.413**	0.465	0.235	**0.761**	**0.428**
RefSeer	random (standard reg.)	0.266	0.109	0.292	0.132	0.264	0.122	0.268	0.123
	random (strict reg.)	0.300	0.116	0.324	0.147	0.297	0.139	0.301	0.137
	prefiltered	0.261	0.130	0.296	0.151	0.282	0.143	0.307	0.150
	graph neighbors	**0.390**	**0.183**	**0.418**	**0.210**	0.397	0.208	**0.418**	**0.210**
	most citations	0.269	0.111	0.299	0.136	0.267	0.126	0.278	0.130
	cited	0.247	0.104	0.289	0.132	0.245	0.112	0.265	0.123
	graph neighbors (only)	0.389	0.182	0.417	0.208	**0.398**	**0.209**	0.414	0.208

Table 5.5: Results in terms of *R*@10 and *MRR* of different negative sampling strategies used with a combination of a dataset, prefiltering, and a reranking model. TBENH-WS corresponds to TEXTBBIBENH-WS. **Bold** values indicate the best-performing strategy for a given combination of a dataset, prefiltering model, reranking model, and metric. Strategies with the "(only)" suffix correspond to sampling strategy variants trained with only a hard negative set.

random sampling in terms of both *R*@10 and *MRR*. Specifically, improvements are in the range from 0.015 in *MRR* (for the ACL-ARC dataset with BM25 prefiltering and a TEXTBIBENH-WS reranking model) to 0.133 in *R*@10 (for the RefSeer dataset with SPECTER prefiltering and the TEXTENH variant). This confirms that negative sampling is influential in improving a reranking model's performance.

Comparing the strategies across the datasets, it is clear that different strategies yield the best results in each dataset. In the case of ACL-ARC, the best-performing strategy is *cited*, in the case of both BM25 and SPECTER reranking, while in RefSeer, the best-performing one is *graph neighbors*, again both with BM25 and SPECTER reranking. Also noticeable is that the strategies performing well on one dataset perform the worst on the other. For example, while *most citations* strategy performs worse than random sampling in the ACL-ARC dataset, the same strategy outperforms random sampling in the case of RefSeer. These results indicate

		BM25				SPECTER			
		TEXTENH		TBENH-WS		TEXTENH		TBENH-WS	
Dataset	Negative sampling	R@10	MRR	R@10	MRR	R@10	MRR	R@10	MRR
ACL-ARC	prefiltered	0.535	**0.279**	0.689	0.388	0.555	0.300	0.719	0.407
	cited	**0.556**	0.275	0.735	0.407	**0.573**	**0.301**	0.751	0.422
	cited (only)	0.445	0.196	**0.741**	**0.421**	0.481	0.246	**0.761**	**0.431**
RefSeer	prefiltered	0.377	**0.190**	0.395	**0.205**	**0.403**	**0.214**	**0.420**	**0.217**
	graph neighbors	**0.379**	0.161	**0.409**	0.204	0.388	0.199	0.397	0.195

Table 5.6: Results in terms of *R*@10 and *MRR* of different negative sampling strategies used with a combination of a dataset, prefiltering, and a reranking model when trained using a strict training regime. TBENH-WS corresponds to TEXTBBIBENH-WS. **Bold** values indicate the best-performing strategy for a given combination of a dataset, prefiltering model, reranking model, and metric. For brevity, we only report the best-performing strategies.

that the best-performing strategy is dataset-dependent and more research is needed into what constitutes the best strategy for a specific dataset based on its characteristics.

We also analyzed whether the improvement when using a custom negative sampling strategy comes from combining easy with hard negatives or using only the hard negatives is sufficient. To investigate this, we trained reranking models with the best-performing strategy in each dataset but with sampling all negative articles in triplets only from the A_h^- set instead of from both A_h^- and A_e^-. These results are reported in Table 5.5 with the "*(only)*" suffix appended to the strategy name. These strategies also perform differently across the datasets. While *cited (only)* performs worse than *cited* on ACL-ARC, *graph neighbors (only)* performs similarly to *graph neighbors* in RefSeer (except in TEXTENH variant with SPECTER prefiltering). This indicates that combining easy with hard negatives yields overall better performance. A different ratio of hard versus easy negatives per each query might improve the performance even more.

In Table 5.6, we provide the results of the best-performing strategies when coupled with a strict training regime. Comparing these results with those from Table 5.5 allows us to find the best combination of the training regime and a negative sampling strategy. Looking at the results from the strict training regime, we notice that the best-performing strategies, in this case, are inconsistent with the best-performing ones in the standard regime. Specifically, while the best-performing strategies in the ACL-ARC dataset are the same as in the standard regime (except for *MRR* in the TEXTENH variant with BM25 prefiltering where the *prefiltered* strategy obtained the best score), the best-performing strategy in RefSeer when trained with the standard regime, *graph neighbors*, is replaced with *prefiltered* in almost all combinations of metric, reranker, and prefiltering model when trained with the strict regime. We speculate that the improvement

of the *prefiltered* strategy, when coupled with the *strict* regime, is due to a bigger impact that prefiltering has on the overall performance when using a strict regime compared to the standard one.

To answer the RQ4, the change in the best-performing strategy across different training regimes demonstrates that the negative sampling strategy is an important design choice in constructing LCR systems and should be carefully chosen based on other parameters, such as the dataset used for training, prefiltering model, and the training strategy used.

With these experiments, we conclude the analysis of the design choices that affect the performance of the LCR systems. Together with the experiments from Chapter 4, these experiments confirmed that in order to ensure the best possible performance of the LCR systems in practice, it is important to design both the system and the training process carefully. In the next chapter, we turn to another deficiency of the CR systems in practice and try to solve it using a new, paragraph-level citation recommendation task.

Chapter 6

Paragraph-level Citation Recommendation

The previous two chapters focused on the improvements of an LCR pipeline system via combining the citation context with global information (Chapter 4) and optimizing different design choices when building such systems (Chapter 5). The results demonstrated that citation context enhancement and specific design choices could improve LCR systems' performance. However, the question still remains whether GCR and LCR, the two existing task in CR, can deliver any practical benefits to the researchers in the process of writing a scientific article.

Looking at GCR, it quickly becomes evident that the recommendations obtained from a GCR model are quite general, as they cover the overall topic of an article. On the other hand, LCR's recommendations are very specific, as the input is often a detailed description of a cited article, which implies that the user already knows enough about the topic. This raises the question of whether we could use the existing scientific datasets and design a task that is somewhere in between GCR and LCR and maybe more fitting to the process of writing an article while also looking for (previously unknown) relevant work (RQ5 from Chapter 1). In this chapter, we propose the paragraph-level citation recommendation (PCR) task, which provides citation recommendations for a given paragraph in a scientific article. Specifically, given a paragraph's topic sentence, which serves as an introduction to the topic that is discussed in the paragraph, the task is to output all the articles that should be cited in the rest of the paragraph's sentences. The intuition is that using topic sentences instead of (only) global information will enable the model to focus more on certain aspects of the overall article's topic while at the same time avoiding the use of citation contexts would make the model more fitting for the stages of writing in which the authors do not yet know enough about the articles they ought to cite.

We start this chapter with a formal definition of PCR and then present the dataset we used to train the model for this task. We then proceed to describe the model proposed for PCR and present the evaluation results and some interesting findings from a qualitative analysis.

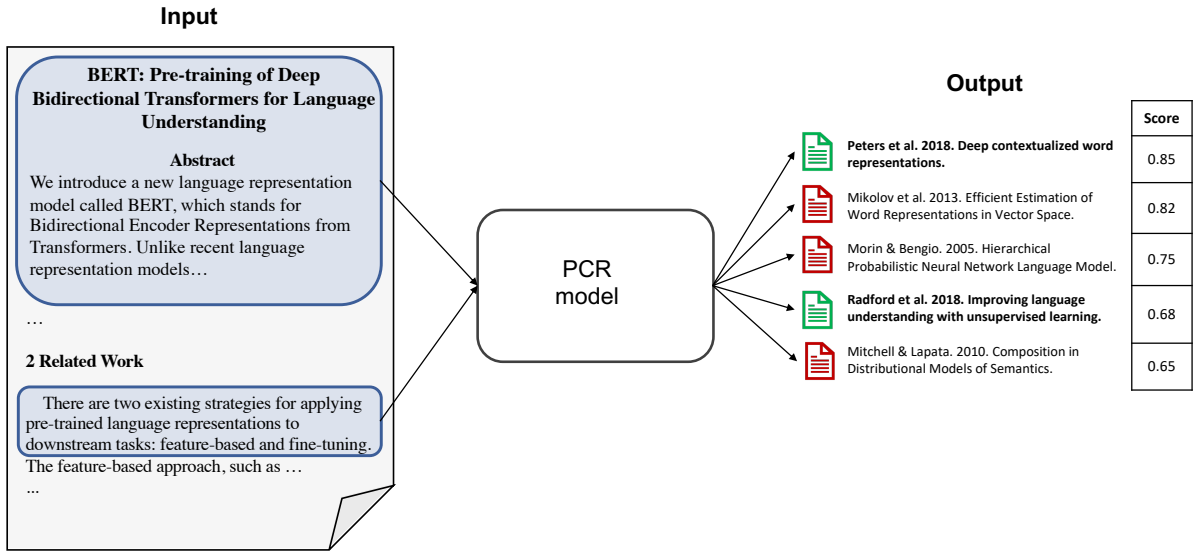


Figure 6.1: Illustration of an input to a PCR model and the generated output. As the input, we propose combining a topic sentence from an RW paragraph with the article's title and abstract. During the training, the model is tasked to output a list of recommendations where the articles cited in the paragraph (green icons) should be ranked higher than those not cited in the paragraph (red icons).

6.1 Task

We framed PCR as a text ranking task like GCR and LCR. Let p_q represent a paragraph from a citing article a_q and let p_q consist of m sentences $[s_1, ..., s_m]$. To construct a query in PCR, we concatenated a_q's title and abstract together with the sentence s_1 from p_q, i.e., p_q's *topic sentence*. The model's task is to output articles as recommendations that should be cited in the rest of the paragraph, i.e., in sentences $[s_2, ..., s_m]$. Figure 6.1 illustrates the input and the output of a PCR model, given the definition from above.

6.2 Dataset

In order to construct the dataset for PCR in a text ranking setup, it is necessary to create a set of queries, a set of articles cited in the paragraphs defined by those queries, and a pool of articles from which recommendations are selected. Here we describe how we got to each of these.

As defined in the previous section, queries were constructed from the topic sentences of paragraphs, so we needed to detect the topic sentences before constructing queries. To detect the topic sentences, we used CORWA [73], a dataset of articles from ACL venues, in which paragraphs from the "Related Work" (RW) sections contain sentence-level annotations indicating each sentence's discourse role. Among the discourse roles available in CORWA, we made use of the *Transition* label, i.e., the one assigned to the sentences that "serve as topic introductions or transitions from one topic to another". If the first sentence of a paragraph was labeled as a *Transition* sentence and there were citations in that paragraph, we added this topic sentence

Title: Predicting Word Concreteness and Imagery **Abstract:** Concreteness of words has been studied extensively in psycholinguistic literature. A number of datasets have been created with average values for perceived concreteness of words. We show that we can train a regression model on these data, using word embeddings and morphological features, that can predict these concreteness values with high accuracy. We evaluate the model on 7 publicly available datasets. Only for a few small subsets of these datasets prediction of concreteness values are found in the literature. Our results clearly outperform the reported results for these datasets. **Topic sentence:** A few studies deal with the question whether values for concreteness can be predicted by machine learning techniques.
Title: Cross-lingual Argumentation Mining: Machine Translation (and a bit of Projection) is All You Need! **Abstract:** Argumentation mining (AM) requires the identification of complex discourse structures and has lately been applied with success monolingually. In this work, we show that the existing resources are, however, not adequate for assessing cross-lingual AM, due to their heterogeneity or lack of complexity. We therefore create suitable parallel corpora by (human and machine) translating a popular AM dataset consisting of persuasive student essays into German, French, Spanish, and Chinese. We then compare (i) annotation projection and (ii) bilingual word embeddings based direct transfer strategies for cross-lingual AM, finding that the former performs considerably better and almost eliminates the loss from cross-lingual transfer. Moreover, we find that annotation projection works equally well when using either costly human or cheap machine translations. Our code and data are available at `http://github.com/UKPLab/coling2018-xling_argument_mining`. **Topic sentence:** Cross-lingual Word Embeddings are the (modern) basis of the direct transfer method.
Title: Cognate-aware morphological segmentation for multilingual neural translation **Abstract:** This article describes the Aalto University entry to the WMT18 News Translation Shared Task. We participate in the multilingual subtrack with a system trained under the constrained condition to translate from English to both Finnish and Estonian. The system is based on the Transformer model. We focus on improving the consistency of morphological segmentation for words that are similar orthographically, semantically, and distributionally; such words include etymological cognates, loan words, and proper names. For this, we introduce Cognate Morfessor, a multilingual variant of the Morfessor method. We show that our approach improves the translation quality particularly for Estonian, which has less resources for training the translation model. **Topic sentence:** Improving segmentation through multilingual learning has been studied before.

Table 6.1: Examples of query instances from the dataset we constructed for the PCR task. Queries are composed by concatenating the citing article's title, abstract, and topic sentence from a paragraph.

(together with the citing article's title and abstract) to the set of PCR queries and all the articles cited in the paragraph to the set of articles relevant for that query. This left us with a total of 790 PCR queries with their corresponding sets of relevant articles.

The limitations of using CORWA for constructing the PCR dataset are that we only analyze and process RW sections of ACL articles. While the limitation of using only ACL articles is not that worrying, as in CR datasets of specific domains are often used (e.g., ACL-ARC), focusing on RW sections could mean that the conclusions from these experiments might not be applicable to other article sections. However, since the discourse in RW sections is indeed different from the rest of the article, as in RW sections authors tell a story of articles topically relevant to the citing one, it seems like an appropriate starting point for further research into PCR across the whole article.

Since the 790 queries that we obtained from CORWA's annotated corpus were hardly sufficient to train and evaluate a new model, we expanded this dataset with silver-labeled queries

obtained by applying a CORWA classifier [73] on all the ACL articles available in S2ORC [46], a large dataset of scientific articles. After applying the classifier to ACL articles, we repeated the same procedure as with the gold-labeled paragraphs – we selected the paragraphs starting with a *Transition*-labeled sentence and included them in the set of queries. In the end, the dataset expanded in this way contained 10,481 queries. Some examples of query instances from the generated dataset are given in Table 6.1.

As in other CR tasks, we applied time-based splitting to generate train, validation, and test splits. Specifically, queries from articles published before 2017, in 2017, and after 2017 were added to the train, validation, and test split, respectively. This resulted in train, validation, and test sets containing 7,773, 560, and 2,148 queries, respectively. A pool of articles was constructed from all the ACL articles and all those cited in ACL articles available in S2ORC. The total size of the pool was 94,129 articles. To make the evaluation as realistic as possible, we made sure to exclude the articles published after the test queries when evaluating the proposed model.

6.3 Paragraph-level Citation Recommendation Model

Following the recent trends in CR, where models are designed as dense retrievers, often without multiple stages in a pipeline, we present one such model for PCR [54, 56]. Specifically, we fine-tuned SciNCL [56], a transformer-based article encoder with architecture the same as SPECTER but trained with a different negative sampling strategy for the training triplets. SciNCL obtained state-of-the-art performance on different benchmarks for scientific article representation [54, 58], which makes it a solid starting point for further fine-tuning.

Different from the models presented in Chapters 4 and 5, where we trained cross-encoder models, here we chose a simpler, bi-encoder design. Given a pair of a query and a pool article, we encoded each input separately with our model, obtaining the embeddings that can be used for the nearest neighbor search. For encoding a pool article, we used the standard input for SciNCL, i.e., passed the title and the abstract to the model to obtain the article's embedding using the final layer's "`[CLS]`" token embedding. To encode a query defined with a topic sentence, we concatenated the citing article's title and abstract with the topic sentence, separating the topic sentence from the rest of the input with a special `[TS]` token and using the final layer's "`[CLS]`" token embedding as the query embedding.

SciNCL was trained with triplet loss, using a careful selection of negative articles, i.e., making sure that the negative examples are challenging and informative for the model to learn. When fine-tuning SciNCL for PCR, we experimented with training with triplet loss and quadruplet loss, a modification of triplet loss that uses multiple positive examples for a query instead of only one. Quadruplet loss has been used previously in LCR, where it outperformed models

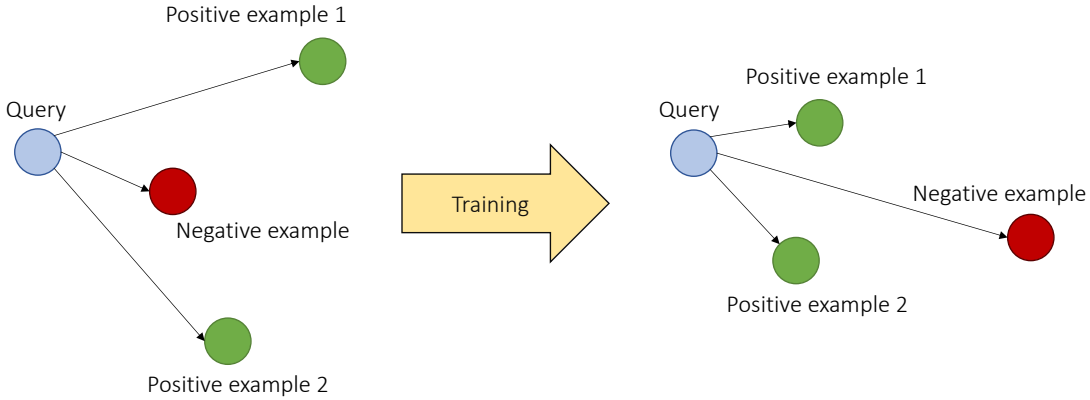


Figure 6.2: Illustration of a change in the arrangement of the query, two positive examples, and a negative example before and after the training with the quadruplet loss. Before the training step, a negative example is closer to the query than the positive ones are, resulting in a non-zero value of the quadruplet loss. After training and updating the parameters, the model "pushes" the positive examples closer to the query and the negative example further away while also enforcing that the positive examples are closer to each other than each of them is closer to the negative example.

trained with triplet loss [74]. Adding this additional item to the triplet introduces new constraints that can be added to the loss calculation. Specifically, let (q, a_1^+, a_2^+, a^-) represent a quadruplet, where q represents the query, a_1^+ and a_2^+ two positive items (articles cited in the query), and a^- a negative item (an article not cited in the query). Having encoded each quadruplet item into its embedding $\mathbf{e}$, we define the quadruplet loss as:

$$\mathscr{L}_q = \sum_i \max(0, \|\mathbf{e^q} - \mathbf{e_i^+}\| - \|\mathbf{e^q} - \mathbf{e^-}\| + m) + \sum_i \max(0, \|\mathbf{e_1^+} - \mathbf{e_2^+}\| - \|\mathbf{e_i^+} - \mathbf{e^-}\| + m) \tag{6.1}$$

where $\|\cdot\|$ is the L2 norm, and m is a margin between the two norms. Comparing this loss with the triplet loss from Eq. 2.3, we notice that $\mathscr{L}_q$ contains an additional term that promotes similarity between the second positive item and the query, alongside the two terms that promote similarity between the two positive items compared to the similarity between them and the negative item. Intuitively, such loss provides a richer signal in the gradient than the standard triplet loss, which might be why it obtained better results than triplet loss-trained models [74]. Figure 6.2 illustrates the change in the arrangement of the quadruplet items during the training (compare to Figure 2.2 for the triplet loss).

Following our results from Chapter 5 and previous work on the importance of carefully sampling negative items in triplets [56], we sampled the negative items for quadruplets from the three pools of negatives. The first pool consisted of the articles cited in the RW section of the citing article but not in the query's paragraph. The second pool constituted articles cited in any other section of the citing article but not in RW. Finally, the third pool contained articles cited in the articles cited in RW section but not in the citing article. In total, we sampled four negative articles from the first and three from the second and third each, resulting in a total of

Model	R-prec	R@5	R@10	MRR
SciNCL-base	6.92	8.47	12.70	17.32
SciNCL-ts	7.62	9.86	14.58	19.23
SciNCL-aclrw	7.61	9.87	14.77	19.21
SciNCL-tri	7.63	9.57	14.58	19.30
qPCR	**8.00**	**10.21**	**15.19**	**20.05**

Table 6.2: Performance of the evaluated PCR models on the test set queries. The results of the best-performing model for each metric are shown in **bold**.

10 quadruplets per query.

To avoid overfitting when fine-tuning SciNCL, we only fine-tuned the last two layers of the model. We fine-tuned the model for five epochs, tracking the performance on the validation set queries regarding R-precision. We call our model **q**uadruplet-based **P**aragraph **C**itation **R**ecommendation model (qPCR).

6.4 Evaluation

We used three baselines in the evaluation of qPCR. The first one was SciNCL, which takes only the citing article's title and abstract as input (SciNCL-base). The second one was again SciNCL, but this time with a topic sentence concatenated with title and abstract at the input (SciNCL-ts), and the third one was SciNCL fine-tuned with triplets of ACL RW articles (SciNCL-aclrw). The third baseline was trained using triplets where a query was an ACL article, a positive item was an article cited in the query's RW section, and a negative item was an article not cited in the query. This baseline helps us analyze whether the improvement of qPCR comes from domain adaptation or training with topic sentences. We also report the results of SciNCL fine-tuned with triplets instead of quadruplets (SciNCL-tri).

As the evaluation metrics in this task, we used *R*-precision, *R*@5, *R*@10, and *MRR*. The evaluation with *R*-precision here is needed as the queries differ with respect to the number of articles cited in the paragraphs they represent. However, as authors often miss out on all the relevant citations or choose not to cite all of them since that would be infeasible, we also report *R*@5 and *R*@10 as more lenient metrics that do not penalize false positives that much.

Table 6.2 shows the results of our evaluation, i.e., average metrics values over three fine-tuning runs for each model. Looking at the baselines, we observe that SciNCL-ts outperforms the SciNCL-base variant, which shows that topic sentence-based queries lead to an improvement in PCR. Furthermore, fine-tuning SciNCL with triplets does not yield improvement over using SciNCL out-of-the-box and simply appending the topic sentence to the query (SciNCL-

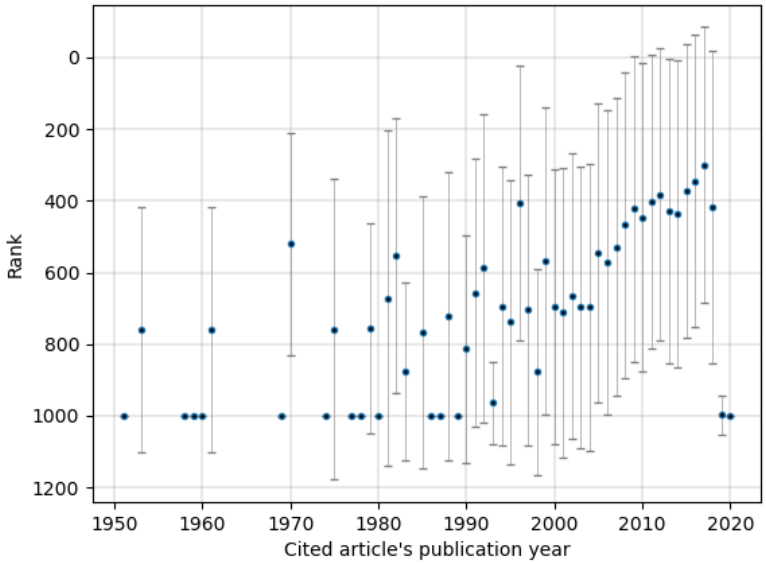


Figure 6.3: Average ranks of cited articles per publication year for the ranking obtained with QPCR.

TS). Finally, domain adaptation over ACL RW data is insufficient to improve the performance over a SCINCL-TS baseline.

Analyzing QPCR's performance, we notice that it outperforms its triplet-trained counterpart SCINCL-TS, confirming that quadruplet loss indeed leads to an improvement over triplet loss-based models. This finding further confirms our conclusions from Chapter 5 – specific design choices when training the CR models can improve the default settings, going even beyond the sampling of negative items in the type of a contrastive loss.

However, the improvement of QPCR over SCINCL-TS, the best-performing model that does not use topic sentences for training, is not very large: it ranges from 0.35 for *R*@5 to 0.82 for *MRR*. This indicates that the task is quite hard, which is not surprising given that, in this setup, we do not model the recommendations using metadata but only with text data (different from the models presented in the previous two chapters). As previous research has shown, authors choose citations based on many different criteria, often not based merely on text [38].

Qualitative Analysis. To understand better what drives QPCR's performance, we conducted a qualitative analysis and manually compared the recommendations obtained by QPCR and the most competitive baseline SCINCL-TS. We found that both models often provided recommendations that seem relevant for a given paragraph, but those were not counted as correct recommendations because they were not cited in the paragraph. In reality, this problem occurs not just in PCR but in other CR tasks as well, especially if the recommendations are obtained solely based on text information, as is the case here. However, this also highlights the evaluation issue in CR, where datasets with stronger relevance signals than pure citations are needed to track the performance of CR systems reliably.

Our analysis found that both QPCR and SCINCL-TS performed worse as that gap in years

<table>
<tr><td colspan="4">Title: CommonsenseQA: A Question Answering Challenge Targeting Commonsense Knowledge</td></tr>
<tr><td colspan="4">Year: 2018</td></tr>
<tr><td colspan="4">Abstract: When answering a question, people often draw upon their rich world knowledge in addition to some task-specific context. Recent work has focused primarily on answering questions based on some relevant document or content, and required very little general background. To investigate question answering with prior knowledge, we present CommonsenseQA: a difficult new dataset for commonsense question answering. To capture common sense beyond associations, each question discriminates between three target concepts that all share the same relationship to a single source drawn from ConceptNet (Speer et al., 2017). This constraint encourages crowd workers to author multiple-choice questions with complex semantics, in which all candidates relate to the subject in a similar way. We create 9,500 questions through this procedure and demonstrate the dataset's difficulty with a large number of strong baselines. Our best baseline, the OpenAI GPT (Radford et al., 2018), obtains 54.8% accuracy, well below human performance, which is 95.3%.</td></tr>
<tr><td colspan="4">Paragraph: Machine common sense, or the knowledge of and ability to reason about an open ended world, has long been acknowledged as a critical component for natural language understanding. Early work sought programs that could reason about an environment in natural language(McCarthy, 1959), or leverage a world-model for deeper language understanding (Winograd, 1972) . Many commonsense representations and inference procedures have been explored (McCarthy and Hayes, 1969; Kowalski and Sergot, 1986) and large-scale commonsense knowledge-bases have been developed (Lenat, 1995; Speer et al., 2017) . However, evaluating the degree of common sense possessed by a machine remains difficult.</td></tr>
<tr><td>Rank</td><td>SciNCL-TS</td><td>Rank</td><td>QPCR</td></tr>
<tr><td>1</td><td>Philosophers are Mortal: Inferring the Truth of Unseen Facts</td><td>1</td><td>Philosophers are Mortal: Inferring the Truth of Unseen Facts</td></tr>
<tr><td>2</td><td>Acquiring comparative commonsense knowledge from the web</td><td>2</td><td>NewsQA: A Machine Comprehension Dataset</td></tr>
<tr><td>3</td><td>SemEval-2012 Task 7: Choice of Plausible Alternatives: An Evaluation of Commonsense Causal Reasoning</td><td>3</td><td>MCTest: A Challenge Dataset for the Open-Domain Machine Comprehension of Text</td></tr>
<tr><td>4</td><td>Learner: a system for acquiring commonsense knowledge by analogy</td><td>4</td><td>Learner: a system for acquiring commonsense knowledge by analogy</td></tr>
<tr><td>5</td><td>Predicting Answer Location Using Shallow Semantic Analogical Reasoning in a Factoid Question Answering System</td><td>5</td><td>Deep Questions without Deep Understanding</td></tr>
<tr><td>>1000</td><td>Programs with common sense</td><td>>1000</td><td>Programs with common sense</td></tr>
<tr><td colspan="4">Cited abstract: This paper discusses programs to manipulate in a suitable formal language (most likely a part of the predicate calculus) common instrumental statements. The basic program will draw immediate conclusions from a list of premises. These conclusions will be either declarative or imperative sentences. When an imperative sentence is deduced the program takes a corresponding action. These actions may include printing sentences, moving sentences on lists, and reinitiating the basic deduction process on these lists.</td></tr>
</table>

Table 6.3: An example of a paragraph that cites an article that is older than the citing. Citation marker in the paragraph is printed inolive, as well as the title of the cited article. Although an experienced researcher might conclude from the title and abstract alone that the cited article might be suitable for citing in recent articles on commonsense reasoning, the vocabulary mismatch between the articles might introduce difficulties for CR models.

Title: BanditSum: Extractive Summarization as a Contextual Bandit

Year: 2018

Abstract: In this work, we propose a novel method for training neural networks to perform single-document extractive summarization without heuristically-generated extractive labels. We call our approach BanditSum as it treats extractive summarization as a contextual bandit (CB) problem, where the model receives a document to summarize (the context), and chooses a sequence of sentences to include in the summary (the action). A policy gradient reinforcement learning algorithm is used to train the model to select sequences of sentences that maximize ROUGE score. We perform a series of experiments demonstrating that BanditSum is able to achieve ROUGE scores that are better than or comparable to the state-of-the-art for extractive summarization, and converges using significantly fewer update steps than competing approaches. In addition, we show empirically that BanditSum performs significantly better than competing approaches when good summary sentences appear late in the source document.

Paragraph: *Extractive summarization has been widely studied in the past.* Recently, neural network-based methods have been gaining popularity over classical methods (Luhn, 1958; Gong and Liu, 2001; Conroy and O'leary, 2001; Mihalcea and Tarau, 2004; Wong et al., 2008) , as they have demonstrated stronger performance on large corpora. Central to the neural network-based models is the encoderdecoder structure. These models typically use either a convolution neural network (Kalchbrenner et al., 2014; Kim, 2014; Yin and Pei, 2015; Cao et al., 2015) , a recurrent neural network (Chung et al., 2014; Cheng and Lapata, 2016; Nallapati et al., 2017) , or a combination of the two (Narayan et al., 2018; Wu and Hu, 2018) to create sentence and document representations, using word embeddings (Mikolov et al., 2013; Pennington et al., 2014) to represent words at the input level. These vectors are then fed into a decoder network to generate the output summary. The use of reinforcement learning (RL) in extractive summarization was first explored by Ryang and Abekawa (2012) , who proposed to use the TD(λ) algorithm to learn a value function for sentence selection. ...

Rank	SciNCL-TS	Rank	QPCR
1	Classify or Select: Neural Architectures for Extractive Document Summarization	1	Extractive Multi-Document Summarization with Integer Linear Programming and Support Vector Regression
2	Neural Summarization by Extracting Sentences and Words	2	Abstractive Text Summarization Using Sequence-to-Sequence RNNs and Beyond
3	A Neural Attention Model for Abstractive Sentence Summarization	3	Neural Summarization by Extracting Sentences and Words
4	Abstractive Text Summarization Using Sequence-to-Sequence RNNs and Beyond	4	Multi-Document Summarization Using A* Search and Discriminative Learning
5	A Neural Attention Model for Sentence Summarization	5	SummaRuNNer: A Recurrent Neural Network based Sequence Model for Extractive Summarization of Documents
>1000	The automatic creation of literature abstracts	>1000	The automatic creation of literature abstracts

Cited abstract: Excerpts of technical papers and magazine articles that serve the purposes of conventional abstracts have been created entirely by automatic means. In the exploratory research described, the complete text of an article in machine-readable form is scanned by an IBM 704 data-processing machine and analyzed in accordance with a standard program. Statistical information derived from word frequency and distribution is used by the machine to compute a relative measure of significance, first for individual words and then for sentences. Sentences scoring highest in significance are extracted and printed out to become the "auto-abstract."

Table 6.4: Another example of an article that cites an older work. Similarly as in Table 6.3, an experienced researcher might find the cited article fairly relevant as a classical method for extractive summarization, but CR models might struggle with vocabulary mismatch (e.g., "model" in citing is a "machine" in the cited article, "document" is an "article", and "summary" is an "excerpt").

Title: MOROCO: The Moldavian and Romanian Dialectal Corpus
Year: 2019
Abstract: In this work, we introduce the MOldavian and ROmanian Dialectal COrpus (MOROCO), which is freely available for download at this https URL. The corpus contains 33564 samples of text (with over 10 million tokens) collected from the news domain. The samples belong to one of the following six topics: culture, finance, politics, science, sports and tech. The data set is divided into 21719 samples for training, 5921 samples for validation and another 5924 samples for testing. For each sample, we provide corresponding dialectal and category labels. This allows us to perform empirical studies on several classification tasks such as (i) binary discrimination of Moldavian versus Romanian text samples, (ii) intra-dialect multi-class categorization by topic and (iii) cross-dialect multi-class categorization by topic. We perform experiments using a shallow approach based on string kernels, as well as a novel deep approach based on character-level convolutional neural networks containing Squeeze-and-Excitation blocks. We also present and analyze the most discriminative features of our best performing model, before and after named entity removal.
Paragraph: *Other languages.* The Nordic Dialect Corpus (Johannessen et al., 2009) contains about 466K spoken words from Denmark, Faroe Islands, Iceland, Norway and Sweden. The authors transcribed each dialect by the standard official orthography of the corresponding country. Francom et al. (2014) introduced the ACTIV-ES corpus, which represents a cross-dialectal record of the informal language use of Spanish speakers from Argentina, Mexico and Spain. The data set is composed of 430 TV or movie subtitle files. The DSL corpus collection (Tan et al., 2014) comprises news data from various corpora to emulate the diverse news content across different languages. The collection is comprised of six language vari- ety groups. For each language, the collection contains 18K training sentences, 2K validation sentences and 1K test sentences. The ArchiMob corpus (Samardžić et al., 2016) contains manuallyannotated transcripts of Swiss German speech collected from four different regions: Basel, Bern, Lucerne and Zurich. The data set was used in the 2017 and 2018 VarDial Evaluation Campaigns (Zampieri et al., 2017 (Zampieri et al., , 2018 . Kumar et al. (2018) constructed a corpus of five Indian dialects consisting of 307K sentences. The samples were collected by scanning, passing through an OCR engine and proofreading printed stories, novels and essays from books, magazines or newspapers. Romanian. To our knowledge, the only empirical study on Romanian dialect identification was conducted by Ciobanu and Dinu (2016) . In their work, Ciobanu and Dinu (2016) used only a short list of 108 parallel words in a binary classification task in order to discriminate between Daco-Romanian words versus Aromanian, Istro-Romanian and Megleno-Romanian words. Different from Ciobanu and Dinu (2016) , we conduct a large scale study on 33K documents that contain a total of about 10 million tokens.

Rank	**SciNCL-TS**	Rank	**QPCR**
1	German Dialect Identification Using Classifier Ensembles	1	Exploiting Convolutional Neural Networks for Phonotactic Based Dialect Identification
2	Classifying English Documents by National Dialect	2	German Dialect Identification Using Classifier Ensembles
3	Exploiting Convolutional Neural Networks for Phonotactic Based Dialect Identification	3	Native Language Identification with String Kernels
4	A Character-level Convolutional Neural Network for Distinguishing Similar Languages and Dialects	4	Discriminating between Indo-Aryan Languages Using SVM Ensembles
5	CLUZH at VarDial GDI 2017: Testing a Variety of Machine Learning Tools for the Classification of Swiss German Dialects	5	UnibucKernel Reloaded: First Place in Arabic Dialect Identification for the Second Year in a Row

Table 6.5: An example of recommendations for the paragraph describing work on dialect identification in other languages. None of the recommendations of SciNCL-TS and QPCR was cited in the paragraph, although some of the articles could fit into the paragraph's narrative.

Title: Introducing two Vietnamese Datasets for Evaluating Semantic Models of (Dis-)Similarity and Relatedness			
Year: 2018			
Abstract: We present two novel datasets for the low-resource language Vietnamese to assess models of semantic similarity: ViCon comprises pairs of synonyms and antonyms across word classes, thus offering data to distinguish between similarity and dissimilarity. ViSim-400 provides degrees of similarity across five semantic relations, as rated by human judges. The two datasets are verified through standard co-occurrence and neural network models, showing results comparable to the respective English datasets.			
Paragraph: *For other languages, only a few gold standard sets with scored word pairs exist.* Among others, Gurevych (2005) replicated Rubenstein and Goodenough's experiments after translating the original 65 word pairs into German. In later work, Gurevych (2006) used the same experimental setup to increase the number of word pairs to 350. Leviant and Reichart (2015) translated two prominent evaluation sets, WordSim-353 (association) and SimLex-999 (similarity) from English to Italian, German and Russian, and collected the scores for each dataset from the respective native speakers via crowdflower 6 .			
Rank	**SciNCL-TS**	Rank	**QPCR**
1	Comparing Semantic Relatedness between Word Pairs in Portuguese Using Wikipedia	1	Comparing Semantic Relatedness between Word Pairs in Portuguese Using Wikipedia
2	Gathering Information About Word Similarity from Neighbor Sentences	2	A Framework for the Construction of Monolingual and Cross-lingual Word Similarity Datasets
3	Duluth : Measuring Degrees of Relational Similarity with the Gloss Vector Measure of Semantic Relatedness	3	Human and Machine Judgements for Russian Semantic Relatedness
4	A Study on Similarity and Relatedness Using Distributional and WordNet-based Approaches	4	SemEval-2014 Task 3: Cross-Level Semantic Similarity
5	ExB Themis: Extensive Feature Extraction from Word Alignments for Semantic Textual Similarity	5	An English-Chinese Cross-lingual Word Semantic Similarity Measure Exploring Attributes and Relations

Table 6.6: An example of recommendations for the paragraph describing work in other languages on the construction of word pairs similarity datasets. None of the recommendations of SciNCL-TS and QPCR were cited in the paragraph, although some of the articles could fit into the paragraph's narrative.

Title: Part-of-Speech Tagging on an Endangered Language: a Parallel Griko-Italian Resource			
Year: 2018			
Abstract: Most work on part-of-speech (POS) tagging is focused on high resource languages, or examines low-resource and active learning settings through simulated studies. We evaluate POS tagging techniques on an actual endangered language, Griko. We present a resource that contains 114 narratives in Griko, along with sentence-level translations in Italian, and provides gold annotations for the test set. Based on a previously collected small corpus, we investigate several traditional methods, as well as methods that take advantage of monolingual data or project cross-lingual POS tags. We show that the combination of a semi-supervised method with cross-lingual transfer is more appropriate for this extremely challenging setting, with the best tagger achieving an accuracy of 72.9%. With an applied active learning scheme, which we use to collect sentence-level annotations over the test set, we achieve improvements of more than 21 percentage points.			
Paragraph: *Most of the previous approaches are rarely tested on under-represented languages, with research on POS tagging for endangered languages being sporadic.* Ptaszynski and Momouchi (2012) , for example, presented an HMM-based POS tagger for the extremely endangered Ainu language, based on dictionaries, 12 narratives (yukar), using one annotated story (200 words) for evaluation. To our knowledge, no other previous work has extensively tested several approaches on an actual endangered language.			
Rank	**SciNCL-ts**	Rank	**qPCR**
1	CombiTagger: A System for Developing Combined Taggers	1	Distantly Supervised POS Tagging of Low-Resource Languages under Extreme Data Sparsity: The Case of Hittite
2	Distantly Supervised POS Tagging of Low-Resource Languages under Extreme Data Sparsity: The Case of Hittite	2	What Can We Get From 1000 Tokens? A Case Study of Multilingual POS Tagging For Resource-Poor Languages
3	TagMiner: A Semisupervised Associative POS Tagger Effective for Resource Poor Languages	3	Unsupervised adaptation of supervised part-of-speech taggers for closely related languages
4	Improving the PoS tagging accuracy of Icelandic text	4	TagMiner: A Semisupervised Associative POS Tagger Effective for Resource Poor Languages
5	Wiki-ly Supervised Part-of-Speech Tagging	5	Part-of-Speech Tag Disambiguation by Cross-Linguistic Majority Vote

Table 6.7: An example of recommendations for the paragraph describing work on POS tagging for endangered languages. Similar to Table 6.6, none of the recommendations of SciNCL-ts and qPCR were cited in the paragraph, although some of the articles could fit into the paragraph's narrative.

Title: Dependency Parsing for Spoken Dialog Systems			
Year: 2019			
Abstract: Dependency parsing of conversational input can play an important role in language understanding for dialog systems by identifying the relationships between entities extracted from user utterances. Additionally, effective dependency parsing can elucidate differences in language structure and usage for discourse analysis of human-human versus human-machine dialogs. However, models trained on datasets based on news articles and web data do not perform well on spoken human-machine dialog, and currently available annotation schemes do not adapt well to dialog data. Therefore, we propose the Spoken Conversation Universal Dependencies (SCUD) annotation scheme that extends the Universal Dependencies (UD) (Nivre et al., 2016) guidelines to spoken human-machine dialogs. We also provide ConvBank, a conversation dataset between humans and an open-domain conversational dialog system with SCUD annotation. Finally, to demonstrate the utility of the dataset, we train a dependency parser on the ConvBank dataset. We demonstrate that by pre-training a dependency parser on a set of larger public datasets and fine-tuning on ConvBank data, we achieved the best result, 85.05% unlabeled and 77.82% labeled attachment accuracy.			
Paragraph: *Previous work has demonstrated that expanding the UD annotation scheme can result in successful parsers for other domains.* For example, Liu et al. (2018) expand the UD scheme to train a parser for Twitter data. Thus we take a similar approach in expanding the UD scheme to encompass issues common to automated speech transcription.			
Rank	**SciNCL-TS**	Rank	**QPCR**
1	Adding Syntactic Annotations to Transcripts of Parent-Child Dialogs	1	Discourse parsing for multi-party chat dialogues
2	Discourse parsing for multi-party chat dialogues	2	Annotation for and Robust Parsing of Discourse Structure on Unrestricted Texts
3	A Dependency Treebank for English	3	A Dependency Parser for Tweets
4	Discovering Conversational Dependencies between Messages in Dialogs	4	An Iterative Approach for Joint Dependency Parsing and Semantic Role Labeling
5	Annotating Spoken Dialogs: From Speech Segments to Dialog Acts and Frame Semantics	5	Multilingual Dependency Learning: A Huge Feature Engineering Method to Semantic Dependency Parsing

Table 6.8: An example of recommendations demonstrating high topical relevancy of recommended titles for the topic sentence in the case of QPCR. While SciNCL-TS's recommendations are more focused on dialog systems, as that is the main topic of the citing article, QPCR's recommendations are more about universal dependencies, which are mentioned in the topic sentence.

Title: Cross-lingual Argumentation Mining: Machine Translation (and a bit of Projection) is All You Need!			
Year: 2018			
Abstract: Argumentation mining (AM) requires the identification of complex discourse structures and has lately been applied with success monolingually. In this work, we show that the existing resources are, however, not adequate for assessing cross-lingual AM, due to their heterogeneity or lack of complexity. We therefore create suitable parallel corpora by (human and machine) translating a popular AM dataset consisting of persuasive student essays into German, French, Spanish, and Chinese. We then compare (i) annotation projection and (ii) bilingual word embeddings based direct transfer strategies for cross-lingual AM, finding that the former performs considerably better and almost eliminates the loss from cross-lingual transfer. Moreover, we find that annotation projection works equally well when using either costly human or cheap machine translations. Our code and data are available at http://github.com/UKPLab/coling2018-xling_argument_mining.			
Paragraph: *Cross-lingual Word Embeddings are the (modern) basis of the direct transfer method.* As with monolingual embeddings, there exists a veritable zoo of different approaches, but they often perform very similarly in applications (Upadhyay et al., 2016) and seemingly very different approaches are oftentimes also equivalent on a theoretical level (Ruder et al., 2017) .			
Rank	**SciNCL-TS**	Rank	**QPCR**
1	A Two-Phase Approach Towards Identifying Argument Structure in Natural Language	1	Inducing Multilingual Text Analysis Tools Using Bidirectional Recurrent Neural Networks
2	Argumentative Writing Support by means of Natural Language Processing	2	Cross-Lingual Induction and Transfer of Verb Classes Based on Word Vector Space Specialisation
3	Baselines and test data for cross-lingual inference	3	Cross-lingual sentiment transfer with limited resources
4	Using Discourse Structure Improves Machine Translation Evaluation	4	Transferring Coreference Resolvers with Posterior Regularization
5	A Shared Task on Argumentation Mining in Newspaper Editorials	5	Leveraging Monolingual Data for Crosslingual Compositional Word Representations

Table 6.9: Another example of recommendations that demonstrate high topical relevancy of recommended titles for the topic sentence in the case of QPCR. SciNCL-TS's recommendations dominantly cover topics of argumentation mining and machine translation, while QPCR's recommendations are about the cross-lingual transfer, i.e., focused on the topic of the topic sentence.

between a citing and a cited article grew. In Figure 6.3, we plot the average ranking of cited articles per their publication years. The plot confirms this trend, with older articles, on average, ranked lower in the list of recommendations, while newer ones are closer to the top. To verify whether the rank is indeed correlated with the difference in the years between the citing and the cited article, we calculated the Pearson correlation coefficient between the two and obtained the coefficient value of 0.15 ($p<0.05$), indicating a small positive correlation. One possible reason for such a correlation might be the difference in the vocabulary between the two articles from different times – as fields develop, certain concepts get renamed, which might be difficult for the model to figure out. To find out whether our hypothesis holds, we again calculated the Pearson correlation coefficient, but this time between the difference in the publication years and the Jaccard index between the tokens in the query as defined in PCR and the tokens in the cited article. We obtained a coefficient of -0.24 ($p<0.05$), indicating that a larger difference in publication years is correlated with a smaller vocabulary overlap between the citing and the cited article. Some examples of queries and articles cited in them with a large publication year difference are given in Tables 6.3, and 6.4.

While conducting the manual analysis, we noticed that QPCR more often than SCINCL-TS recommended articles with titles more topically relevant to the topic sentence. To verify whether this holds for all queries, we encoded topic sentences and titles of articles recommended by both QPCR and SCINCL-TS with CoSentBert [75], a variant of Sentence-BERT [76]. Sentence-BERT is a transformer-based model that outputs sentence embeddings so that semantically similar sentences end up with a similar embedding. CoSentBert is a variant of Sentence-BERT trained on a corpus of scientific data, which makes it a suitable model for analyzing how topically similar two sentences from a scientific domain (i.e., a topic sentence and a title) are. After embedding topic sentences and titles with CoSentBert, we calculate an average L2 distance between the topic sentence embeddings and the embeddings of the titles of articles recommended by both models. We obtain an average distance of 14.16 between topic sentences and titles obtained by QPCR, while in the case of SCINCL-TS, the average distance is 14.38. This confirms that QPCR provides recommendations with titles more similar to the topic sentence than SCINCL-TS. Examples of such recommendations can be found in Tables 6.5, 6.6, and 6.7.

We also analyzed whether QPCR's attention heads focus more on the topic sentence tokens than SCINCL-TS. If that were the case, that would mean that the model finds the information from the topic sentences useful for generating query embeddings. Indeed, on average, QPCR's attention heads assign higher weights to the topic sentence tokens than SCINCL-TS's attention heads. Specifically, in the case of QPCR, the average sum of the attention heads assigned to the topic sentence tokens across all layers is 0.25, while the same value for SCINCL-TS is 0.12. Examples of QPCR's recommendations with titles that seem more topically relevant to the topic

sentence than those of SciNCL-TS are in Tables 6.8 and 6.9.

All in all, in this chapter, we have introduced a new approach to the task of CR that is designed to output citation recommendations for the paragraphs in the RW sections of the scientific articles. We proposed and evaluated a transformer-based model for this task, which showed a promising performance and outperformed the baseline models it was compared to. Our analysis showed that the difference in publication years between the query and a pool article affects the article's ranking and that the model uses topic sentence information when embedding its queries, confirming the potential of the new task for the future. The results of our experiments give an affirmative answer to the RQ5 from Chapter 1, i.e., the existing datasets and models can be successfully combined to design a task and a model that is better suited for the intermediate stages of the scientific writing process. In the next chapter, we give an overview of our findings from the three previous chapters and relate them to other findings from the domain of SDP and analysis of scientific writing.

Chapter 7

Discussion and Outlook

This thesis's findings and proposals have focused on the performance improvements of an LCR system when designed as a two-stage pipeline and on the improvements in the practical usage of CR systems through the proposal of a new CR task and model. This chapter highlights the obtained answers and some avenues for future work in CR systems that build on the findings.

7.1 Findings

In Chapter 1, we have introduced the research questions this thesis investigated. Here we list them again and provide answers for each one based on the results from the experiments described in the previous chapters.

RQ1: Can enhancing citation context with global information in a DL-based model for LCR improve the performance of the model? Yes, the experiments from Chapter 4 demonstrated that enhancing citation context with global information in a cross-encoder setup of a reranking model for LCR leads to an improvement in performance in the case of datasets with smaller citation contexts. This finding suggests that further research is needed to ensure that citation contexts in the LCR task contain sufficient information for the model to make recommendation decisions.

RQ2: Does optimization of prefiltering model's hyperparameters affect the performance of an LCR system? Yes, careful optimization of the prefiltering model's hyperparameters can further boost the performance of an LCR system, which we demonstrated with the experimental results from Chapter 5. Based on these results, we conclude that further research on LCR should not skip the optimization of the hyperparameters, as was done in many previous works.

RQ3: Does using different training regimes for training the reranking model in the second stage of an LCR pipeline system affect the system's performance? Yes, using the strict training regime instead of the standard one improves performance, as demonstrated in experiments in Chapter 5. We connect this improvement to the fact that a strict regime successfully mitigates dataset shift that affects the model's performance when using a standard training regime, which is dominantly used in practice.

RQ4: How does the performance of the LCR system change depending on the negative sampling strategy for the construction of training triplets in the training of a triplet-based reranking model in an LCR pipeline? Different negative sampling strategies lead to the best performance in different datasets, as demonstrated in experiments from Chapter 5. While no single strategy works for all datasets, it is worth noting that the strategies we considered outperformed the random sampling strategy in many cases, which is the one most often used in previous work. This finding informs the future work on LCR systems that should pay more attention to sampling strategies to get the best possible performance on the dataset of interest.

RQ5: Can the existing datasets and models be used to design a new CR task and model better suited for the intermediate stages of the scientific article writing process? Finally, the answer to the last RQ is affirmative, as presented in Chapter 6. Using CORWA, a dataset of discourse role-labeled sentences from RW sections of ACL articles, and SciNCL, a state-of-the-art article encoder based on the transformer architecture, we have created a dataset for the task of PCR and trained a model for it. The results we obtained look promising and show potential for including such models in the scientific writing process, especially in the intermediate stages of it, as the model relies on the topic sentences of the RW paragraphs.

7.2 Future Work

Having outlined the findings of the thesis, we now turn to ideas for future work that span the development and evaluation of CR models, as well as combining CR with other tasks in SDP.

Pretraining tasks. Experiments from Chapter 4 and 5 demonstrated that certain technical design choices when building LCR systems could improve the overall performance of a system. Admittedly, the choice of the system's parameters that we analyzed is somewhat arbitrary in the sense that many others could also be analyzed and probably would also lead to an improvement in the performance of such systems. One such design choice is the choice of the pretraining task used for pretraining the transformer-based encoders, such as SPECTER and SciNCL. Recent work in the domain of question-answering, which is also typically framed in part as a text

ranking task, demonstrated that pretraining the encoders on the "inverse cloze task" improves the model's performance on the downstream tasks [77, 78]. The idea behind the inverse cloze task is first to encode a sequence of words by the model and then predict which document in a pool of documents this sequence was extracted from. Such pretraining helps the model to relate sentences with the documents from which they originated, which is especially helpful in the text ranking tasks, even though the query, in practice, is not entirely identical to a sentence in a document that is relevant for that query. This pretraining, or some other IR-relevant variant, was not used in the previous work on CR, but could also help improve the overall performance.

Citation context identification. While our experiments from Chapter 4 showed that the model's performance could be enhanced with global information when the citation context is smaller, we did not answer how to identify a sufficient and complete citation context. The research on the citation function classification, in which the task is to classify a citation in a context based on the author's intent behind it, demonstrated the importance of identifying all the text subsequences that refer to a specific citation before classifying its function [79]. As authors might coreference a citation marker often and sometimes in discontinued text segments, using fixed length citation contexts in the LCR task introduces noise in the training process, as not necessarily all the context is about the same article. Combining citation context identification with LCR might improve the model's performance and help answer the question of what constitutes a sufficient and complete citation context.

Multi-task learning. Another line of work that could potentially pay off is combining CR with other tasks that take citation context as input, similar to [57]. For example, a multi-task approach in which a model would be trained to produce both recommendations for a given context and classify the context's citation function might improve both tasks. In addition, it could also expand the capabilities of a typical LCR system, which could now provide a function as information about how the recommended article is related to the citing one.

Retrieval-grounded generation. The improvement in language modeling brought about numerous pre-trained language models, some of which are generative and can produce high-quality text that can often seem human-written [80, 81, 82]. Following this line of work, [83] introduced Galactica, a language model trained on a large corpus of scientific articles that is capable of generating text resembling scientific articles and overall scientific discourse. However, as such generative models tend to hallucinate, i.e., generate false information in the produced text, Galactica and the like are not still suited for usage in knowledge-intensive text-producing tasks such as the writing of scientific articles. The approaches that generate text based on the retrieved documents are able to reduce the hallucinations in such cases [84] and could be adapted

to CR as well. For example, PCR queries could be used to retrieve the relevant articles, which could then be used to ground the paragraphs produced by a generative language model.

Field-customized CR. In all the experiments conducted in this thesis, we treated all the citations equally, regardless of the reason behind the citation. However, [85] reviewed different studies that analyzed the reasons authors cite other work in their own and find out that citations differ with respect to their influence on the citing article and that authors across different scientific fields behave differently when citing other work. Specifically, the studies showed that the reasons for citing are different in different scientific fields, indicating that field-specific CR systems that could catch these reasons might be a better approach than a single CR system for all the fields. In their work, [85] also highlight that bias towards popular authors or venues exists in many fields and that citation function labels assigned by the annotators in the popular citation function datasets often do not match the intent the authors had when citing the relevant work. Overall, it seems that insights from citation behavior should be better incorporated into developing CR systems in the future.

Chapter 8

Conclusion

Finding relevant articles for citing when writing research articles is becoming more and more difficult in today's era of information overload. Systems that automate this process and provide relevant articles as recommendations for citing in a given query are the focus of much previous research in the task of citation recommendation. Improving such systems could potentially ease access to information for scientists, which could benefit science.

In this thesis, we have focused on improving local citation recommendation (LCR) systems, i.e., systems that output recommendations for citing in a citation context extracted from a citing article's full text. We first analyzed whether including the citing article's title and abstract alongside the citation context helps the model's performance. We described a cross-encoder model based on a bidirectional LSTM network that uses the attention mechanism to provide relevance scores for a given pair of a citation context and an article being scored for relevance. The results on two different datasets showed that enhancing citation context with global information helps in the case of smaller citation contexts.

Next, we investigated how different design choices in building LCR systems affect the model's performance. Although the choices we analyzed were often overlooked in previous research, our experiments have shown that a careful design of both the system and the training process can improve the overall performance. Specifically, we have demonstrated that careful optimization of the model's parameters in the prefiltering stage of an LCR pipeline can boost the system's performance. Furthermore, we have identified a strict training regime, in which the reranking model in the second pipeline stage is trained on a subset of the training queries for which the previous model was successful, as a better alternative to the standard training regime, where the same training set is used for training both pipeline models. Finally, we have experimented with different strategies for sampling negative items in triplet-based reranking models. The evaluation identified different sampling strategies as the best-performing for different datasets, indicating that a dataset-specific selection of a negative sampling strategy can boost the model's performance.

Finally, to improve the practical usability of CR systems, we have proposed a new CR task called paragraph-level citation recommendation (PCR). In the proposed task, the goal is to output citation recommendations for citing in a paragraph that is described with its topic sentence alongside the title and abstract of the citing article. Defined in such a way, the task presents a middle ground between the general and specific recommendations which are obtained in the existing global and local citation recommendation tasks, respectively. We have also constructed a dataset for PCR and trained a model based on the transformer architecture and quadruplet loss function, which outperformed the baseline models. The qualitative analysis of the proposed model's recommendations revealed that the model uses the topic sentence information when generating query embeddings and provides recommendations that are topically more related to the topic sentence than those from the baseline model.

While the experiments have provided guidelines for future work on developing CR systems and making them practically useful as a tool for overcoming the information overload problem, we believe the experiments have just scratched the surface of the potential work on this topic. We hope the findings from this thesis will serve as a starting point and motivation for creating practically usable CR models, either as standalone systems or integrated into bigger systems that offer other solutions in the domain of scholarly document processing.

Bibliography

[1]Tenopir, C., King, D. W., Christian, L., Volentine, R., "Scholarly article seeking, reading, and use: a continuing evolution from print to electronic in the sciences and social sciences", Learned Publishing, Vol. 28, No. 2, 2015, pages 93–105.

[2]Tenopir, C., King, D. W., Edwards, S., Wu, L., "Electronic journals and changes in scholarly article seeking and reading patterns", in Aslib proceedings, Vol. 61, No. 1. Emerald Group Publishing Limited, 2009, pages 5–32.

[3]Gupta, S., Manning, C. D., "Analyzing the Dynamics of Research by Extracting Key Aspects of Scientific Papers", in Proceedings of 5th International Joint Conference on Natural Language Processing, 2011, pages 1–9.

[4]Kim, S. N., Medelyan, O., Kan, M.-Y., Baldwin, T., "Semeval-2010 Task 5: Automatic Keyphrase Extraction from Scientific Articles", in Proceedings of the 5th International Workshop on Semantic Evaluation, 2010, pages 21–26.

[5]Teufel, S., "Argumentative Zoning: Information Extraction from Scientific Text", Doktorski rad.

[6]Green, N., "Identifying Argumentation Schemes in Genetics Research Articles", in Proceedings of the 2nd Workshop on Argumentation Mining, 2015, pages 12–21.

[7]Lauscher, A., Glavaš, G., Ponzetto, S. P., "An Argument-annotated Corpus of Scientific Publications", in Proceedings of the 5th Workshop on Argument Mining, 2018, pages 40–46.

[8]Abu-Jbara, A., Ezra, J., Radev, D., "Purpose and Polarity of Citation: Towards NLP-based Bibliometrics", in Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2013, pages 596–606.

[9]Teufel, S., Siddharthan, A., Tidhar, D., "Automatic Classification of Citation Function", in Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, 2006, pages 103–110.

[10]Pride, D., Knoth, P., “An authoritative approach to citation classification”, in Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020, 2020, pages 337–340.

[11]Qazvinian, V., Radev, D. R., “Scientific Paper Summarization using Citation Summary Networks”, in Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1. Association for Computational Linguistics, 2008, pages 689–696.

[12]Chen, J., Zhuge, H., “Summarization of Scientific Documents by Detecting Common Facts in Citations”, Future Generation Computer Systems, Vol. 32, 2014, pages 246–252.

[13]Bethard, S., Jurafsky, D., “Who Should I Site: Learning Literature Search Models from Citation Behavior”, in Proceedings of the 19th ACM International Conference on Information and Knowledge Management. ACM, 2010, pages 609–618.

[14]He, Q., Pei, J., Kifer, D., Mitra, P., Giles, L., “Context-aware Citation Recommendation”, in Proceedings of the 19th International Conference on World Wide Web. ACM, 2010, pages 421–430.

[15]Huang, W., Wu, Z., Liang, C., Mitra, P., Giles, C. L., “A Neural Probabilistic Model for Context Based Citation Recommendation”, in Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015.

[16]Bhagavatula, C., Feldman, S., Power, R., Ammar, W., “Content-Based Citation Recommendation”, in Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), 2018, pages 238–251.

[17]Ali, Z., Qi, G., Muhammad, K., Kefalas, P., Khusro, S., “Global Citation Recommendation employing Generative Adversarial Network”, Expert Systems with Applications, 2021, page 114888, available at: https://www.sciencedirect.com/science/article/pii/S0957417421003298

[18]Ebesu, T., Fang, Y., “Neural Citation Network for Context-Aware Citation Recommendation”, in Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval. ACM, 2017, pages 1093–1096.

[19]Gu, N., Gao, Y., Hahnloser, R. H., “Local citation recommendation with hierarchical-attention text encoder and scibert-based reranking”, in European Conference on Information Retrieval. Springer, 2022, pages 274–288.

[20]Wang, L., Lin, J., Metzler, D., “A Cascade Ranking Model for Efficient Ranked Retrieval”, in Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval, 2011, pages 105–114.

[21]Asadi, N., Lin, J., “Effectiveness/Efficiency Tradeoffs for Candidate Generation in Multi-Stage Retrieval Architectures”, in Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval, 2013, pages 997–1000.

[22]Lin, J., Nogueira, R., Yates, A., “Pretrained transformers for text ranking: Bert and beyond”, Synthesis Lectures on Human Language Technologies, Vol. 14, No. 4, 2021, pages 1–325.

[23]Luhn, H. P., “The automatic creation of literature abstracts”, IBM Journal of research and development, Vol. 2, No. 2, 1958, pages 159–165.

[24]Salton, G., Wong, A., Yang, C.-S., “A vector space model for automatic indexing”, Communications of the ACM, Vol. 18, No. 11, 1975, pages 613–620.

[25]Robertson, S., Walker, S., Jones, S., Hancock-Beaulieu, M. M., Gatford, M., “Okapi at trec-3”, in Overview of the Third Text REtrieval Conference (TREC-3). Gaithersburg, MD: NIST, January 1995, pages 109-126, available at: https://www.microsoft.com/en-us/research/publication/okapi-at-trec-3/

[26]Lipani, A., Lupu, M., Hanbury, A., Aizawa, A., “Verboseness Fission for BM25 Document Length Normalization”, in Proceedings of the 2015 International Conference on the Theory of Information Retrieval, 2015, pages 385–388.

[27]Lv, Y., Zhai, C., “Adaptive Term Frequency Normalization for BM25”, in Proceedings of the 20th ACM international conference on Information and knowledge management, 2011, pages 1985–1988.

[28]Knuth, D. E., “Retrieval on secondary keys”, The art of computer programming: Sorting and Searching, Vol. 3, 1997, pages 550–567.

[29]Furnas, G. W., Landauer, T. K., Gomez, L. M., Dumais, S. T., “The vocabulary problem in human-system communication”, Communications of the ACM, Vol. 30, No. 11, 1987, pages 964–971.

[30]Johnson, J., Douze, M., Jégou, H., “Billion-scale similarity search with GPUs”, IEEE Transactions on Big Data, Vol. 7, No. 3, 2019, pages 535–547.

[31]Schultz, M., Joachims, T., “Learning a Distance Metric from Relative Comparisons”, Advances in Neural Information Processing Systems, Vol. 16, 2003, pages 41–48.

[32]Mikolov, T., Chen, K., Corrado, G., Dean, J., “Efficient estimation of word representations in vector space”, in 1st International Conference on Learning Representations, ICLR

2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings, Bengio, Y., LeCun, Y., (ur.), 2013, available at: http://arxiv.org/abs/1301.3781

[33]Harris, Z. S., “Distributional structure”, Word, Vol. 10, No. 2-3, 1954, pages 146–162.

[34]Hochreiter, S., Schmidhuber, J., “Long Short-term Memory”, Neural Computation, Vol. 9, No. 8, 1997, pages 1735–1780.

[35]Bahdanau, D., Cho, K. H., Bengio, Y., “Neural machine translation by jointly learning to align and translate”, in 3rd International Conference on Learning Representations, ICLR 2015, 2015.

[36]Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., Polosukhin, I., “Attention is all you need”, Advances in neural information processing systems, Vol. 30, 2017.

[37]Järvelin, K., Kekäläinen, J., “Cumulated Gain-based Evaluation of IR Techniques”, ACM Transactions on Information Systems (TOIS), Vol. 20, No. 4, 2002, pages 422–446.

[38]Mammola, S., Piano, E., Doretto, A., Caprio, E., Chamberlain, D., “Measuring the influence of non-scientific features on citations”, Scientometrics, 2022, pages 1–15.

[39]Küçüktunç, O., Saule, E., Kaya, K., Çatalyürek, Ü. V., “Diversifying citation recommendations”, ACM Transactions on Intelligent Systems and Technology (TIST), Vol. 5, No. 4, 2014, pages 1–21.

[40]Chakraborty, T., Modani, N., Narayanam, R., Nagar, S., “Discern: a diversified citation recommendation system for scientific queries”, in 2015 IEEE 31st international conference on data engineering. IEEE, 2015, pages 555–566.

[41]Jiang, Z., Yin, Y., Gao, L., Lu, Y., Liu, X., “Cross-language Citation Recommendation via Hierarchical Representation Learning on Heterogeneous Graph”, in The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, 2018, pages 635–644.

[42]Bird, S., Dale, R., Dorr, B., Gibson, B., Joseph, M., Kan, M.-Y., Lee, D., Powley, B., Radev, D., Tan, Y. F., “The ACL Anthology Reference Corpus: A Reference Dataset for Bibliographic Research in Computational Linguistics”, in Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC’08). Marrakech, Morocco: European Language Resources Association (ELRA), May 2008.

[43]Councill, I. G., Giles, C. L., Kan, M.-Y., “Parscit: an Open-source CRF Reference String Parsing Package.”, in LREC, Vol. 8, 2008, pages 661–667.

[44]Giles, C. L., Bollacker, K. D., Lawrence, S., “Citeseer: an Automatic Citation Indexing System”, in Proceedings of the Third ACM Conference on Digital Libraries. ACM, 1998, pages 89–98.

[45]Weis, M., Naumann, F., Brosy, F., “A Duplicate Detection Benchmark for XML (and Relational) Data”, in Proc. of Workshop on Information Quality for Information Systems (IQIS), 2006.

[46]Lo, K., Wang, L. L., Neumann, M., Kinney, R., Weld, D. S., “S2orc: The semantic scholar open research corpus”, in Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020, pages 4969–4983.

[47]Wadden, D., Lin, S., Lo, K., Wang, L. L., van Zuylen, M., Cohan, A., Hajishirzi, H., “Fact or fiction: Verifying scientific claims”, in Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2020, pages 7534–7550.

[48]Dasigi, P., Lo, K., Beltagy, I., Cohan, A., Smith, N. A., Gardner, M., “A dataset of information-seeking questions and answers anchored in research papers”, in Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2021, pages 4599–4610.

[49]Ingwersen, P., Larsen, B., “Using Citations for Ranking in Digital Libraries”, in Proceedings of the 6th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL’06). IEEE, 2006, pages 370–370.

[50]Larsen, B., “Exploiting Citation Overlaps for Information Retrieval: Generating a Boomerang Effect from the Network of Scientific Papers”, Scientometrics, Vol. 54, No. 2, 2002, pages 155–178.

[51]Meij, E., De Rijke, M., “Using Prior Information Derived from Citations in Literature Search”, in Large scale semantic access to content (text, image, video, and sound). LE CENTRE DE HAUTES ETUDES INTERNATIONALES D’INFORMATIQUE DOCUMENTAIRE, 2007, pages 665–670.

[52]Page, L., Brin, S., Motwani, R., Winograd, T., “The pagerank citation ranking: Bringing order to the web”, Stanford InfoLab, Tech. Rep., 1999.

[53]Ren, X., Liu, J., Yu, X., Khandelwal, U., Gu, Q., Wang, L., Han, J., “Cluscite: Effective Citation Recommendation by Information Network-based Clustering”, in Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2014, pages 821–830.

[54]Cohan, A., Feldman, S., Beltagy, I., Downey, D., Weld, D. S., “Specter: Document-level representation learning using citation-informed transformers”, in Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020, pages 2270–2282.

[55]Beltagy, I., Lo, K., Cohan, A., “Scibert: A pretrained language model for scientific text”, in Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 2019, pages 3615–3620.

[56]Ostendorff, M., Rethmeier, N., Augenstein, I., Gipp, B., Rehm, G., “Neighborhood Contrastive Learning for Scientific Document Representations with Citation Embeddings”, in The 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP 2022). Abu Dhabi: Association for Computational Linguistics, December 2022, 7-11 December 2022. Accepted for publication.

[57]Cohan, A., Ammar, W., van Zuylen, M., Cady, F., “Structural Scaffolds for Citation Intent Classification in Scientific Publications”, in Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), 2019, pages 3586–3596.

[58]Medi ć, Z., Šnajder, J., “Large-scale evaluation of transformer-based article encoders on the task of citation recommendation”, in Proceedings of the Third Workshop on Scholarly Document Processing. Gyeongju, Republic of Korea: Association for Computational Linguistics, Oct. 2022, pages 19–31, available at: https://aclanthology.org/2022.sdp-1.3

[59]Pelletier, F. J., “The Principle of Semantic Compositionality”, Topoi, Vol. 13, No. 1, 1994, pages 11–24.

[60]Sutskever, I., Vinyals, O., Le, Q. V., “Sequence to Sequence Learning with Neural Networks”, Advances in Neural Information Processing Systems, Vol. 27, 2014, pages 3104–3112.

[61]Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., Lang, K. J., “Phoneme Recognition using Time-Delay Neural Networks”, IEEE transactions on acoustics, speech, and signal processing, Vol. 37, No. 3, 1989, pages 328–339.

[62]Yang, L., Zhang, Z., Cai, X., Dai, T., “Attention-Based Personalized Encoder-Decoder Model for Local Citation Recommendation”, Computational Intelligence and Neuroscience, Vol. 2019, 2019.

[63]Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.-A., "Extracting and Composing Robust Features with Denoising Autoencoders", in Proceedings of the 25th International Conference on Machine Learning, 2008, pages 1096–1103.

[64]Medi ć, Z., Šnajder, J., "Improved local citation recommendation based on context enhanced with global information", in Proceedings of the First Workshop on Scholarly Document Processing. Online: Association for Computational Linguistics, Nov. 2020, pages 97–103, available at: https://aclanthology.org/2020.sdp-1.11

[65]Rokach, L., Mitra, P., Kataria, S., Huang, W., Giles, L., "A Supervised Learning Method for Context-Aware Citation Recommendation in a Large Corpus", in Proceedings of the Large-Scale and Distributed Systems for Information Retrieval Workshop, (LSDSIR'13). Citeseer, 2013, pages 17–22.

[66]Livne, A., Gokuladas, V., Teevan, J., Dumais, S. T., Adar, E., "CiteSight: Supporting Contextual Citation Recommendation using Differential Search", in Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval, 2014, pages 807–816.

[67]Storkey, A., "When Training and Test Sets are Different: Characterizing Learning Transfer", Dataset shift in machine learning, Vol. 30, 2009, pages 3–28.

[68]Quionero-Candela, J., Sugiyama, M., Schwaighofer, A., Lawrence, N. D., Dataset Shift in Machine Learning. The MIT Press, 2009.

[69]Moreno-Torres, J. G., Raeder, T., Alaiz-Rodríguez, R., Chawla, N. V., Herrera, F., "A Unifying View on Dataset Shift in Classification", Pattern recognition, Vol. 45, No. 1, 2012, pages 521–530.

[70]Schroff, F., Kalenichenko, D., Philbin, J., "Facenet: A Unified Embedding for Face Recognition and Clustering", in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pages 815–823.

[71]Li, J., Tao, C., Feng, Y., Zhao, D., Yan, R. *et al.*, "Sampling Matters! An Empirical Study of Negative Sampling Strategies for Learning of Matching Models in Retrieval-Based Dialogue Systems", in Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 2019, pages 1291–1296.

[72]Medi ć, Z., Šnajder, J., "An empirical study of the design choices for local citation recommendation systems", Expert Systems with Applications, Vol. 200, 2022, page 116852.

[73]Li, X., Mandal, B., Ouyang, J., “CORWA: A citation-oriented related work annotation dataset”, in Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Seattle, United States: Association for Computational Linguistics, Jul. 2022, pages 5426–5440, available at: https://aclanthology.org/2022.naacl-main.397

[74]Zhang, Y., Ma, Q., “Recommending multiple positive citations for manuscript via content-dependent modeling and multi-positive triplet”, in IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, 2021, pages 629–634.

[75]Mysore, S., Cohan, A., Hope, T., “Multi-vector models with textual guidance for fine-grained scientific document similarity”, in Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Seattle, United States: Association for Computational Linguistics, Jul. 2022, pages 4453–4470, available at: https://aclanthology.org/2022.naacl-main.331

[76]Reimers, N., Gurevych, I., “Sentence-BERT: Sentence embeddings using Siamese BERT-networks”, in Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pages 3982–3992, available at: https://aclanthology.org/D19-1410

[77]Lee, K., Chang, M.-W., Toutanova, K., “Latent retrieval for weakly supervised open domain question answering”, arXiv preprint arXiv:1906.00300, 2019.

[78]Guu, K., Lee, K., Tung, Z., Pasupat, P., Chang, M., “Retrieval augmented language model pre-training”, in International conference on machine learning. PMLR, 2020, pages 3929–3938.

[79]Lauscher, A., Ko, B., Kuehl, B., Johnson, S., Cohan, A., Jurgens, D., Lo, K., “Multicite: Modeling realistic citations requires moving beyond the single-sentence single-label setting”, in Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2022, pages 1875–1889.

[80]Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A. *et al.*, “Language models are few-shot learners”, Advances in neural information processing systems, Vol. 33, 2020, pages 1877–1901.

[81]Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X. V. *et al.*, "Opt: Open pre-trained transformer language models", arXiv preprint arXiv:2205.01068, 2022.

[82]Scao, T. L., Fan, A., Akiki, C., Pavlick, E., Ili ć, S., Hesslow, D., Castagné, R., Luccioni, A. S., Yvon, F., Gallé, M. *et al.*, "Bloom: A 176b-parameter open-access multilingual language model", arXiv preprint arXiv:2211.05100, 2022.

[83]Taylor, R., Kardas, M., Cucurull, G., Scialom, T., Hartshorn, A., Saravia, E., Poulton, A., Kerkez, V., Stojnic, R., "Galactica: A large language model for science", arXiv preprint arXiv:2211.09085, 2022.

[84]Zhang, Y., Sun, S., Gao, X., Fang, Y., Brockett, C., Galley, M., Gao, J., Dolan, B., "Retgen: A joint framework for retrieval and grounded text generation modeling", in Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 36, No. 10, 2022, pages 11 739–11 747.

[85]Bornmann, L., Daniel, H.-D., "What do citation counts measure? a review of studies on citing behavior", Journal of documentation, 2008.

List of Figures

1.1. Illustration of the differences between variants of CR tasks. On the left side of the figure is the first page of the article "*Language Models are Unsupervised Multitask Learners*" by Radford et al. When a user is either writing the article or only looking at it, a CR system could provide the citation recommendation based on the article's content. For example, if a user is looking for general recommendations for the article at hand, a GCR model takes as input the article's title and abstract (in yellow) and might recommend the articles in the yellow box as suitable for citing. In another scenario, a user might highlight only a sentence or two from the article's text (in red), and an LCR model would recommend articles in the red box for citing. Finally, for the task we propose in the thesis, PCR, a corresponding model takes the paragraph's topic sentence as input (in green) and outputs a list of recommended articles as citations in the rest of the paragraph. 4

2.1. An illustration of the differences between cross-encoder and bi-encoder design in text ranking. In the cross-encoder, shown in Subfigure 2.1a, terms from the query and the document are typically independently encoded until a certain point in the model, when they start to be combined in order to produce the final recommendation score. Here the combination of the term encodings from the query and the document happens inside the "Relevance score calculation module". On the other hand, in the bi-encoder design, shown in Subfigure 2.1b, the query and the document are independently encoded into two encodings, which are then processed to produce a single relevance score. The processing typically means calculating the cosine similarity or L2 distance between the two embeddings. 10

2.2. An illustration of a change in the arrangement of the query, positive, and a negative example before and after the training with the triplet loss. Before the training step, a negative example is closer to the query than the positive one, resulting in a non-zero triplet loss value when calculated on such a triplet. After training and updating the parameters, the model "pushes" the positive example closer to the query and a negative example further away. 11

2.3. An illustration of a two-stage pipeline model containing a prefiltering model in the first stage and the reranking model in the second stage. Icons of documents in red represent irrelevant documents for a given query, while the green icon represents the relevant document. Both models in the pipeline output a list of documents sorted by the relevance scores obtained by each model. However, the pools of the documents each model operates over differ. While the prefiltering model operates over a complete pool of articles, the reranking model produces only those articles produced in the output of the prefiltering model, i.e., a subset of the complete pool of articles. 12

2.4. An illustration of a transformer encoder module described in [36]. The input text is first tokenized and encoded with the tokens' embeddings, which are combined with positional embeddings and, as such, passed to the layer's first submodule, i.e., a multi-head attention layer. The output of the multi-head attention layer is summed with its input (residual connection) and passed through a layer normalization module to produce the output of the first submodule. This output is passed to the multi-layer perceptron module, which is again followed by a residual connection and another layer normalization module. A transformer encoder consists of N such layers. 15

3.1. An illustration of the difference between GCR and LCR tasks. While the input to GCR is typically an article's title and abstract, the input to LCR in most cases consists of only the citation context, i.e., a text snippet from the article's body, in which a citation marker is masked with a placeholder (in this case, "`[TARGET_CITATION]`"). Models for both tasks produce a list of articles sorted by the relevance scores obtained by the model, where the goal during the training of these models is to rank the articles cited in the query (in green) higher than others (in red). The article titles printed in **bold** represent those that were indeed cited in the given article or context. 20

3.2. An overview of the steps in the recommendation process when using ClusCite. First, each query manuscript is matched with an interest group, while in the second step, articles from the selected interest groups are scored for relevance for a given query based on the metadata features. Figure taken from [53]. . . .24

3.3. An overview of a GCR model from [16]. In the NNSelect module, embeddings for all seven documents d are in a shared document embedding space. The K nearest neighbors for a query are detected and passed as input to the NNRank module. NNRank reranks these K articles using a reranker model that produces a recommendation score for each article. Figure taken from [16]. 25
3.4. An overview of SPECTER's training process [54]. The three articles (query, related, and unrelated) are passed as input to the same transformer model, which outputs an embedding for each article. The loss is calculated as triplet loss using L2 distance from both the related and unrelated articles to the query. Figure taken from [54]. 26
3.5. An overview of the NCN model proposed by [18]. Figure taken from [18]. . . .28

4.1. The text module of the proposed LCR model. All the text sequences are passed as input to the same layers. The difference in the processing of each input is in the definition of the attention query (arrows indicate which embedding is passed as a query in each input). The green oval corresponds to the model variant in which global information is not included in the citation context representation. .31
4.2. The architecture of the bibliographic module of the proposed LCR model. Author names are extracted from the candidate article and matched with their corresponding embeddings. These are then passed through a convolutional layer and concatenated with a citations vector, consisting of the number of candidate article's citations overall and throughout the last n years. The final score is produced by passing the concatenation through a non-linear layer. 31

5.1. Recall @ k for different values of k for combinations of the two datasets and the two prefiltering models. 40
5.2. Construction of hard and easy negatives sets using four different negative sampling strategies. The input is a citing article with context c. The arrow labels indicate the sampling strategy: (1) prefiltered, (2) graph neighbors, (3) most citations, and (4) cited. Red ovals correspond to the hard negatives set (A_h^-), while yellow ovals correspond to the easy negatives set (A_e^-) for a given strategy.43

6.1. Illustration of an input to a PCR model and the generated output. As the input, we propose combining a topic sentence from an RW paragraph with the article's title and abstract. During the training, the model is tasked to output a list of recommendations where the articles cited in the paragraph (green icons) should be ranked higher than those not cited in the paragraph (red icons). 49

6.2. Illustration of a change in the arrangement of the query, two positive examples, and a negative example before and after the training with the quadruplet loss. Before the training step, a negative example is closer to the query than the positive ones are, resulting in a non-zero value of the quadruplet loss. After training and updating the parameters, the model "pushes" the positive examples closer to the query and the negative example further away while also enforcing that the positive examples are closer to each other than each of them is closer to the negative example. 52
6.3. Average ranks of cited articles per publication year for the ranking obtained with QPCR. 54

List of Tables

4.1. Number of contexts and articles in each dataset's train, validation, and test splits, alongside the total number of articles in each dataset. 34
4.2. Results on the RefSeer and the two ACL-ARC dataset variants for baselines and variants of our proposed model. ** and * indicate a statistically significant difference between TEXTBIBENH-WS and TEXTBIBCON-WS for p<0.05 and p<0.01, respectively (two-sided t-test for MRR and two-proportion z-test for R@10). 35

5.1. Performance of BM25 as a prefiltering model in terms of $R@1000$ on the validation sets of the ACL-ARC and RefSeer datasets for different b and k_1 parameter values. Values in **bold** indicate the best scores. Underlined values indicate the performance obtained when using the default parameter values. 38
5.2. Performance of SPECTER as the prefiltering model in terms of $R@1000$ on the validation sets of the ACL-ARC and RefSeer datasets for different input variants. Values in **bold** indicate the performance for the input variant that obtained the best score. 38
5.3. Performance of BM25 and SPECTER in terms of $R@1000$, $R@10$, and MRR on the test sets of ACL-ARC and RefSeer datasets. 39
5.4. Results in terms of $R@10$ and MRR of the reranking model variants TEXTENH and TEXTBIBENH-WS on the ACL-ARC and RefSeer datasets for the standard and strict training regimes. **Bold** values indicate the best-performing training regime for a reranking model variant and a combination of a dataset and a prefiltering model. 42
5.5. Results in terms of $R@10$ and MRR of different negative sampling strategies used with a combination of a dataset, prefiltering, and a reranking model. TBENH-WS corresponds to TEXTBBIBENH-WS. **Bold** values indicate the best-performing strategy for a given combination of a dataset, prefiltering model, reranking model, and metric. Strategies with the "(only)" suffix correspond to sampling strategy variants trained with only a hard negative set. 45

5.6. Results in terms of *R*@10 and *MRR* of different negative sampling strategies used with a combination of a dataset, prefiltering, and a reranking model when trained using a strict training regime. TBENH-WS corresponds to TEXTBBIBENH-WS. **Bold** values indicate the best-performing strategy for a given combination of a dataset, prefiltering model, reranking model, and metric. For brevity, we only report the best-performing strategies. 46

6.1. Examples of query instances from the dataset we constructed for the PCR task. Queries are composed by concatenating the citing article's title, abstract, and topic sentence from a paragraph. 50

6.2. Performance of the evaluated PCR models on the test set queries. The results of the best-performing model for each metric are shown in **bold**. 53

6.3. An example of a paragraph that cites an article that is older than the citing. Citation marker in the paragraph is printed inolive, as well as the title of the cited article. Although an experienced researcher might conclude from the title and abstract alone that the cited article might be suitable for citing in recent articles on commonsense reasoning, the vocabulary mismatch between the articles might introduce difficulties for CR models. 55

6.4. Another example of an article that cites an older work. Similarly as in Table 6.3, an experienced researcher might find the cited article fairly relevant as a classical method for extractive summarization, but CR models might struggle with vocabulary mismatch (e.g., "model" in citing is a "machine" in the cited article, "document" is an "article", and "summary" is an "excerpt"). 56

6.5. An example of recommendations for the paragraph describing work on dialect identification in other languages. None of the recommendations of SCINCL-TS and QPCR was cited in the paragraph, although some of the articles could fit into the paragraph's narrative. 57

6.6. An example of recommendations for the paragraph describing work in other languages on the construction of word pairs similarity datasets. None of the recommendations of SCINCL-TS and QPCR were cited in the paragraph, although some of the articles could fit into the paragraph's narrative. 58

6.7. An example of recommendations for the paragraph describing work on POS tagging for endangered languages. Similar to Table 6.6, none of the recommendations of SCINCL-TS and QPCR were cited in the paragraph, although some of the articles could fit into the paragraph's narrative. 59

6.8. An example of recommendations demonstrating high topical relevancy of recommended titles for the topic sentence in the case of QPCR. While SciNCL-TS's recommendations are more focused on dialog systems, as that is the main topic of the citing article, QPCR's recommendations are more about universal dependencies, which are mentioned in the topic sentence. 60
6.9. Another example of recommendations that demonstrate high topical relevancy of recommended titles for the topic sentence in the case of QPCR. SciNCL-TS's recommendations dominantly cover topics of argumentation mining and machine translation, while QPCR's recommendations are about the cross-lingual transfer, i.e., focused on the topic of the topic sentence. 61

Biography

Zoran Medić was born on November 9, 1992, in Metković, Croatia. He received his B.Sc. in Computing from the University of Zagreb, Faculty of Electrical Engineering and Computing in 2014 and M.Sc. in Computer Science from the same university in 2016. From July 2016 until July 2017, he was employed as a project associate at the Department of Electronics, Microelectronics, Computer and Intelligent Systems at the Faculty of Electrical Engineering and Computing. From October 2017 until March 2018, he worked as a business intelligence consultant at Adacta in Zagreb. From March 2018 until December 2018, he was employed as a data scientist at Realnetworks in Zagreb. Since January 2019, he has been a teaching and research assistant at the Department of Electronics, Microelectronics, Computer and Intelligent Systems at the Faculty of Electrical Engineering and Computing. In September 2022, he was on a research visit to Julius-Maximilians University of Würzburg. His research interests include natural language processing, machine learning, and information retrieval, focusing on learning text representations for dense retrieval.

List of publications

Journal publications

1.Medi ć, Z., Šnajder, J. “An empirical study of the design choices for local citation recommendation systems”, Expert Systems with Applications, Volume 200, August 2022, 116852.

2.Medi ć, Z., Šnajder, J. “A survey of citation recommendation tasks and methods”, Journal of computing and information technology, Volume 28(3), 2020, 183-205.

Conference publications

1.Medi ć, Z., Šnajder, J. “Large-scale Evaluation of Transformer-based Article Encoders on the Task of Citation Recommendation”, Proceedings of the Third Workshop on Scholarly Document Processing, October 2022, pp. 19-31.

2.Medi ć, Z., Šnajder, J. “Improved Local Citation Recommendation Based on Context Enhanced with Global Information”, Proceedings of the First Workshop on Scholarly Document Processing, November 2020, pp. 97-103.

3.Gombar, P., Medi ć, Z., Alagić, D., Šnajder, J. “Debunking Sentiment Lexicons: A Case of Domain-specific Sentiment Classification for Croatian”, Proceedings of the 6th Workshop on Balto-Slavic Natural Language Processing, April 2017, pp. 54-59.

4.Lozi ć, D., Šarić, D., Tokić, I., Medić, Z., Šnajder, J. “TakeLab at SemEval-2017 Task 4: Recent Deaths and the Power of Nostalgia in Sentiment Analysis in Twitter”, Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017), August 2017, pp. 784-789.

5.Medi ć, Z., Šnajder, J., Padó, S. “Does Free Word Order Hurt? Assessing the Practical Lexical Function Model for Croatian”, Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (SEM 2017), August 2017, pp. 115-120.

Životopis

Zoran Medić rođen je 9. studenoga 1992. u Metkoviću, Hrvatska. Preddiplomski sveučilišni studij računarstva završio je 2014. godine na Fakultetu elektrotehnike i računarstva Sveučilišta u Zagrebu, na kojem je i 2016. godine završio diplomski sveučilišni studij računarske znanosti. Od srpnja 2016. godine do srpnja 2017. godine zaposlen je kao projektni suradnik na Zavodu za elektroniku, mikroelektroniku, računalne i inteligentne sustave Fakulteta elektrotehnike i računarstva Sveučilišta u Zagrebu. Od listopada 2017. godine do ožujka 2018. godine zaposlen je kao konzultant iz područja poslovne inteligencije u tvrtki Adacta u Zagrebu. Od ožujka 2018. do prosinca 2018. godine zaposlen je kao podatkovni znanstvenik u tvrtki Realnetworks u Zagrebu. Od siječnja 2019. zaposlen je kao asistent i znanstveni suradnik na Zavodu za elektroniku, mikroelektroniku, računalne i inteligentne sustave Fakulteta elektrotehnike i računarstva Sveučilišta u Zagrebu. U rujnu 2023. godine boravio je kao gostujući znanstvenik na znanstvenom usavršavanju na Julius-Maximilians Sveučilištu u Würzburgu. Njegovi istraživački interesi uključuju obradu prirodnog jezika, strojno učenje, te pretraživanje informacija, s naglaskom na učenje reprezentacija teksta za gusto pretraživanje.