KEYPHRASIFICATION: SUMMARIZING TEXT INTO KEYPHRASES - USING NEURAL LANGUAGE GENERATION METHODS

by

Rui Meng

B.S., Wuhan University, China, 2012

M.S., Wuhan University, China, 2015

Submitted to the Graduate Faculty of

the School of Computing and Information in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2024

UNIVERSITY OF PITTSBURGH

SCHOOL OF COMPUTING AND INFORMATION

This dissertation was presented

by

Rui Meng

It was defended on

April 2nd 2024

and approved by

Dr. Daqing He, School of Computing and Information, University of Pittsburgh

Dr. Peter Brusilovsky, School of Computing and Information, University of Pittsburgh

Dr. Paul Munro, School of Computing and Information, University of Pittsburgh

Dr. Cornelia Caragea, College of Engineering, The University of Illinois at Chicago

KEYPHRASIFICATION: SUMMARIZING TEXT INTO KEYPHRASES - USING NEURAL LANGUAGE GENERATION METHODS

Rui Meng, PhD

University of Pittsburgh, 2024

Keyphrases encapsulate the core information of a text, acting as effective tools for organizing and retrieving extensive data. Their utility spans various applications, including information retrieval, document classification, and automatic summarization. Given the cost and limitations of manual keyphrase assignment, there has been a growing interest in automating this process. Traditional approaches to keyphrase assignment are categorized into extraction, which involves selecting phrases directly from the text, and tagging, where pre-defined tags are applied. Both methods often fail to address the complexity of natural language. For instance, a substantial fraction of keyphrases are absent from the source text and are missed by extraction methods. This observation highlights the need to reevaluate the paradigms within keyphrase studies and refine methodologies in automatic keyphrase prediction.

This dissertation introduces **KeyPhrasification** to formulate the task of keyphrase prediction. By developing a conceptual framework and defining essential properties, this work aims to deepen the understanding of keyphrase prediction and facilitate the development of more effective techniques. Furthermore, I propose a novel modeling approach, keyphrase generation (**KpGen**), utilizing neural language generation to learn the mapping between texts and keyphrases directly from data to predict contextually relevant phrases in varied forms. The dissertation further presents various enhancements and mechanisms to refine this approach.

This work makes several pivotal contributions. It reformulates keyphrase prediction as a specialized form of summarization, thereby broadening the previous research scope. It innovates in automatic keyphrasification with a data-driven approach, employing neural networks to predict context-relevant phrases, overcoming the limitations of prior methodologies. Furthermore, the study explores a range of advanced language generation techniques, from basic to pre-trained and large language models, making it a comprehensive investigation into the task of keyphrasification.

Keywords: Keyphrase; Keyphrasification; Keyphrase Generation; Language Generation

TABLE OF CONTENTS

List of Tables

List of Figures

PREFACE

This dissertation is dedicated to my dear family –

Yu, Avery, Pocky, and my parents

– for their unwavering love and support.

Also a tribute to the magnificent world we are sharing and experiencing.

This dissertation marks the end of my Ph.D. journey.

Embarking on this research journey and pursuing a Ph.D. was a monumental decision I made during my master's program in Wuhan in 2013. At that time, I struggled to see the fun in research, lacking a clear sense of its purpose. Yet, it was during this time that I first encountered deep learning, sparked by a reading comprehension test that mentioned Geoffrey Hinton, and possibly AlexNet. This introduction ignited a passion that resonated with my reflections on intelligence and brain functionality, providing me with a clear purpose and passion for both research and life.

I express my deepest gratitude to Dr. Daqing He for his invaluable support and mentorship throughout this period. His teachings extended beyond research, and his trust has been fundamental to my growth. I am also deeply thankful to my committee members: Professor Peter Brusilovsky, my co-advisor, whose guidance and encouragement were essential to me (the NSF project gave birth to this study); Professor Paul Munro, who demystified neural networks for me (I will miss the reading group); and Professor Cornelia Caragea, whose pioneering research on keyphrase has been a source of inspiration (the productivity infographic was incredibly helpful). Their continuous support and insightful feedback have been pivotal in my doctoral study.

My heartfelt thanks go to Yu, my constant companion, who has been with me through every peak and valley. Having shared nearly half my life with you, I want to spend the rest of it by your side. I am also grateful to my parents for their unconditional love and encouragement in my pursuits.

My gratitude extends to everyone I have encountered during my Ph.D. studies, as each interaction has shaped me into the person I am. Special thanks to Sanqiang Zhao, Shuguang Han, Xingdi Yuan, and Tong Wang for their collaboration on numerous thrilling projects. I am also grateful for my friends in Pittsburgh, including but not limited to Sungmin Kim, Danchen Zhang, Ning Zou, Tonghao Han, Sunny Xiao, Zhendong Wang, Runhua Xu, Jinlai Xu, Zhimeng Luo, Xin Liu, Mendi Wang, Daniel Steinberg, Chun-Hua Tsai, Shih-Yi Chien, Ke Zhang.

The memories of time spent in Pittsburgh, at the universities, and in the coffee room on the 5th floor will always be cherished.

Let the journey begin.

Thank you,

Rui Meng, Lexington, KY

1.0 INTRODUCTION

A keyphrase, also referred to as a keyword, encapsulates the essential information of a piece of data, such as a longer text, in a succinct manner. Throughout this dissertation, I use the terms "keyphrase" and "keyword" interchangeably; however, the former is preferred as it implies the potential inclusion of multiple words. High-quality keyphrases play a crucial role in facilitating the understanding, organization, and retrieval of data. They find widespread applications in various domains, such as information retrieval [92, 89, 223], text summarization [238, 46, 207], text categorization [87], opinion mining [14] and online advertising [228].

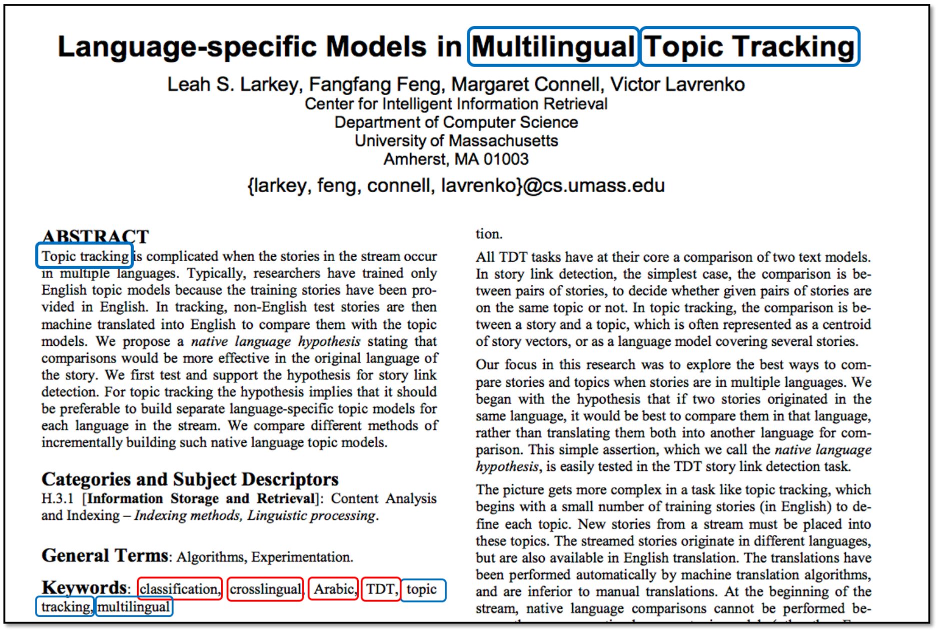
Despite the numerous automatic keyphrase techniques developed over the past decades, a significant performance gap remains between these systems and human capabilities. In my perspective, several inherent shortcomings of current methodologies hinder further improvement.

This dissertation aims to address the challenge of keyphrasification, a concept encompassing various keyphrase-related studies. Additionally, in response to the inherent limitations of traditional automatic methods, I propose a novel paradigm that treats keyphrasification as a form of language generation.

As the first chapter of the dissertation, this section provides a general introduction to the study. It delves into the research motivation and outlines several core research questions.

1.1 PROBLEM STATEMENT

Keyphrases provide a concise overview of rich content and find utility across a wide range of applications. Examples of keyphrases extracted from a scientific paper and a question posting on StackExchange are depicted in Figure 1 and 2. In these examples, authors and question askers employ several phrases to underline the core concepts or high-level topics discussed in the text. This allows readers to quickly grasp the essence of the content while enabling system developers to construct document indices and term hierarchies using these keyphrases. Academic publications and websites featuring social tagging, such as StackExchange and Del.icio.us, serve as common resources for constructing datasets for keyphrase studies due to the availability of public data.



Language-specific Models in Multilingual Topic Tracking

Leah S. Larkey, Fangfang Feng, Margaret Connell, Victor Lavrenko
Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts
Amherst, MA 01003
{larkey, feng, connell, lavrenko}@cs.umass.edu

ABSTRACT

Topic tracking is complicated when the stories in the stream occur in multiple languages. Typically, researchers have trained only English topic models because the training stories have been provided in English. In tracking, non-English test stories are then machine translated into English to compare them with the topic models. We propose a *native language hypothesis* stating that comparisons would be more effective in the original language of the story. We first test and support the hypothesis for story link detection. For topic tracking the hypothesis implies that it should be preferable to build separate language-specific topic models for each language in the stream. We compare different methods of incrementally building such native language topic models.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing – *Indexing methods, Linguistic processing*.

General Terms: Algorithms, Experimentation.

Keywords: classification, crosslingual, Arabic, TDT, topic tracking, multilingual

tion.

All TDT tasks have at their core a comparison of two text models. In story link detection, the simplest case, the comparison is between pairs of stories, to decide whether given pairs of stories are on the same topic or not. In topic tracking, the comparison is between a story and a topic, which is often represented as a centroid of story vectors, or as a language model covering several stories.

Our focus in this research was to explore the best ways to compare stories and topics when stories are in multiple languages. We began with the hypothesis that if two stories originated in the same language, it would be best to compare them in that language, rather than translating them both into another language for comparison. This simple assertion, which we call the *native language hypothesis*, is easily tested in the TDT story link detection task.

The picture gets more complex in a task like topic tracking, which begins with a small number of training stories (in English) to define each topic. New stories from a stream must be placed into these topics. The streamed stories originate in different languages, but are also available in English translation. The translations have been performed automatically by machine translation algorithms, and are inferior to manual translations. At the beginning of the stream, native language comparisons cannot be performed be-

Figure 1: A scientific publication with author-annotated keyphrases. Blue boxes denote the enclosed keyphrases are present in the source text (title and abstract) and red boxes denote absent keyphrases.

Keyphrases are not inherently available and acquiring them through human annotation can be costly and inefficient. Consequently, techniques for automating the process of keyphrase extraction have been actively researched for decades. Two primary strands of automatic methods have garnered extensive attention: extracting keyphrases directly from the original content [115, 127, 216], and automatically assigning tags from a pre-defined tag vocabulary [175, 102, 182, 181].

Keypharase Extraction aims to identify the most important or representative phrases from a given source text. This provides models with greater flexibility in selecting appropriate words and phrases to summarize a text, which is particularly crucial for text-based applications where nuanced information is necessary to distinguish one document from others.

Most existing keyphrase extraction algorithms address this problem through two main steps [115, 193]. The first step involves acquiring a list of keyphrase candidates, with researchers exploring the use of n-grams or noun phrases with specific part-of-speech patterns to identify potential candidates [85, 104, 114, 209].

The second step entails ranking these candidates based on their importance to the document, either through supervised or unsupervised machine learning methods utilizing manually-defined features [57, 115, 114, 96, 125, 139, 178, 216] or learned representations [150, 120, 221]. Alternatively, some researchers propose methods such as sequence labeling to directly output potential keyphrases without following the explicit two-step procedure [233, 152, 2, 166, 239, 69, 212, 167].

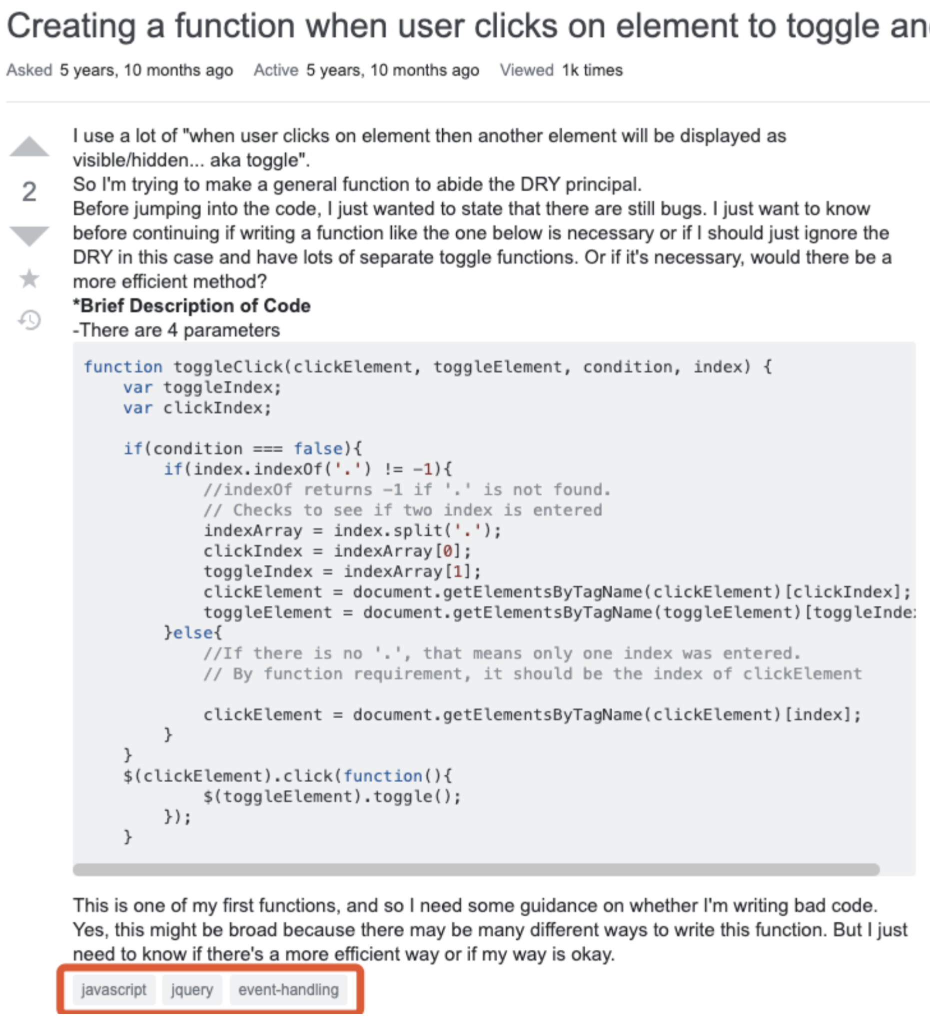


Figure 2: A question posting on StackExchange with user-annotated labels.

Alternatively, *Automatic Tagging* or *Tag Recommendation* refers to the automated process of suggesting useful tags for objects such as documents, images, or videos [181]. Popular websites like Del.icio.us, Flickr, and StackExchange employ tagging systems to enable users to organize content. However, automatic tagging faces inherent limitations, notably the inflexibility of tag vocabulary, which is typically constructed based on existing user annotations and remains fixed. Moreover, tags are generally used to describe high-level topics rather than details. Consequently, *Automatic Tagging* is often implemented as a fine-grained multi-label classification system.

While automatic tagging shares similarities with keyphrase extraction and the focus of this study, there are notable differences in technical solutions, objectives, and applications. Therefore, this dissertation does not delve extensively into automatic tagging and does not include comparisons with tagging models.

While studies on *Keyphrase Extraction* and *Automatic Tagging* have achieved certain success and demonstrated considerable value in real-world applications, their modeling paradigms are beset by several major limitations [39, 128].

1. Both *Keyphrase Extraction* and *Automatic Tagging* methods are significantly hampered by their output limitations, which inherently do not align with how humans assign keyphrases.
    a. Concerning *Keyphrase Extraction*, all existing methods can only identify phrases present in the source text, thus failing to capture instances such as synonymous phrases representing key concepts, phrases describing overarching topics, or phrases with different word orders. To further substantiate the argument regarding the output limitations of *Keyphrase Extraction*, I examine the proportion of phrases that do not match any sub-sequence of the source text verbatim (referred to as absent keyphrases) across four widely-used keyphrase datasets constructed from academic publications, as shown in Table 1. The table highlights substantial portions of absent keyphrases across all datasets. Even in datasets like NUS and SEMEVAL, where fulltext is considered, nearly 10% of phrases cannot be located. These absent phrases cannot be predicted using any extractive approaches, underscoring the imperative for novel methodologies in keyphrase tasks and more robust models.
    b. Regarding *Automatic Tagging*, which operates in the opposite direction, its output tags tend to be high-level and abstract, often overlooking the details of the content. Additionally, it is significantly constrained by the predefined tag vocabulary, which is fixed and limited in terms of the number of tags available.
2. From a machine learning perspective, both methods simplify the underlying problem in distinct ways. For instance, a common approach in keyphrase extraction involves converting the keyphrase prediction task into binary classification or ranking problems based on n-grams

Table 1: Statistics on absent keyphrases in four academic datasets

Dataset	# Document	# Keyphrase	% Absent
Inspec	500	4,913	21.5%
Krapivin	460	2,641	43.8%
NUS	211	2,461	12.3%
SemEval	100	1,507	8.9%
KP20k	19,987	105,181	36.69%

within a text. In doing so, models focus on identifying features that differentiate these n-grams rather than understanding why certain n-grams are deemed more significant or phrase-like than others. For instance, when ranking phrase candidates, traditional extractive methods often rely on statistical features such as TF-IDF and PageRank. However, these features only reflect the importance of each word based on word occurrence statistics and do not capture the semantic meaning underlying the text. Similarly, in automatic tagging, classification models are known to prioritize capturing relationships between trivial features and class labels to optimize cost functions [66], rather than comprehensively understanding the internal structure of the data.

1.2 MOTIVATIONS AND RESEARCH QUESTIONS

The preceding section provides an overview of two primary research directions in automatic keyphrase generation: keyphrase extraction and automatic tagging. It also outlines the fundamental challenges associated with these approaches. Before delving into the motivation behind this research, it is beneficial to illustrate how and why humans assign keyphrases to a given piece of text, providing a comprehensive understanding of this task. For instance, academic authors are often asked with providing keyword information for their publications. By providing precise and comprehensive keyphrases, they increase the chances of potential readers finding their work via retrieval systems. Given their role as the primary intellectual contributors to their publications, authors possess an intimate familiarity with the content, rendering them more qualified than third-party annotators for such intellectual tasks. Moreover, authors commonly assign keyphrases based on their semantic understanding of the content, selecting representative phrases from the text and abstracting appropriate high-level concepts and ideas.

To the best of my knowledge, there is no existing term that accurately encapsulates the general process or action of extracting keyphrases from rich content. Therefore, I propose the introduction of a new term – **KeyPhrasification**– to describe the series of studies focused on keyphrase prediction. This term is not only independent of specific techniques used for keyphrase prediction but also expands the conceptual scope of existing research areas such as keyphrase extraction and automatic tagging. To illustrate, I delineate the scope addressed by each major automatic paradigm using a Venn diagram in Figure 3. The term **KeyPhrasification** is partly inspired by summarization, as it shares many similarities with keyphrasification. This study aims to define, formulate and characterize **KeyPhrasification** from various perspectives, shedding light on research endeavors in this domain.

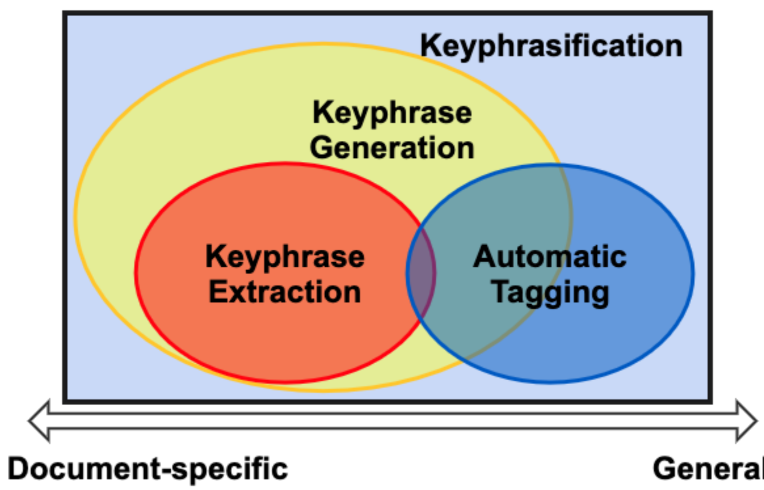


Figure 3: A Venn diagram illustrating the scope of keyphrases addressed by the three major paradigms in **KeyPhrasification**.

Moreover, I observe that a phrase, being a concise unit of natural language, inherently comprises a sequence of words. Thus, we can approach the task of predicting phrases as a language generation problem. Consequently, I introduce a method to automate keyphrasification using language generation techniques, termed **KpGen**. This innovative approach to modeling keyphrases enables prediction without strict dependence on source text or predefined tag vocabulary. Leveraging Sequence-to-Sequence Learning (**Seq2Seq**) and Deep Neural Networks, **KpGen** operates in a completely data-driven manner, capable of generating informative and natural-sounding phrases. Additionally, I propose several variants to enhance the performance of this method across different dimensions.

Last but not least, I explore the depth of understanding of keyphrases by large language models and explore the potential for conducting evaluations using LLMs.

Following the elucidation of my research motivation above, I will explore the following research questions in this dissertation, each of which is divided into several sub-questions.

1.2.1 RQ1: Defining and Understanding Keyphrasification

Currently, few studies delve into the formulation of keyphrasification and the elucidation of its key factors. To comprehensively understand keyphrasification, I will address the following three sub-questions for RQ1:

- RQ 1.1: How keyphrasification should be defined and formulated?
- RQ 1.2: What are the core properties of a keyphrase?
- RQ 1.3: How do humans understand, use and assign keyphrases?

1.2.2 RQ2: Keyphrasification by Language Generation

Moving beyond conventional keyphrase extraction and automatic tagging, I propose to investigate automatic keyphrasification using language generation methods. While deep neural networks have achieved remarkable success in various machine learning domains, including language generation tasks such as machine translation and summarization, several new questions emerge when applying these techniques to keyphrasification:

- RQ 2.1: Given that neural language generation has primarily been applied to produce single sequences, what distinct challenges does keyphrasification introduce, and which strategies could effectively tackle these issues?

- RQ 2.2: Since most existing sequence learning models aim to maximize the likelihood of generating a single sequence, I question if this learning objective remains ideal for keyphrase generation? Are there any approaches that could enhance this learning goal?
- RQ 2.3: Supervised learning methods typically demand considerable data and annotation labor. Can we identify approaches to keyphrase generation that diminish the dependency on extensively annotated datasets?
- RQ 2.4: Large Language Models (LLMs) have shown exceptional prowess in a variety of NLP tasks. In the context of keyphrasification, how do LLMs perform, and what potential do they exhibit in addressing keyphrase-centric challenges?

1.3 SCOPE DEFINITION

In this dissertation, I define the scope of my study with the following statements:

1. The focus of this study is on text-based keyphrasification, as it represents the most studied and widely applied form of this task. While there are studies on keyphrase prediction for multimedia sources such as image tagging [31] and audio tagging [37], these fall primarily within the realm of automatic tagging. Given the significant variations in techniques for processing different media types, this dissertation does not address keyphrase prediction for image, video, or audio data. However, these areas are potential avenues for future exploration. For instance, we may leverage data from image or video captioning studies for vision-based keyphrase prediction, and transcriptions from phone calls or meetings for audio-based keyphrase prediction.
2. This study primarily focuses on methods based on neural networks, although a wide range of other methods and models have been utilized in existing keyphrase prediction studies [115, 127, 216]. One of the main contributions of this work is the introduction of a novel paradigm that predicts phrases as language generation. Currently, neural networks are the most practical and effective backbone for language generation models. Additionally, recent studies have demonstrated significant improvements by replacing traditional keyphrase extraction or automatic tagging methods with neural network-based variants, owing to their superior ability to represent semantics and fit data.

3. This dissertation does not include automatic tagging methods for comparison, as they exhibit noticeable differences in methods, scope, and applications compared to most keyphrase prediction studies. While automatic tagging shares similarities with keyphrase prediction in assigning tags (which can be phrases) to indicate the main content, there are significant distinctions in technical solutions, goals, and applications. Automatic tagging is typically formulated and implemented as a multi-label classification problem, with a fixed and predefined tag vocabulary. This fixed vocabulary greatly limits its ability to predict phrases effectively. Moreover, automatic tagging is often applied for coarse-grained categorization rather than as a discriminative descriptor for individual data points. Consequently, comparing keyphrase prediction methods with automatic tagging methods under current evaluation settings is unnecessary.

1.4 OVERVIEW OF THE CHAPTER STRUCTURE

Chapter 3 provides an overview of related studies covering various topics pertinent to this dissertation. Chapter 3 delves into the formulation of keyphrasification and explores its essential properties. In Chapter 4, I introduce keyphrase generation, a novel paradigm for automatic keyphrasification, along with a comprehensive examination of critical method designs tailored for this task. Chapter 5 presents an enhanced keyphrase generation method, which seeks to integrate the properties introduced in Chapter 3. Chapter 6 introduces a new method aimed at reducing the reliance on annotated data. Chapter 7 discusses preliminary results concerning the utilization of large language models for keyphrase generation and understanding. Finally, Chapter 8 concludes this dissertation by summarizing its contributions and discussing potential future research directions.

2.0 RELATED WORK

In this chapter, I will provide a comprehensive review of related studies on keyphrasification, organized into three main paradigms aimed at automating this task: extraction, generation, and tagging. Additionally, I will delve into recent advancements in natural language generation, which is integral to the discussion in this dissertation. It's important to note that my focus will primarily be on studies utilizing deep neural networks, as they serve as the foundational technology for this work and have shown significant improvements compared to traditional machine learning methods. While there are several high-quality survey articles [78, 10, 144, 52, 151, 26, 136, 60] that organize and analyze keyphrasification studies in various ways, many of them do not prioritize covering the latest research in deep learning-based approaches.

2.1 KEYPHRASE EXTRACTION

2.1.1 Methods based on Candidate Selection

This line of methods is typically performed in two steps. Simply put, people could first collect a large volume of phrase candidates, then compute scores on them in various ways and return the top-ranking phrases.

The first step is to generate a list of phrase candidates with heuristic methods. One easy way is to take all the n-grams within a text as candidates [86, 71, 128, 221], which ensures a high recall possibility but also introduces many noises. Another way is to utilize Part-of-Speech or syntactic parsing techniques to extract all the noun phrases as candidates since most keyphrases play as noun phrases in the text [113, 205, 139, 104].

The second step is to score each candidate phrase for its likelihood of being a keyphrase in the given document. The top-ranked candidates are returned as keyphrases. Both supervised and unsupervised machine learning methods are widely employed here. For supervised methods, this task is solved as a binary classification problem, and various types of learning methods and features have been explored [57, 216, 85, 129, 117, 68]. Besides the classification paradigm, [113] modeled keyphrase extraction as a problem of translating from the language of documents to the language of keyphrases. To this end, they train a word alignment model that learns a translation from the documents to the keyphrases. This approach alleviates the problem of vocabulary gaps between source and target to a certain degree.

As for unsupervised approaches, primary ideas include finding the central nodes in text graph [139, 71], detecting representative phrases from topical clusters [115, 114], and so on. One interesting study [193] proposed to value the importance of a phrase by measuring its informativeness and phraseness, and this method is implemented with two n-gram language models.

Many recent studies based on deep learning also fall into this category. [221] ranked n-grams with a convolutional transformer model, which models both the text in the document and corresponding visual features, which visually strengthens the discrimination between keyphrases and normal content. Besides, they introduced a general-domain benchmark annotated on webpages which greatly extended the richness of testbeds, which were previously dominated by academic text.

BERT [42] has achieved great success in Natural Language Understanding as well as downstream applications. [142] was one of the first studies using BERT to rank noun phrases in a text. They also proposed to incorporate span-based features into learning, since phrases appear as consecutive words in a text. Similarly, [188] used the BERT [42] as the base document encoder to rank n-grams, combining a chunking module to identify high-quality phrases and a ranking module to assign a score for each phrase.

2.1.2 Methods based on Sequence Labeling

Sequence Labeling is an effective way of modeling keyphrase extraction problems, which assigns a label to each token in a source sequence, indicating whether each word should be part of a keyphrase or not.

On academic datasets, [118] used sequence labeling models to extract keyphrases from text and introduced a semi-supervised algorithm that propagates labels to leverage unannotated articles. [70] incorporated linguistic and document-structure information into sequence labeling. [212] discussed the issue of cross-domain keyphrase extraction, and they proposed learning a topicaware representation with the help of adversarial learning. [152, 2] examined the effectiveness of Conditional Random Fields and word embeddings trained on different corpora. [166] explored BiLSTM-CRF models with a wider range of text embeddings, and they found BERT [42] and SciBERT [11] models outperform the other representations.

Datasets built on social networks [124] are also widely used testbeds for Keyphrase Extraction. [233] proposed a joint-layer recurrent neural network model to extract keyphrases from tweets, which is one of the first applications of deep neural networks in the context of keyphrase extraction. [163] improved [233]'s work by incorporating stacked RNN models with word embeddings, POS tags, phonetics, and phonological features. Since re-tweets could provide contextual and historical information for current tweets, [234, 236] proposed to explicitly utilize the retweet history to improve keyphrase extraction. [235] proposed a mechanism to mimic human attention by analyzing eye-tracking corpus, and it was effectively integrated with keyphrase extraction models, for both supervised and unsupervised ones.

2.1.3 Other Methods

A special way of keyphrase extraction is to directly output the position of keyphrases in the source text. [185] used Pointer Networks to point to the start and end positions of keyphrases in a source text. Similarly, [190] also utilized a pointer network as the base model. However, instead of pointing to the position of keyphrases in a source text, this model points to the words to be outputted. Besides, they leveraged a graph convolutional network as a means to explicitly model the short- and long-term dependency between words.

2.2 KEYPHRASE GENERATION

The main drawback of keyphrase extraction is that oftentimes keyphrases can be absent from the source text, thus an extractive model will fail to predict those keyphrases [39]. [135] noticed this missing piece in existing studies and first proposed the CopyRNN to address this issue, a neural generation model that both generates words from vocabulary and points to words from the source text. Not only the CopyRNN can recall a significant amount of absent phrases, but also it outperforms traditional extractive methods consistently.

Following the CopyRNN architecture, there are several studies that improve keyphrase generation with enhanced model architectures. [237] leveraged a Convolutional sequence-to-sequence model to speed up the training and inference. [30] proposed to incorporate a coverage mechanism to help reduce duplication in generated results and improve coverage. [240] leveraged additional linguistic information in both inputs and outputs to facilitate the model learning, in order to reduce generating overlapping phrases. [38] proposed KPDrop, which drops out certain present keyphrases from the source text. The results with various training paradigms showed consistent improvements in absent keyphrase performance.

Besides, a few studies tried to improve the model beyond training with maximum likelihood estimation. [226] proposed semi-supervised methods by leveraging both labeled and unlabeled data for training. [34, 226] proposed to use structure information (e.g., title of source text) to improve keyphrase generation performance. [28] introduced reinforcement learning and adaptive rewards to train generation models. [32] retrieved similar documents from training data to help produce more accurate keyphrases.

The multiple target phrases pose a difficulty for sequence-to-sequence learning, and researchers seek for novel ways to deal with the multiple phrase decoding. [231] proposed to concatenate multiple keyphrases as a sequence for training, with which the model is expected to handle the correlation among phrases and avoid duplicate generation. Additionally, they propose two auxiliary modules to promote the diversity in generated phrases. Most generation models rely on Beam Search for propagating a large number of predictions, but it also leads to great waste in decoding due to duplicate generation. Regarding this issue, [141] used multiple decoders (each of them generates only one keyphrase) that focus on different words of the source text by subtracting the attention value derived from the previous decoder. [33] proposed an exclusive hierarchical decoding framework to prevent models from generating duplicate phrases.

A newly introduced training paradigm for keyphrase generation is One2Set [227, 220], in which target keyphrases are treated as a set instead of a concatenated sequence. Specifically, all keyphrases are predicted in parallel, or a semi-autoregressive manner, and the mapping between the generated and target tokens is dynamically aligned.

Pre-training models have shown great effectiveness on various NLP tasks. [111] utilized BERT as the base encoder and combined extractive and generative losses in training, which led to impressive improvement on various datasets. [217] conducted a comprehensive investigation of leveraging pre-trained SEQ2SEQ models for keyphrase generation and compared a set of effective decoding strategies. [219] leveraged an extractor-generator backbone based on UniLM [45] and showcased the possibility of unifying keyphrase extraction and generation.

A few studies focus on the keyphrase generation problem under the long context setting. [64] introduced an academic keyphrase dataset FullTextKP and demonstrated that extractive summary can serve as an efficient surrogate for the original full text. LDKP [119] is another corpus for long context keyphrase generation containing the full text of 1.3 million scientific articles. EUROPA [168] is a multilingual keyphrase dataset derived from legal judgments, covering 24 EU official languages.

To reduce the need for annotated data for training models, [65] examined four ways of augmentation for keyphrase generation (dropout, back-translation, synonym replacement and a keyphrase-specific variant) and showed their effectiveness for KPG in low-resource settings. [218] proposed multiple pertaining objectives to pretrain a generation model and it exhibited decent performance with limited training data. [133] proposed pertaining KPG models using Wikipedia, a general-domain phrase corpus, and they generalized to unseen domains with limited training data.

2.3 AUTOMATIC TAGGING

Automatic tagging [37, 128, 12], also called *Tag Recommendation* [181, 122] or *Social Tag Prediction* [80], refers to the automatic process assigning appropriate tags to describe an object, e.g., video, text and audio. Since tags provide a cheap and flexible way of organizing content than taxonomy, it has been widely used in different applications, such as image tagging [31, 192], audio tagging [37, 31, 157] and text tagging [182, 102, 102]. Broadly speaking, assigning keyphrases in scientific publications could also be considered as social tagging. But, in my opinion, the difference between automatic tagging and keyphrase extraction can be seen in two ways. First, tag is usually considered atomic, i.e. as the base unit, in automatic tagging studies. Therefore they do not pay much attention to linguistic characteristics of tags. Second, in practice only a fixed number of tags are considered for modeling, since long-tail tags are of less quality and value [49]. Therefore multi-label classification becomes the primary method for automatic tagging, especially on non-textual contents.

For textual content, according to [181], user-centered approaches and document-centered approaches are the two major ways to address tag recommendation. The former aims at recommending tags to a user from similar users based on their user interests. Collaborative filtering is one typical user-centered technique. However, it is considered less effective due to the sparsity of tag annotation [49]. Document-centered approaches utilize the rich information contained in the documents. Therefore, keywords or keyphrases are often exploited as tag candidates, and techniques for rating [22, 50, 80] or ranking [27] tags can actually be shared between automatic tagging and keyphrase extraction studies [12]. But again, the typical setting of automatic tagging only considers a fixed tag vocabulary, which is significantly different from the goal of general keyphrase studies.

To the best of my knowledge, [127] is the only study discussing the connection between the two areas. In their study, collaboratively tagged documents from CiteULike were leveraged in evaluation, to counteract the subjectivity in the author keywords. They proposed using a supervised keyphrase extraction method for automatic tagging. However, though it alleviates the current evaluation issue to some extent, the rarity of social tags and the difficulty of collecting them prevent people from following this evaluation method.

2.4 NATURAL LANGUAGE GENERATION

In general, Natural Language Generation studies can be categorized into two classes, conditional generation, and unconditional generation, depending on whether a source text is given. The former is more widely researched and applied than the latter.

Conditional generation is to generate a target text given certain source information, and the relationship between the source and target is defined by the task. Many NLP tasks can be formulated as conditional generation, such as machine translation [35, 6, 173], text summarization [170, 165], and sentence simplification [51, 241]. The Keyphrase Generation proposed in this dissertation also falls into this category.

Due to the great efficacy of end-to-end training for deep neural networks, Sequence-to-Sequence Learning [36, 191] becomes the fundamental method for language generation. It is a special case of Encoder-Decoder models, in which an encoder module is leveraged to extract information from the source data and a decoder module is to express this information into the target format. A basic Sequence-to-Sequence model is illustrated in Figure 4.

Different strategies have been explored to improve the performance of Sequence-to-Sequence models. The attention mechanism [6] is a soft alignment approach that allows the model to automatically locate the relevant input components. To make use of the important information in the source text, some studies sought ways to copy certain parts of content from the source text and paste them into the target text [1, 73, 232]. A discrepancy exists between the optimizing objective during training and the metrics during evaluation. A few studies attempted to eliminate this discrepancy by incorporating new training algorithms [121] or by modifying the optimizing objectives [173].

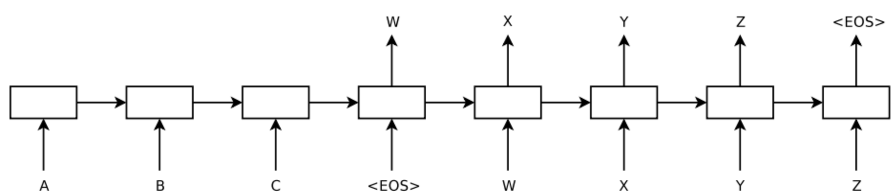


Figure 4: An illustration of a basic Sequence-to-Sequence architecture. The encoding process is placed on the left side. Three words, A, B, and C, denote the source text and `<EOS>` in the middle denotes the end of the source sequence. The right side shows the decoding process. The model outputs a sequence conditioned on the source information and the ending `<EOS>` denotes the end of decoding/generation.

As for unconditional generation models, the generated text is text-free and does not condition on any source information. Generally, an unconditional generation model cannot be directly applied to a specific task without any modification or parameter tuning, but it can provide a probability distribution of the next word and this can be leveraged in different ways, say improving language readability and fluency [109] or facilitate text generation for few-shot learning [230, 9].

Unconditional generation models typically generate text from left to right, which follows the natural way humans read or write. However, many models are also trained without following certain orders, resulting in a better understanding of contextual information. In terms of left-to-right unconditional models, GPT-2 [160], the successor of GPT [159], gained wide attention due to its superior performance. It is based on the Transformer architecture and trained with a large amount of task-free textual data. For bidirectional unconditional models, ELMo [154] utilized bidirectional LSTM and multiple objectives for language modeling. BERT [42] proposed a bidirectional Transformer architecture and it yields excellent performance on various NLP tasks. BERT might be utilized as a base text encoder in this dissertation.

Large language models (LLMs) have become increasingly prominent in the field of NLP, characterized by their billions of parameters and training on extensive datasets, leading to outstanding performance in a wide array of NLP tasks. These models have demonstrated emergent capabilities not previously seen [215, 214], surpassing traditional benchmarks in areas like question answering and summarization [23, 195]. Studies have benchmarked LLMs, indicating their capacity to match or exceed existing standards in various NLP domains [16, 103]. The integration of techniques such as instruction tuning [149, 40] and Reinforcement Learning from Human Feedback (RLHF) [149, 184] could further amplify the potential of LLMs. Instruction tuning, where models are fine-tuned to follow natural language instructions, can enhance the ability of LLMs to generate accurate and relevant outputs by providing clear, task-specific guidance. RLHF, which refines model outputs based on human feedback, could be instrumental in fine-tuning the generation process, ensuring that the produced outputs are not only precise but also contextually appropriate. The application of these advanced training methodologies could lead to significant improvements in the way LLMs understand and execute the keyphrasification task, pushing the boundaries of what is achievable with current language model technologies.

3.0 UNDERSTANDING KEYPHRASIFICATION

3.1 MOTIVATION

In Chapter 1, I propose to use a new term – **KeyPhrasification**– to describe the general process of obtaining keyphrases from rich content, such as lengthy documents, whether performed by humans or automated systems. This term is inspired by “keyword” [44] and “phrasify” [43], the latter referring to “to utter or express in a phrase”. Unlike specific techniques used for keyphrase extraction, keyphrasification extends the research scope beyond existing areas such as keyphrase extraction and automatic tagging, providing a more comprehensive framework for understanding keyphrase-related tasks.

To date, there has been limited theoretical discussion on the formalization of keyphrasification. In [193], the authors emphasize two crucial features: *phraseness* and *informativeness*. Phraseness characterizes the extent to which a sequence of words constitutes a phrase, while informativeness measures how effectively a phrase captures the key concepts of a document. In their study, informativeness is specifically defined as the disparity between the foreground document and the background knowledge. Additionally, [52] outlines a set of properties to illustrate keyness in keyphrases. Recent research [155] introduces and quantifies several fundamental concepts in summarization from the standpoint of Information Theory.

Motivated by these studies, I endeavor to define and formulate the task of keyphrasification in this study. A well-defined formulation can facilitate a deeper understanding of this task and pave the way for the development of automated methods in a more systematic manner. The primary objective of this chapter is to address RQ1. Herein, I will delve into the formulation of keyphrasification from an intuitive standpoint. Building upon previous discussions, I will delineate several pivotal properties inherent to keyphrasification. Furthermore, I will conduct an interview study to gain insights into how humans perceive keyphrases, as well as how they utilize and assign them. The findings of this interview can offer valuable insights into the task and serve as crucial evidence to substantiate my formulation.

3.2 FORMULATING KEYPHRASIFICATION

The concept **KeyPhrasification** can be broadly defined as *the process of summarizing rich content using a collection of phrases, a task that can be undertaken by either humans, such as librarians, or automated systems*. Keyphrasification, akin to summarization, which represents a more general scenario, can be viewed as a form of lossy information compression. Both processes aim to succinctly capture the main ideas of a piece of data. However, a fundamental distinction lies in the output format: keyphrasification condenses the content into a list of short phrases, whereas summarization typically yields complete sentences. In this study, I exclusively focus on text keyphrasification, which requires the source content to be in textual format.

Keyphrasification can conceptually be performed by *first summarizing the essential ideas of the source text and then expressing them succinctly in the form of phrases*, as illustrated in Fig 5. The abstract and intermediate representation, which encapsulates the core ideas of the source text, is referred to as a semantic unit [152]. A semantic unit is considered an atomic piece of information [8, 54]. This concept is also manifested in various forms such as factoids [200], summary content units [145], or atomic content units [112].

In this way, we can assume that a text X, which can be a word, a phrase, or a sentence, can be represented as a distribution over a set of abstract semantic units. The role of semantic units can be understood like the topics in topic modeling [83, 15] and dimensions in distributional representation [140]. This assumption aligns well with the essence of keyphrasification, as keyphrases are commonly perceived as natural, meaningful, and often unambiguous semantic units. And we can further assume that keyphrases can be regarded as embodiments of semantic units. There are studies that leverage keyphrases as an interpretable representation for documents [110, 211, 5]. Notably, unlike in summarization studies, semantic units in keyphrasification focus more on specific entities and concepts related to the content in the data, rather than encompassing facts and events.

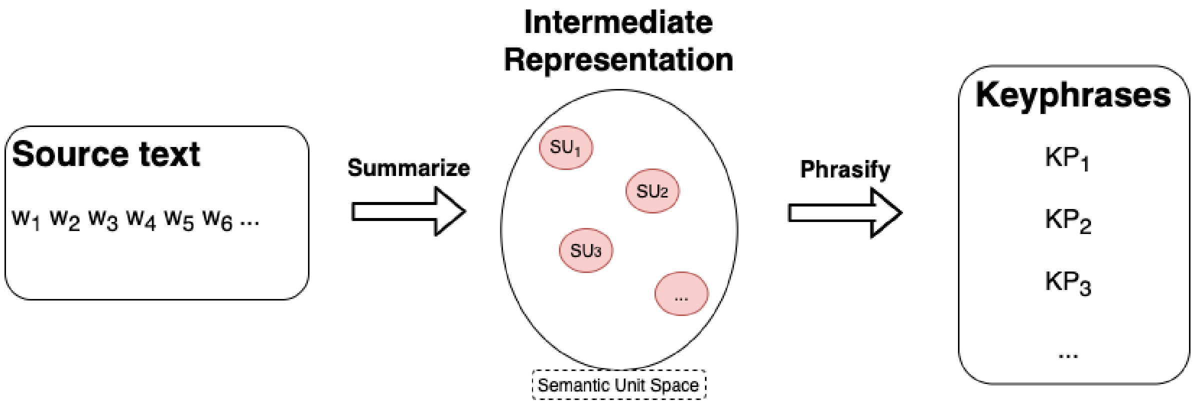


Figure 5: The conceptual formulation of keyphrasification.

3.3 PROPERTIES OF KEYPHRASES

Keyness and phraseness are two fundamental properties of keyphrases. *Keyness* refers to the significance or importance of the summarized information, while *phraseness* pertains to the syntactic quality of the output, focusing on whether the resulting phrases are well-formed and natural. While there is existing literature and concepts from Information Theory that can help us understand keyness, there is a notable gap in studies focusing on phraseness. Therefore, I will consolidate several aspects of phraseness from various perspectives. All properties and their definitions will be summarized and presented in Table 2.

3.3.1 Keyness

Keyness, as described in existing studies, encompasses concepts such as importance [155], salience [48], and quality [39]. These terms encapsulate various facets but collectively signify that certain words or phrases carry more weight in conveying the essence of a text than others. Essentially, keyness involves a balance between informativeness and communication cost [105]. Informativeness pertains to the keyphrases' role in representing the core information of the source text, while communication cost acts as a regularization factor, encouraging a minimal "keyphrase capacity."

Table 2: Keyness/phraseness properties that are discussed in this study.

Category	Property	Description
Keyness - keyphrases as a whole	Coverage	Whether the set of keyphrases contains a range of topics that the source text covers
	Low-redundancy	Whether the semantic overlap among keyphrases in a set is low
Keyness - individual keyphrases	Importance	Whether the keyphrase represents the essence of the source text
	Relevance	Whether the keyphrase is accurate and contains the same information in the source text
	Specificity	Whether the keyphrase is specific to the content of the source text, in comparison to general knowledge
	Uniqueness	Whether the keyphrase captures unique points of the source text
	Prevalence*	The extent to which it is commonly used and encountered within the research community
Phraseness	Conciseness	Whether the keyphrase is short and concise rather than long or descriptive
	Synonymity	Whether the keyphrases contain multiple synonymous phrase candidates that correspond to one semantic meaning

In what follows, I will outline a range of specific properties related to keyness. Coverage and low redundancy address the overall relationships between the source data and all potential keyphrases, while the remaining properties focus on individual keyphrases.

Coverage: This property pertains to the comprehensiveness of keyphrases in covering all major topics within the source text. For instance, consider the phrase "topic tracking" in Fig 1, which encapsulates the core research task discussed in the paper. A set of keyphrases that includes "topic tracking" is deemed superior to one that does not, assuming all other phrases remain the same.

Low-Redundancy: This property addresses the avoidance of duplicate or semantically overlapping phrases among keyphrases. For example, "multiple language" and "multilingual" convey similar meanings, leading to redundancy. A set of keyphrases without such redundancies is considered more optimal.

Importance: This property underscores the significance of the information conveyed by keyphrases to the text's core content, excluding peripheral details. For instance, "story link detection," while mentioned in the paper in Fig 1, may not be essential to its main contributions and thus may not be included as a keyphrase. Similarly, dataset names or evaluation methods, being less critical, are typically excluded as keyphrases. Therefore, a set of keyphrases containing important information is preferable over those lacking such vital elements, assuming other aspects remain equal.

Relevance: Relevance emphasizes that keyphrases should directly relate to the information presented in the source text. For instance, if a paper does not discuss "face detection" despite its importance in the Computer Vision domain, including this phrase in the keyphrases would be considered irrelevant. Thus, in a comparison where all other phrases are identical, sets of keyphrases containing irrelevant terms are deemed less effective.

Note that the relevance property can overlap with attributes like accuracy, consistency, and faithfulness, as they all pertain to how accurately keyphrases represent the content of the original data.

Specificity: Specificity gauges the extent to which a phrase is tailored to the content of the source text as opposed to general knowledge. Given the hierarchical nature of human knowledge perception, keyphrases can range from specific terms found in the text to high-level concepts that encapsulate the text's theme. Both specific and general keyphrases serve different purposes depending on the context.

Uniqueness: Uniqueness refers to whether a keyphrase captures distinctive aspects of the source text that differentiate it from other texts. A low uniqueness keyphrase implies that its information is commonplace and easily derived from general knowledge. For example in Fig 1, terms like "hypothesis" or "stories" may lack uniqueness if they are commonly used and do not convey the unique information of the text. Thus, in a scenario where all other keyphrases are constant, sets of keyphrases containing common terms are considered less valuable.

3.3.2 Phraseness

Unlike keyness, which emphasizes the informational delivery of keyphrases, phraseness pertains to the structure of keyphrases, that is, their formation and syntactic construction in actual usage. In linguistics, a phrase is recognized as a word sequence or a singular word that holds syntactic significance and functions as a sentence component [21]. Grammatically, phrases can assume various forms and lengths, but in practice, keyphrases often appear as concise noun phrases.

While there are limited theoretical discussions on keyphrase morphology, I will outline some phraseness characteristics from different perspectives.

Conciseness: Keyphrases are preferred for their brevity compared to the more descriptive and extended sentences. Analysis of keyphrase lengths in KP20K [135] shows that 97.82% of free-form keyphrases comprise fewer than five words, with bigrams constituting the largest portion (43.16%). The preference for shorter n-grams and concise expressions lacks a formal theoretical explanation but is intuitively linked to reducing cognitive load [88, 126]. Overly long phrases can burden communication and comprehension, leading to the creation of new, shorter symbols to convey the same meaning, in line with the Principle of Least Effort.

Synonymity: Synonymity refers to the existence of multiple synonymous phrases conveying the same semantic content. A semantic unit can be represented in various ways, including detailed sentences or succinct phrases. Due to the natural language diversity, a semantic unit may be expressed through different synonymous phrases. For instance, the terms “TDT” and “topic tracking” in Fig 1 refer to the same concept, which could also be labeled as "topic detection and tracking" or "topic evolution." Thus, several keyphrases might represent the same semantic idea, each serving as a valid expression of that concept.

However, in practice, synonym redundancy is usually minimized to avoid overloading the text with repetitive terms. This creates challenges for evaluating keyphrasification with real-world data, as often only a limited set of synonymous keyphrases are annotated for each semantic unit, underscoring the need for semantic-based evaluation methods in keyphrasification.

Diversity in Syntactic Forms Humans often use noun phrases as keyphrases, yet research indicates that not all keyphrases are nouns. Syntactically, keyphrases can embody various parts of speech, contributing differently to a sentence’s structure. For instance, in Fig 1, terms like “multilingual” and “crosslingual” are employed to emphasize the study’s focus on multilanguage topic tracking. Linguistics recognizes phrases as noun, adjective, adverb, verb, and prepositional phrases. Research in [39] shows noun phrases dominating in academic publications (64.95%), with verb phrases also present (7%). However, the analysis didn’t categorize or examine the remaining keyphrases’ types. In another study [186], focus was on entity keyphrases, finding that about 40% of ground-truth keyphrases in OpenKP were non-entities, exhibiting diverse forms like “join a world” and “care tenderly”. This diversity presents challenges for models identifying keyphrases. Both studies have limitations: [39] used an automatic POS tagger, potentially inaccurately for absent phrases, and [186] provided only an estimated figure without methodological clarity. Additionally, OpenKP ’s

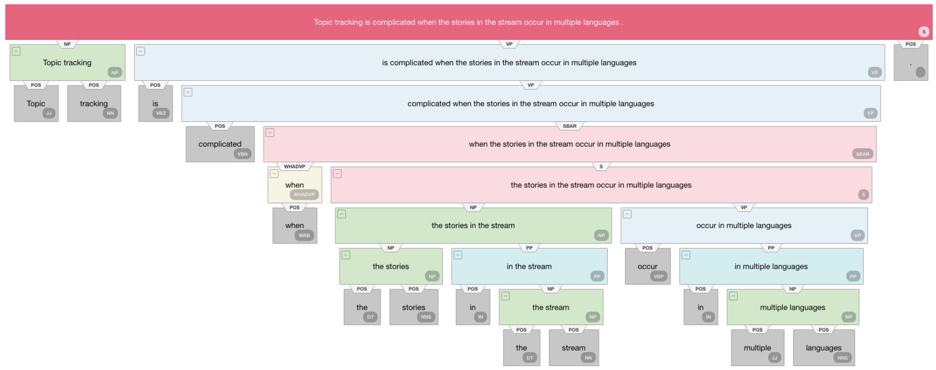


Figure 6: An example of constituency parse tree (demo provided by AllenNLP [63, 95]).

extractive nature may lead to syntactically diverse keyphrases since annotators were restricted to the original text.

Despite these complexities, noun phrases are predominantly used as keyphrases, possibly because they serve as robust semantic units, offering specific, concrete information crucial for understanding, especially in lengthy texts. Noun phrases typically face less ambiguity compared to verbs or adjectives and can be more morphologically stable. Moreover, phrases like "is red" or "was walking" are less informative and often can be converted into more substantive noun forms, such as "complexity" from "complex". Figure 6 illustrates the syntactic parsing, highlighting the noun phrase's centrality in sentence structure.

Practically, the prevalence of noun phrases in keyphrasification aids in the initial extraction of keyphrase candidates. Studies like [39] suggest that simple POS tagging can identify most keyphrase candidates, enabling a focused approach on selecting the most representative ones. This efficiency underpins many successful keyphrase extraction methods, which typically employ a two-step process yielding effective results.

3.4 HUMAN PERSPECTIVES ON KEYPHRASIFICATION: INSIGHTS FROM AN INTERVIEW STUDY

3.4.1 Motivation

Building upon the conceptual formulation of keyphrasification discussed earlier, the practical understanding, assignment, and utilization of keyphrases by humans in real-world settings remain less explored. To address this knowledge gap, I conducted an interview study focusing on the academic realm to gather firsthand insights on how keyphrases are perceived and employed. The aim was not only to enhance our understanding within the academic context but also to extrapolate potential applications and utilities of keyphrases in broader scenarios.

The study was designed to investigate the following research questions, with a particular emphasis on the academic use of keyphrases:

- RQ1: In the academic sphere, what purposes do keyphrases serve?
- RQ2: How important are the keyness/phraseness properties outlined in Sec 3.3, and are there any overlooked properties?
- RQ3: What factors are considered when humans assign keyphrases for academic papers?
- RQ4: Beyond academia, in which other contexts might keyphrases prove beneficial?

3.4.2 Methods

3.4.2.1 Recruitment

To understand how keyphrases are comprehended and annotated in academia, experienced academic researchers were targeted for interviews. Participants were recruited through online advertisements, adhering to the following inclusion criteria: (1) they must be academic researchers, including Ph.D. candidates; (2) aged at least 18 years; and (3) experienced in writing academic papers and assigning keyphrases. Each participant received $20 as compensation for their time. Interviews, lasting about an hour each, were approved by the University of Pittsburgh's Institutional Review Board (IRB #: STUDY20110108).

Fifteen participants were interviewed, all of whom were doctoral students or Ph.D. graduates from the University of Pittsburgh, averaging 4.13 years in research experience. Their backgrounds were mainly in Information Science and Computer Science, which could limit the study's generalizability to these fields alone.

3.4.2.2 Interview Design and Analysis

Individual semi-structured interviews were conducted remotely via Zoom, led by myself and Yu Chi, a fellow doctoral student. These discussions focused on participants' experiences and methods in understanding, using, and assigning keyphrases in academic work.

To address RQ1 and RQ4, we explored participants' perspectives on the utility and significance of keyphrases in both academic and broader contexts. For RQ2 and RQ3, we delved into their decision-making processes in keyphrase selection for three of their publications (with specific identifiers, omitting actual titles for anonymity). Questions like *"How do you decide whether a phrase merits being a keyphrase, and what are your reasons?"* were central to these discussions. The complete interview guidelines, including all procedures and questions, are detailed in Appendix A. Interviews were audio-recorded and transcribed for thorough content analysis, performed collaboratively by the interviewers.

3.4.3 Results

3.4.3.1 Academic Use Cases of Keyphrases

Exploring the practical applications of keyphrases, we sought to understand their academic use.

In academic contexts, we inquired about the scenarios in which keyphrases are utilized. The majority (11 out of 15) of the participants viewed their primary role as aiding in search optimization. For instance, P10 highlighted, "*I do expect keywords to help in searches, so if someone is searching for a paper I would expect that the matching algorithm will use keywords as one way to give more relevant search results.*".

In addition to facilitating the paper search, the other four participants also mentioned that keyphrases can be used for (1) identifying relevant reviewers; (2) leveraging trending keywords for paper promotion; and (3) aligning with the intended academic community or venue.

Regarding whether participants review the author-assigned keyphrases in a new research paper, seven mentioned they scan them for a preliminary understanding of the paper, as P02 articulated, "*I think what keyphrases really give me is quickly tell me what this paper is talking about, so I can decide whether I want to read it or not, but I don't think it will actually help me to understand the paper*". Conversely, eight participants felt keyphrases provided insufficient detail to comprehend the paper fully.

3.4.3.2 Keyphrase Assignment Insights

We delved into the rationale and considerations behind keyphrase selection. Initially, we questioned whether participants employed a systematic approach or specific paper sections as keyphrase sources. Five respondents saw it as summarizing the paper's core contributions, typically finalizing keyphrases after completing the manuscript, as P08 explained, "*I mainly look at the main idea of the paper and come up words from the main ideas. I want this paper to pop up and want people to read this paper when keywords match for relevant areas, so that's how and why I select the main ideas of the paper as the keywords.*".

Titles were commonly used as a keyphrase source by six participants, while others mentioned abstracts (4 participants), literature from other authors (2 participants), and section headings (1 participant).

When asked about adherence to keyphrase assignment guidelines, four participants acknowledged familiarity with official guidelines in submission platforms like AMIA [4] and the ACM CCS (ACM Computing Classification System) [55], aimed at thematic categorization for reviewer assignment. One participant recalled guidance from a scientific writing book, whereas the majority described their selection process as intuitive, epitomized by P14's remark, "*my methods of choosing keywords are kind of intuitive I think the method should be like this*"

On the number of keyphrases, norms (community convention) typically guide the selection of three to seven, as P13 noted, "*it's just basically experience. When you read a lot of papers and they all have like five to seven key keywords, you just select that five to seven keywords.*" Three participants aimed for maximum keyphrase inclusion to enhance paper visibility. Three participants commented that the number does not matter and P10 suggested "*quality over quantity*".

Regarding keyphrase ordering, most participants (13) stated that a certain ordering would be applied, and the rest thought "*it has some logic, but it's not that important*". Specifically, six participants consider **importance** as the major aspect, for example, according to P07 "*I think, maybe all people will defaultly, make the most important one, at the first and then gradually to the least important one.*". Three participants use **granularity** as the criterion, arranging the keyphrases from general to specific. **Function** is adopted by three participants, as stated by P11 "*I put the more familiar ones and hot topic in the beginning, and then some interesting topics and contributions at the end*", though others may use other function ordering. Lastly, two participants suggested **relevance** and the most relevant keyphrases would be put first. The disagreement indicates varied strategies in keyphrase organization.

3.4.3.3 Assessing the Importance of Keyphrases Properties

This segment explores participants' perspectives on keyphrase properties, conducted in two stages.

In the first part, we provided the participants with a list of properties and their definitions, inquiring if they wished to suggest additional properties. Subsequently, they were asked to rate the importance of each attribute on a scale from 0 to 5. The average scores are displayed in Table 3. Initially, seven attributes were listed, with **Prevalence** later added on a participant's suggestion, receiving only 12 ratings.

Properties rated above 4.40 were Importance (4.73), Conciseness (4.60), Coverage (4.47), and Prevalence (4.42), each embodying a distinct keyphrase facet. **Importance** reflects the keyphrase's relevance to the paper's core content, as P02 noted: "*I think the most important value of keywords is to let you understand what's the most important part of this paper.*" **Conciseness** pertains to phraseness, with P10 remarking on the need for brevity to avoid confusion. **Coverage** examines how collectively the keyphrases encapsulate the paper's theme, while **Prevalence** indicates the keyphrase's recognition within the research community, per P12: "*it's better to be prevalent and to be well-known by your research community and others*".

Specificity and **Uniqueness** received lower ratings, attributed to their perceived limited impact

Table 3: Participants' rating on the importance of each keyphrase property. Most properties have 15 valid ratings except for prevalence (12 valid ratings). Definitions of most properties are listed in Sec 3.3.

Property	**Avg score**
Coverage	4.47
Low-redundancy	4.33
Importance	4.73
Relevance	4.33
Uniqueness	3.93
Specificity	3.80
Prevalence	4.42
Conciseness	4.60

on broadening the paper's conversational reach or following established research trends, as highlighted by participants' feedback. For the former, participants think, for example, P03: "*general terms are also sometimes important for a lot of papers because you want to have a conversation with more general broader areas not just a very specific group or research community.*". As for uniqueness, P15 commented that "*I'm trying not to show the uniqueness of my paper but I'm trying to follow the previous work so it is easy for others to find*". Basically, from the perspective of promoting the visibility of a paper, a very specific or unique keyphrase would decrease the chance of being accessed by the readers. Nevertheless, in comparison to regular phrases in the same paper, keyphrases should be more specific or unique overall.

Table 4: Participants' rating on the importance of keyphrases by functions/roles.

	Community			**Paper - General**				**Paper - Specific**			
	Domain	Topic	Task	Focus	Contribution	Goal	Application	Method	Setting	Dataset	Total
Count	20	23	15	29	11	5	14	21	5	2	145
Avg score	3.5	3.7	4.7	4.3	4.6	4.0	2.5	4.1	3.4	3.5	3.9

In the second phase, participants were tasked to annotate the function or role of each keyphrase (i.e. the keyphrase represents which aspects of the paper) in three of their papers, and rate their significance. To avoid confining responses, we offered examples like area, method, problem, solution, goal, focus, domain), and we merged similar functions for analysis such as “domain” and “broader area”, “method” and “model”.

The findings, summarized in Table 4, show a preference for properties neither too general nor too specific. Notably, **Task** and **Contribution** were deemed vital, indicating the paper’s objectives and innovations. Conversely, more general attributes like **Domain** and **Topic** received lower importance ratings, possibly due to their perceived lesser direct relevance to the paper’s specific content such as **Task**. Among the more detailed functions, **Method** was valued for its importance in representing the paper’s approach.

3.4.3.4 General Use Cases of Keyphrases

Exploring the application of keyphrases beyond academia, we found diverse utilizations across various sectors. Three participants highlighted the widespread use of hashtags on social media platforms like Twitter and Instagram, noting their effectiveness in aggregating content around specific topics. For instance, one participant described using a phrase related to global warming to retrieve relevant tweets linked to that hashtag.

Keyphrases serve as a tool for organizing information, as observed in resource management applications like Mendeley, where users assign custom keywords to categorize articles based on their interests. This method aids in structuring knowledge, as P15 explained: "*I’m using a Mendeley like application, I’m assigning most of the time my own keywords to papers that I read.*" Furthermore, keyphrases play a role in presenting knowledge and information, such as in word cloud visuals or presentation slides. One participant mentioned: “*I will use keyphrases for bullet points in slides of my presentation.*”

Another significant application is the labeling of digital content with tags, a practice common in news portals, e-commerce platforms like Amazon, multimedia repositories, and Q&A sites. Tags act as quick-reference markers that help users filter and locate specific content efficiently.

Keyphrases, due to their concise nature, are commonly used to label and categorize extensive or complex resources, such as long articles and academic papers, and media that lack textual descriptions, such as movies and products.

3.5 SUMMARY

This chapter established a conceptual framework for keyphrasification and examined its various properties. An interview study was conducted to delve into human perspectives on keyphrase perception, assignment, and usage. The insights from these discussions highlighted the critical role of keyphrases in summarizing content and connecting it to broader community discussions. Findings also indicate that the process of assigning keyphrases is largely influenced by individual and community practices, with less reliance on formalized guidelines.

4.0 KEYPHRASE GENERATION: KEYPHRASIFICATION BY LANGUAGE GENERATION

In previous chapters, I formulate the keyphrasification as an information compression process and it could be implemented in two steps: summarizing the key information of a source text and then phrasifying it. I also notice that keyphrases, though usually short pieces of text, can also demonstrate the astounding complexity and diversity of natural language, which can hardly be captured using traditional paradigms for predicting keyphrases. For example, the tags used in Automatic Tagging are typically high-level, that do not reflect the thorough semantics of a text. Moreover, its tag vocabulary is often fixed and of limited size, which can hardly stay updated with the latest trends. Nevertheless, keyphrase extraction techniques are strictly constrained by the content given in the text, which means they can never predict keyphrases that are absent in the text, contributing to up to 43.8% of total keyphrases (Table 1).

To overcome the aforementioned shortcomings in existing automatic keyphrase techniques, I propose to treat individual keyphrases like a shorter sentence, that is, as a piece of natural language that consists of a sequence of words. Specifically, I propose to utilize language generation methods for modeling keyphrases, expecting models to predict phrases of various forms, instead of being constrained by tag vocabularies or source texts. Furthermore, with the help of recent advances in deep neural networks, the model training is completely data-driven and models are expected to output good phrases in terms of both keyness and phraseness.

I name this novel paradigm for automatic keyphrasification as **Keyphrase Generation** (**KpGen**), and I will discuss this technique in detail in this chapter. To start with, I will introduce Sequence-to-Sequence Learning, which serves as the backbone of **KpGen** and conceptually matches the two-step formulation (summarize and phrasify) in Chapter 3. After that, I will first present **KpGen–One2One** and **KpGen–One2Seq**, two implementations of **KpGen** using the Seq2Seq learning. The former learns to generate keyphrases individually and the latter predicts multiple keyphrases in a concatenated sequence. Meanwhile, several special components are proposed, expecting to resolve the unique challenges in keyphrasification. I conduct extensive empirical studies to investigate their performance. The results show that the proposed **KpGen** models not only outperform previous baselines on present keyphrase prediction (keyphrase extraction), but also predict keyphrases that do not appear in the original text.

4.1 BACKGROUND

4.1.1 Sequence-to-Sequence Learning

Sequence-to-Sequence Learning [36, 191] is a supervised machine learning method for language generation problems, such as machine translation and text summarization [165, 204, 172]. It is a special case of the Encoder-Decoder Model, for the cases that both input and output data are sequences. Many studies use Sequence-to-Sequence Learning and Encoder-Decoder Model interchangeably.

In this study, both the source text and target keyphrases are treated as sequences of words. Then a Sequence-to-Sequence model is adopted to compress the content of source text into a hidden representation with an encoder, and to generate corresponding keyphrases with a decoder. In what follows, I will first overview the Sequence-to-Sequence Learning in general. Then I will discuss the specific RNN-based Encoder-Decoder architecture as well as the attention mechanism in detail, which is the primary model used in this study.

In a basic Encoder-Decode architecture, an encoder module is leveraged to extract information from the source data and a decoder module is to express this information into the target structure. For sequence learning, typical options of encoder/decoder modules include vanilla Recurrent Neural Networks (RNNs) [93, 47], RNNs variants such as LSTM [82] and GRU [35], as well as Transformers [202]. I will use RNNs for example here without loss of generality.

For RNNs as well as their variants, an encoder RNN converts the variable-length input sequence $\mathbf{x} = (x^1, x^2, ..., x^N)$ into a set of hidden representation $\mathbf{h} = (h^1, h^2, \ldots, h^N)$, by iterating the following equations along time t:

$$h^t_{enc} = f_{enc}(x^t, h^{t-1}_{enc}) \tag{1}$$

where f_{enc} is a non-linear function implemented by neural networks. I get the context vector $\mathbf{h_x}$

acting as the representation of the whole input $\mathbf{x}$ through a non-linear function f_{proj}.

$$h_{enc} = f_{proj}(h_{enc}^1, h_{enc}^2, ..., h_{enc}^N) \tag{2}$$

It is also common that I simply take the hidden state of the last time step of the encoder as the context vector:

$$h_{enc} = h_{enc}^N \tag{3}$$

Then a decoder RNN generates a variable-length sequence $\mathbf{y} = (y^1, y^2, ..., y^{N'})$ word by word, conditioned on previous outputting word y^{t-1}, previous decoding state h_{dec}^{t-1}, and the context vector h_{enc}:

$$\begin{aligned} h_{dec}^t &= f_{dec}(y_{t-1}, h_{dec}^{t-1}, h_{enc}) \\ p(y^t|y^{1,\ldots,t-1}, \mathbf{x}) &= f_{out}(y^{t-1}, h_{dec}^t, h_{enc}) \end{aligned} \tag{4}$$

The non-linear function f_{out} is typically a softmax classifier, that outputs the probabilities of all the words in the preset vocabulary. y^t is the word predicted at time t, by taking the one with largest probability after $f_{out}(\cdot)$.

The encoder and decoder networks are optimized jointly with back-propagation [79, 29], by maximizing the conditional probability of the target sequence, given the corresponding source sequence. More formally, the optimization objective of an Encoder-Decoder model is defined by the following likelihood function:

$$\mathcal{L} = -\sum_{t=1}^{T'} log(p(y^t|y^{1\ldots t-1}, \mathbf{x})) \tag{5}$$

4.1.2 Encoder-Decoder Architecture Using Gated Recurrent Unit

Specifically, I use the Gated Recurrent Unit [35], a common variant of Recurrent Neural Networks, to implement the Encoder-Decoder model in this study.

Given a source text consisting of N words $x^1, \ldots, x^N$, the encoder converts their corresponding embeddings $e_x^1, \ldots, e_x^N$ into a set of N real-valued vectors $h_{enc} = (h_{enc}^1, \ldots, h_{enc}^N)$ with a bidirectional network, consisting of one forward GRU (GRU_{fwd}) encoding the source from left to right and one backward GRU (GRU_{bwd}):

$$
\begin{aligned}
h_{fwd}^t &= \mathrm{GRU}_{fwd}(e_x^t, h_{fwd}^{t-1}), \\
h_{bwd}^t &= \mathrm{GRU}_{bwd}(e_x^t, h_{bwd}^{t+1}), \\
h_{enc}^t &= \langle h_{fwd}^t, h_{bwd}^t \rangle.
\end{aligned} \tag{6}
$$

Dropout [183] is applied to both x and h for regularization.

The decoder is a uni-directional GRU, which generates a new state h_{dec}^t at each time-step t given the embedding of previous word e_y^{t-1} and the recurrent state h_{dec}^{t-1}.

$$
h_{dec}^t = \mathrm{GRU}_d(e_y^{t-1}, h_{dec}^{t-1}). \tag{7}
$$

The initial state h_{dec}^0 is derived from the final encoder state h_{enc}^N by applying a single-layer feed-forward neural net (FNN):

$$
h_{dec}^0 = L_0^{\tanh}(h_{enc}^N). \tag{8}
$$

Dropout is applied to both the embeddings e_y and the GRU states h_{dec}.

4.1.3 Attention Mechanism

Attention mechanism [6] is a common strategy used along with Sequence-to-Sequence Learning which allows the decoder to attend to different parts of the source text at each decoding step. Instead of encoding the input sequence into a single fixed context vector as in Eq 2, the model learns to generate a dynamic context vector with necessary information on the source side.

Specifically, at the time step t during decoding, the model infers the importance $\alpha^{t,i}$ of each source word x^i given the current decoder state h_{dec}^t. This importance is measured by an energy function with a 2-layer FNN:

$$
\mathrm{energy}(h_{dec}^t, h_{enc}^i) = L_1(L_2^{\tanh}(\langle h_{dec}^t, h_{enc}^i \rangle)). \tag{9}
$$

After normalizing all the energy values, I can obtain an importance distribution over all the source tokens corresponding to the decoding step t:

$$
\alpha^t = \mathrm{softmax}(\mathrm{energy}(h_{dec}^t, h_{enc})). \tag{10}
$$

These attention scores are then used as weights for a refined context vector, which is then concatenated to the decoder state h^t_{dec} in Eq. 7 to derive a generative distribution p_a, which corresponds to Eq. 4 in the general architecture:

$$p_a(y^t) = L_3^{\text{softmax}}(L_4^{\tanh}(\langle h^t_{dec}, \sum_i \alpha^{t,i} \cdot h^i_{enc} \rangle)), \tag{11}$$

where the output size of L_3 equals to the target vocabulary size. Subscript a indicates the *abstractive* nature of p_a since it is a distribution over a prescribed vocabulary.

4.1.4 Encoder-Decoder Architecture Using Transformer

Transformer [202] is another common architecture for language models. Nowadays, it is more widely adopted due to its great performance and scalability, which are the core of models that consist of billions of parameters [23, 195].

Basically, Transformer is a multi-layer architecture, each layer applies a multi-headed self-attention operation over the input embedding, followed by position-wise feedforward layers to produce an output distribution over target tokens. Similar to RNN, Transformer can be applied to implement Sequence-to-Sequence learning. As shown in Figure 7, given a source I and a target sequence O, the model learns the transformation from I to O.

The encoder part of the model (see the left part of Figure 7) encodes the normal sentence with a stack of L identical layers. Each layer has two sublayers: one layer is for multi-head self-attention and the other one is a fully connected feed-forward neural network for transformation. The multi-head self-attention layer, where H refers to the number of heads, encodes the output from the previous layer into hidden state $e_{(s,l)}$ (step s and layer l) as shown in Equation 12, where $\alpha^{enc}_{(s\prime,l)}$ indicates the attention distribution over the step $s\prime$ and layer l. Each hidden state processes the hidden states in the previous layer through multi-head attention.

The right part of Figure 7 denotes the decoder for generating the target sequence. The decoder also consists of a stack of L identical layers. In addition to the same two sub-layers as those in the encoder part, the decoder also inserts another multi-head attention layer aiming to attend to the encoder outputs. The bottom multi-head self-attention plays the same role as the one in the encoder, where the hidden state $d_{(s,l)}$ is computed in Equation 13. The upper multi-head attention layer is

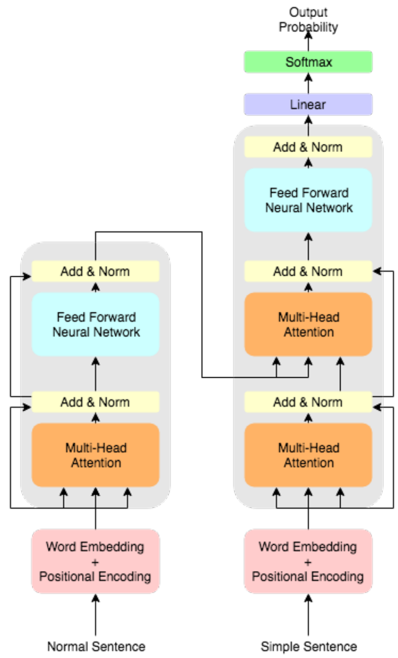


Figure 7: Diagram of the Transformer architecture

used to seek relevant information from encoder outputs. Through the same mechanism, context vector $c_{(s,l)}$ (step s and layer l) is computed in the Equation 14.

$$e_{(s,l)} = \sum_{s'} \alpha^{enc}_{(s',l)} e_{(s',l-1)}, \qquad \alpha^{enc}_{(s',l)} = a(e_{(s,l)}, e_{(s',l-1)}, H) \tag{12}$$

$$d_{(s,l)} = \sum_{s'} \alpha^{dec}_{(s',l)} d_{(s',l-1)}, \qquad \alpha^{dec}_{(s',l)} = a(d_{(s,l)}, c_{(s',l-1)}, H) \tag{13}$$

$$c_{(s,l)} = \sum_{s'} \alpha^{dec2}_{(s',l)} e_{(s',L)}, \qquad \alpha^{dec2}_{(s',l)} = a(d_{(s,l)}, e_{(s',L)}, H) \tag{14}$$

Similarly, the model is trained to minimize the negative log-likelihood of the target sequence, $L_{seq} = -logP(O|I, \theta)$ where θ represents all the parameters in the model.

4.2 PROBLEM DEFINITION

Given a piece of source text $\mathbf{x}$, which can be broken down into a sequence of tokens.

$$\mathbf{x} = x^1, x^2, x^3, \ldots, x^{|\mathbf{x}|} \tag{15}$$

The goal of keyphrasification is to predict a list of keyphrases $\mathbf{y}$ denoted as $\mathbf{y} = \{\mathbf{y_1}, \mathbf{y_2}, \ldots, \mathbf{y_Y}\}$. Similarly, each keyphrase y_i can be seen as a sequence of tokens:

$$\mathbf{y_i} = y_i^1, y_i^2, y_i^3, \ldots, y_i^{|\mathbf{y_i}|} \tag{16}$$

where $|\mathbf{x}|$ and $|\mathbf{y_i}|$ denote the length of $\mathbf{x}$ and $\mathbf{y_i}$ respectively. In general, I do not assume any specific relationship between keyphrases, therefore the target keyphrases $\mathbf{y}$ can be seen as a set.

4.3 METHODOLOGY

4.3.1 Overview

As mentioned in Sec 4.2, a source text can be summarized into multiple keyphrases, which can be dependent or independent of each other. This poses a significant difference from typical Sequence-to-Sequence Learning problems where only a single target sequence is considered. To tackle the unique challenge of generating multiple targets, I propose two training paradigms for applying Sequence-to-Sequence Learning to keyphrasification: `One2One` and `One2Seq`.

Simply put, their main difference lies in how target keyphrase multiplicity is handled in constructing data points (Figure 8). With multiple target phrases $\{p_1, \ldots, p_n\}$, `One2One` takes one phrase at a time and pairs it with the source text s to form n data points $(s, p_i)_{i=1:n}$. During training, a model learns a one-to-many mapping from s to p_i's, i.e., the same source string usually has multiple corresponding target strings. In contrast, `One2Seq` concatenates all ground-truth keyphrases p_i into a single string:
$P =$ <BOS>p_1<SEP>$\cdots$<SEP>p_n<EOS> (i.e., prefixed with <BOS>, joint with <SEP>, and suffixed with <EOS>), thus forming a single data point (s, P). A system is then trained to predict

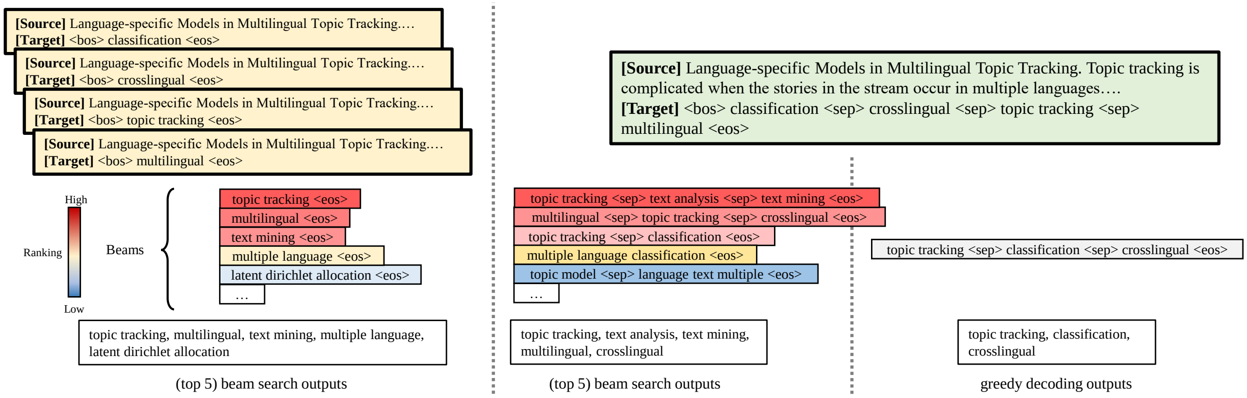


Figure 8: **Top:** comparison between `One2One` (left) and `One2Seq` (right) paradigms on the same data point. **Bottom:** demonstration of the decoding process for `One2One` (left) and `One2Seq` (mid/right) models. `One2Seq` can apply both beam search (mid) and greedy decoding (right).

the concatenated sequence P given s. By default, I construct P following an ordering strategy **PRES-ABS**: I sort present phrases by their first occurrences in the source text, and append absent keyphrases at the end. All ordering strategies will be discussed in §4.3.3.2.

4.3.2 Generating a Keyphrase as a Sequence of Tokens (KPGen-One2One)

One core issue that prohibits us from using the above model directly is that sequence-to-sequence learning is initially designed for generating a single sequence. However, in practice, a text is often assigned multiple keyphrases. In this section, I will discuss several important components to adapt an Encoder-Decoder model for generating keyphrases. As the very first method of Keyphrase Generation, I simplify its goal to generating one phrase at a time. I name this method **KPGEN-ONE2ONE**, since given **ONE** source text, the model is trained to generate **ONE** keyphrase at a time.

4.3.2.1 Data Preparation

In a keyphrase dataset, typically each data example contains one source text and multiple target keyphrases. To apply the Encoder-Decoder model, I will first convert the data format into a format which each data point contains only one source sequence and one target sequence.

Given one data example $\langle \mathbf{x}, \{\mathbf{y_1}, \mathbf{y_2}, \ldots, \mathbf{y_N}\}\rangle$ consisting of one source text $\mathbf{x}$ and $\mathbf{N}$ target keyphrases $\mathbf{y_i}$, I simply convert it into $\mathbf{N}$ data points by copying multiple times the source text $\mathbf{x}$, that is, into pairs like $\langle \mathbf{x}, \mathbf{y_1}\rangle, \langle \mathbf{x}, \mathbf{y_2}\rangle, \ldots, \langle \mathbf{x}, \mathbf{y_N}\rangle$. Then the Encoder-Decoder model is ready to be applied to learn the mapping from the source sequence to the target sequence, as shown in Fig 9. For simplicity, $(\mathbf{x}, \mathbf{y})$ is used to denote each data pair in the rest of this section, where $\mathbf{x}$ is the word sequence of a source text and $\mathbf{y}$ is the word sequence of one of its keyphrases.

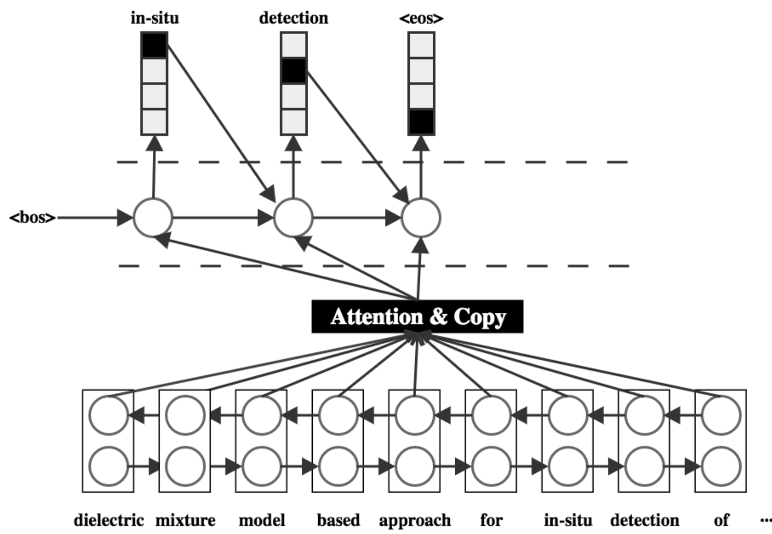


Figure 9: An attentive Encoder-Decoder architecture for Keyphrase Generation (KpGen–One2One).

4.3.2.2 Incorporating Copying Mechanism: Generating with Extractive Attention

Sequence-to-Sequence models are able to generate novel phrases not featured in the source text by their abstractive capability. However, it suffers from certain shortcomings. Firstly, by squeezing the source information into a fixed-size vector, the model is trained to learn a meaningful representation of text. But on the downside, it loses the fine-grained information that can be directly seen in the source text. For example, previous analysis[39] shows that more than 60% of keyphrases are present in the source text. Nevertheless, even for predicting absent phrases, models could utilize some in-text information to infer related phrases. Secondly, for pure Sequence-to-Sequence models, a word vocabulary must be constructed in advance with the most frequent words in the corpus. However, the number of words in the vocabulary is much smaller than the actual words in the real world. Therefore when the models are trying to predict something that is out-of-vocabulary (OOV), they often output a term "<unk>", which denotes an unknown word.

To deal with this shortcoming, I propose to incorporate the Sequence-to-Sequence models with Copying Mechanism [73, 1], or Pointer generator in [170], Forced-Attention Sentence Compression model [137] and Pointer Softmax in [74], to facilitate the abstractive generation with extracted information from the source end. This mechanism works like extractive attention, i.e. at each time step of the decoding process, this extra attention locates the important information that can be directly extracted from the source text, and ultimately the model will output the next word by considering both the extractable information and its original generative distribution, or in another word abstractive distribution, as in Eq. 4 and Eq. 11.

Concretely, I employ an additional module, a Softmax-based switch, to compute a scalar s^t at each decoding step and use it to interpolate the abstractive distribution $p_a(y^t)$ over the pre-set vocabulary (see Eq. 11) and the extractive distribution $p_x(y^t) = \alpha^t$ over the source text tokens (see Eq. 10):

$$p(y^t) = s^t \cdot p_a(y^t) + (1 - s^t) \cdot p_x(y^t), \tag{17}$$

where s^t is conditioned on both the attention-weighted source representation $\sum_i \alpha^{t,i} \cdot h^i_{enc}$ and the decoder state h^t_{dec}, L_i denotes a feed-forward neural network:

$$s^t = L_5^{\text{sigmoid}}(\tanh(L_6(\sum_i \alpha^{t,i} \cdot h^i_{enc}) + L_7(h^t_{dec}))). \tag{18}$$

4.3.2.3 Generating Multiple Keyphrases with Beam Search

Since the model is trained to generate one phrase merely, I propose to utilize Beam Search in order to proliferate the generated phrases.

Beam search is a heuristic search algorithm to predict approximate solutions and it is commonly used in language generation tasks for fast searching optimal sequences. Different from existing studies aiming for a single optimal sequence, I use Beam Search to generate K different sequences.

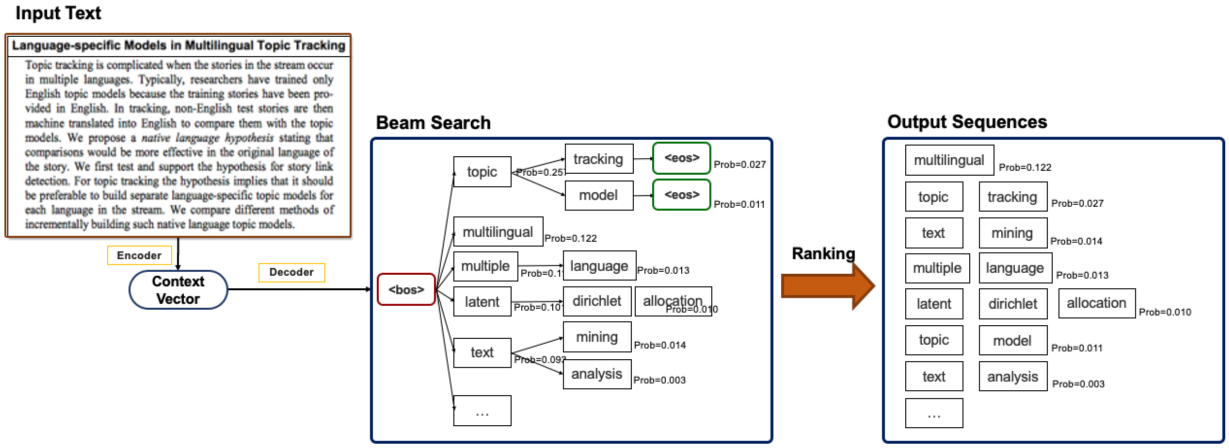


Figure 10: Illustration of Beam Search used in this study. Note that the Beam Search process is simplified due to the space limit, for example, $\langle eos \rangle$ is only shown for the first two sequences and many expanded nodes are omitted.

During the decoding phase, the generation process starts with a special token $\langle bos \rangle$, indicating the beginning of a sequence, and the following words are generated step by step. In the meantime, a max heap is maintained to store the unfinished predictions with the highest probabilities. Once an ending symbol $\langle eos \rangle$ is predicted, the finished sequence will be moved into the result list.

It is worth noting that typically Beam Search is quite time- and memory-consuming. Previous studies in translation and summarization usually use a small beam size/width and large search depth for obtaining longer sequences. But, for keyphrase generation, I opt for a shallow search with large beam width. This is due to two major differences between this task and other problems: (1) Keyphrases are mostly short. Based on existing statistics, the length of 99% keyphrases is less than six words; (2) Many keyphrases are directly extractable or can find helpful cues from the source text, thus exhaustive searching with abstractive vocabulary is unnecessary and wasting.

4.3.3 Generating Keyphrases as a Sequence of Phrases (KPGen-One2Seq)

KPGEN-ONE2ONE is trained to generate a single keyphrase for each source text. During decoding, an exhaustive Beam Search is used to produce a large number of keyphrase sequences.

However, this modeling and decoding strategy has a few problems: (1) Waste in training: to accommodate the Sequence-to-Sequence modeling, one data point with multiple keyphrases has to be split into multiple data points each has only one keyphrase, leading to a tremendous waste in training; (2) Duplicate in generation: As an intrinsic limitation of beam search, a word with high predicting score may dominate the beam search process and diminish the diversity of the final output. Furthermore, the model generates all target phrases independently, therefore the duplicate generation issue cannot be handled by the model itself, resulting in a mass of similar or synonymous phrases being generated;

To overcome the above issues, a simple modification could be utilized, that concatenates all the target keyphrases as one sequence and then trains the model to generate keyphrases as a whole. I denote this way of modeling as **KpGen-One2Seq**, which means given **ONE** source text, the model aims to generate a **SEQ**uence of keyphrases.

4.3.3.1 Data Preparation and Model Architecture

Given one data example $\langle \mathbf{x}, \mathbf{y} = \{\mathbf{y_1}, \mathbf{y_2}, \ldots, \mathbf{y_P}\}\rangle$ consisting of one source text $\mathbf{x}$ and a set of $\mathbf{P}$ target keyphrases $\mathbf{y_i}$ that are assumed equal-weight and order-invariant, I apply an ordering function $\sigma(\cdot)$ on the target keyphrase set, resulting a sorted list $\sigma(\mathbf{y}) = (\hat{\mathbf{y_1}}, \hat{\mathbf{y_2}}, \ldots, \hat{\mathbf{y_P}})$. Then I concatenate this sequence of keyphrases with a delimiter token `<SEP>`, and the eventual target is a sequence of tokens that can be denoted as $\tilde{\mathbf{y}} = \Big(y_1^1, \ldots, y_1^{N_1}, \texttt{<SEP>}, y_2^1, \ldots, y_2^{N_2}, \texttt{<SEP>}, y_m^1, \ldots, y_m^{N_m}\Big)$, where m is the number of keyphrases and N_i is the length of i-th keyphrase.

Similar to the model architecture used in **KpGen-One2One**, I adopt a sequence-to-sequence based framework with pointer-generator and coverage mechanism adopted in [170]. Rather than the model structure itself, I focus more on several critical factors that are specific to the **KpGen-One2Seq** setting, such as the ordering for concatenating target keyphrases and decoding strategies that fit the current setting.

4.3.3.2 Ordering for Concatenating Phrases

Concatenating multiple phrases as a sequence is a critical step for training One2Seq models. As pointed out by [203], output ordering has significant effects on the successful training of sequence models.

I define six ordering strategies for concatenating target phrases as follows to investigate if different orderings affect the performance of keyphrase generation and which orderings may be optimal for training models in the **KPGEN-ONE2SEQ** setting.

RANDOM: Randomly shuffle the target phrases. As the goal of keyphrase generation is to output an order-invariant structure (a set of phrases), I expect models trained with randomly shuffled targets would capture such nature better than other variants with more fixed ordering.

NO-SORT: Keep phrases in original order, implying an intellectual importance by the original authors.

LENGTH: Sort phrases by their lengths from short to long. Phrases of the same length are sorted in original order.

ALPHA: Sort phrases in alphabetical order (by their first word).

APPEAR-PRE: Sort present phrases by their first occurrences in the source text, and prepend absent phrases at the beginning. Absent phrases are randomly shuffled.

APPEAR-AP: Same to **APPEAR-PRE** but append absent phrases at the end.

4.3.3.3 Decoding Strategies

According to different task requirements, various decoding methods can be applied to generate the target sequence y. **KPGEN-ONE2ONE** models focus more on generating an excessive number of phrases by leveraging Beam Search to proliferate the output phrases. In contrast, **KPGEN-ONE2SEQ** models are capable of determining the proper number of phrases to output. In light of previous research in psychology [201, 56], I name these two decoding/search strategies as **Exhaustive Decoding** and **Self-terminating Decoding**, respectively, due to their resemblance to the way humans behave in serial memory tasks. Simply speaking, the major difference lies in whether a model is capable of controlling the number of phrases to output. I describe the detailed decoding strategies used in this study as follows:

Exhaustive Decoding Since one document often comes with many keyphrases, the capability of producing a certain quantity of phrases is necessary for keyphrasification models. For

KpGen–One2One method, models are capable of generating a large number of sequences with the help of Beam Search, and each sequence is a unique phrase. But for **KpGen–One2Seq**, each produced sequence contains several phrases, and additional processes are needed to obtain a unique phrase list.

Here I follow the method proposed by [226], to extract and rank output phrases as follows:

1. Sort all the sequences generated by Beam Search by their likelihood scores from highest to lowest;
2. For each generated sequence, split it by `<SEP>` to get multiple phrases, and remove duplicated ones;
3. Within a sequence, phrases that are generated earlier have a higher rank.

It is worth noting that the time complexity of beam search is $O(Bm)$, where B is the beam width, and m is the maximum length of generated sequences. Therefore the exhaustive decoding is generally very computationally expensive, especially for **KpGen–One2Seq** setting where m is much larger than in **KpGen–One2One**. It is also wasteful as only a very small portion of phrases generated by **KpGen–One2Seq** models are unique.

Exhaustive Decoding with Early Stopping To make **KpGen–One2Seq** decoding more computationally affordable, I propose to use an early-stopping for beam search. Instead of expanding all the search branches until reaching a given maximum depth, I terminate the beam search once the best sequence is found. That is to say, among all the active sequences of Beam Search at time t, if the top-ranked sequence (with the highest likelihood score) yields the end-of-sequence token `<EOS>`, the whole Beam Search process is shut down.

This is a common heuristic for speed-up in single-sequence generation tasks such as translation and summarization. I observe that it is also effective for **KpGen–One2Seq** decoding, leading to up to 10 times faster decoding. By the time that top sequence is completed, there are usually enough number of predicted phrases, meanwhile, the quality of later generated sequences degenerates drastically and most of them are duplicates of existing phrases. Therefore this early-stopping heuristic achieves a significant efficiency gain without sacrificing the quality of output phrases.

Self-terminating Decoding An innate characteristic of keyphrase tasks is that the number of keyphrases varies depending on the document and dataset genre, therefore dynamically outputting a variable number of phrases is a desirable property for keyphrase generation models. For example, the average number of keyphrases in the training set of KP20K is 5.27 with a variance as high as 14.22.

Since the **KPGEN–ONE2SEQ** models are trained to generate a variable number of phrases as a single sequence joined by delimiters, they can output multiple phrases by simply decoding a single sequence. In other words, in contrast with **KPGEN–ONE2ONE** models that output a likely sequence, the **KPGEN–ONE2SEQ** models implicitly perform an additional task: dynamically estimating the proper size of the target phrase set. Once the model believes that an adequate number of phrases have been generated, it outputs a special token <EOS> to terminate the decoding process.

One notable attribute of the self-terminating decoding strategy is that, by generating a set of phrases in a single sequence, the model conditions its current generation on previously generated phrases. Compared to the exhaustive strategy (i.e., phrases being generated independently by Beam Search in parallel), **KPGEN–ONE2SEQ** models can address the dependency among output phrases in a more explicit fashion. Additionally, since multiple phrases are decoded as a single sequence, decoding can be performed more efficiently than exhaustive decoding by conducting the greedy search or Beam Search on only the top-scored sequence.

4.4 EXPERIMENT SETTINGS

Since Sequence-to-Sequence models contain a lot of parameters and typically rely on a large amount of data, I collect several datasets for this purpose featuring different genres and styles. Specifically, I newly construct two datasets, one is a scientific publication dataset in the Computer Science domain, called KP20K, and one is a social tagging dataset from an online social question-answering community, called STACKEX.

Table 5: Statistics on number of documents and keyphrases of each test set. #Doc#KP denotes the number of documents/ground-truth keyphrases in the dataset. #PreKP/#AbsKP denotes the number of present/absent ground-truth keyphrases, and #PreDoc/#AbsDoc denotes the number of documents that contain at least one present/absent ground-truth keyphrase.

Dataset	#Doc	#KP	#PreDoc	#PreKP	#AbsDoc	#AbsKP
KP20K-TRAIN	≈514k					
KP20K-VALID	2,000	10,530	1,900	6,655	1,646	3,875
KP20K-TEST	19,987	105,181	19,048	66,595	16,357	38,586
INSPEC	500	4,913	497	3,858	381	1,055
KRAPIVIN	460	2,641	437	1,485	417	1,156
NUS	211	2,461	207	1,263	195	1,198
SEMEVAL	100	1,507	100	671	99	836
DUC-2001	308	2,484	308	2,421	38	63

4.4.1 Training Datasets

For both training datasets, I remove duplicates and papers that overlap with the ones in testing datasets.

– **KP20K**: There are several publicly-available keyphrase datasets used in previous studies but the largest one [101] only contains 2,304 scientific publications. This amount of data can hardly train a workable **KPGEN** model. Therefore, I build a brand new dataset, dubbed KP20K, consisting of a large amount of high-quality scientific metadata in the computer science domain from various online digital libraries, including ACM Digital Library, ScienceDirect, Wiley, Web of Science, etc. In total, metadata of 567,830 articles is collected. After preprocessing and cleaning, around 514,000 articles are used as the training set (KP20K-TRAIN), and the rest are used for the validation set (KP20K-VALID, 2,000 data points) and test set (KP20K-TEST, 19,987 data points). I manually clean the data examples in the valid/test set of KP20K (clean noisy text, replace erroneous keyphrases with actual author keyphrases, remove examples without any ground-truth keyphrases), and use scripts to remove invalid training examples (without any author keyphrase). I use the title and abstract as the source text and author-assigned keywords as target keyphrases.

4.4.2 Test Datasets

Several datasets are included in this study for evaluation use. They are constructed and annotated in different ways, covering a wide range of topics and label distributions.

- **KP20K-TEST** includes around 20,000 data examples from the same source as KP20K.
- **INSPEC** [85] provides 2,000 journal paper abstracts and reader-annotated keyphrases. In the original dataset, each abstract has two sets of keywords assigned by a professional annotator: a set of controlled terms, i.e., terms restricted to the Inspec thesaurus; and a set of uncontrolled terms that can be any suitable terms. Both the controlled terms and the uncontrolled terms may or may not be present in the abstracts. I adopt the 500 testing papers and their corresponding *uncontrolled keyphrases* for evaluation, and the remaining 1,500 papers are not used. Since all the phrases are assigned by third-party annotators instead of the original authors, it may deviate from the distribution of our training dataset.
- **KRAPIVIN** [101]: This dataset provides 2,304 papers with full-text and author-assigned keyphrases. However, the author did not mention how to split testing data, so I selected the first 400 papers in alphabetical order as the testing data, and the remaining papers are used to train the supervised baselines.
- **NUS** [146] contains 211 papers and I use both *author-assigned* and *reader-assigned* keyphrases, which contribute to 897 and 1,600 phrases out of 2,336 total phrases. I notice that though the latter takes the majority, there are 57 papers that do not have any *reader-assigned* keyphrases. That explains why it has smaller average number of phrases than the other two reader annotated datasets (INSPEC and SEMEVAL).
- **SEMEVAL** [98] provides 288 articles, spanning 4 categories in Computer Science. 100 articles are used for testing and the rest 144 articles are not used. I use both author-annotated and reader-anntoated keyphrases for testing.
- **DUC-2001** [205] consists of 308 news articles and keyphrases are annotated by readers. This is the only dataset used in this study that is out of the academic domain and its goal is to test the generalizability of each model.

4.4.3 Evaluation Metrics

I pre-process both document texts and ground-truth keyphrases, including word segmentation, lowercasing, and replacing all digits with symbol <DIGIT>. Data examples with empty ground-truth keyphrases are ignored in the evaluation.

I evaluate models' performance in predicting present and absent phrases separately. Specifically, I first lowercase the text, stem each word in the text with Porter Stemmer, and then determine the presence of each ground-truth keyphrase by checking whether it is a sub-string of the source text. To evaluate present phrase performance, I compute Precision/Recall/F1-score (see Eq. 19-21) for each document taking only present ground-truth keyphrases as target and ignoring the absent ones.

$$P@k = \frac{\#(correct@k)}{min\{k, \#(pred)\}} \tag{19}$$

$$R = \frac{\#(correct@k)}{\#(target)} \tag{20}$$

$$F_1@k = \frac{2 * P@k * R}{P@k + R} \tag{21}$$

where $\#(pred)$ and $\#(target)$ are the number of predicted and ground-truth keyphrases respectively; and $\#(correct@k)$ is the number of correct predictions among the first k results. Besides the common k value taken as 5 and 10, in this study, I am also interested in two other values:

- $\mathcal{O}$ denotes the number of oracle (ground truth) keyphrases. In this case, $k = |\mathcal{Y}|$, which means for each data example, the number of predicted phrases taken for evaluation is the same as the number of ground-truth keyphrases.
- $\mathcal{M}$ denotes the number of predicted keyphrases. In this case, $k = |\hat{\mathcal{Y}}|$ and we simply take all the predicted phrases for evaluation without truncation.

Specifically, **$\mathbf{F_1@10}$** and **$\mathbf{F_1@\mathcal{O}}$** are the two primary metrics I use for evaluating KPG models on present keyphrase prediction – the former is the most commonly used metric and the latter improves **$\mathbf{F_1@10}$** by considering the number of actual ground-truth phrases. I leverage **R@10**, **R@50** and $\mathcal{R}@\mathcal{M}$ to evaluate the performance on absent keyphrase prediction. I report the macro-averaged scores over documents that have at least one present ground-truth phrase (corresponding to the column #PreDoc in Table 5) and similarly to the case for absent phrase evaluation.

4.5 RESULTS

4.5.1 Main Results

We first compare the proposed keyphrase generation models with the classic keyphrase prediction models, i.e. keyphrase extraction models.

4.5.1.1 Model Settings

Six classic extraction baselines are used for comparison, including (1) **four unsupervised algorithms** (Tf-Idf, TextRank [139], SingleRank [205], and ExpandRank [205]); (2) **two supervised algorithms** (KEA [216] and Maui [127]) are adopted as baselines. We set up the four unsupervised methods following the optimal settings in [77], and the two supervised methods following the default setting as specified in their papers. I include **four KPGEN models**, using a basic `RNN` architecture (embedding dim 150, hidden dim 300, vocabulary size 50k) and two training paradigms (`One2One` and `One2Seq`). The copying mechanism is ablated for both models. For inference, I use exhaustive decoding and beam width of 200 and 50 for `One2One` and `One2Seq`, respectively.

4.5.1.2 Present Keyphrase Prediction

For the present keyphrase prediction, the setting is the same as the keyphrase extraction task in prior studies, in which we investigate how well the proposed models can be used to identify keyphrases that appear in the original documents.

In Table 6, the results show that the four unsupervised models (Tf-idf, TextTank, SingleRank, and ExpandRank) have a robust performance across different datasets. The ExpandRank fails to return any result on the KP20k dataset, due to its high time complexity. The performance of the two supervised models (i.e., Maui and KEA) is unstable on some datasets, but Maui achieved the best performances on three datasets among all the baseline models.

As for the four **KPGEN** models, the RNN models with the regular attention mechanism did not perform as well as we expected. It might be because the RNN model is only concerned with finding the hidden semantics behind the text, which may tend to generate keyphrases or words that are too general and may not necessarily refer to the source text. In addition, we observe that 2.5% (70,891/2,780,316) of keyphrases in our dataset contain out-of-vocabulary words, which the RNN model is not able to recall, since the RNN model can only generate results with the 50,000 words in the vocabulary. This indicates that a pure generative model may not fit the extraction task.

Table 6: Performance (F_1-score) of present keyphrase prediction on scientific publications datasets and `DUC-2001`. The best performing score in each column is highlighted with bold.

	KP20K			**Inspec**			**Krapivin**			**NUS**			**SemEval**			**DUC**		
Model	**@5**	**@10**	**@$\mathcal{O}$**	**@5**	**@10**	**@$\mathcal{O}$**	**@5**	**@10**	**@$\mathcal{O}$**	**@5**	**@10**	**@$\mathcal{O}$**	**@5**	**@10**	**@$\mathcal{O}$**	**@5**	**@10**	**@$\mathcal{O}$**
Extraction Models																		
TfIdf	7.2	9.4	6.3	16.0	24.4	20.8	6.7	9.3	6.8	11.2	14.0	12.2	8.8	14.7	11.3	-	**27.0**	-
TextRank	18.1	15.1	18.4	28.6	33.9	33.5	18.5	16.0	21.1	23.0	21.6	23.8	21.7	22.6	22.9	-	9.7	-
SingleRank	9.9	12.4	-	21.4	29.7	-	9.6	13.7	-	14.5	16.9	-	13.2	16.9	-	-	25.6	-
ExpandRank	-	-	-	21.1	29.5	-	9.6	13.6	-	13.7	16.2	-	13.5	16.3	-	-	26.9	-
KEA	4.6	4.4	5.1	2.2	2.2	2.2	1.8	1.7	1.7	7.3	7.1	8.1	6.8	6.5	6.6	-	-	-
Maui	0.5	0.5	0.4	3.5	4.6	3.9	0.5	0.7	0.6	0.4	0.6	0.6	1.1	1.4	1.1	-	-	-
Generation Models																		
`RNN One2One`	8.3	8.5	8.4	5.6	5.8	5.0	4.9	5.3	5.3	10.3	10.9	10.4	8.4	8.4	7.5	0.0	0.0	0.0
`+COPY`	**33.1**	**27.9**	**35.5**	**32.1**	26.9	**36.0**	29.0	32.4	33.6	**40.2**	36.2	**43.2**	**34.2**	**35.2**	34.8	11.3	13.9	13.2
`RNN One2Seq`	10.4	10.2	11.1	8.1	7.9	9.9	4.5	4.5	4.5	11.0	10.5	10.9	8.8	8.6	8.8	0.1	0.1	0.1
`+COPY`	31.2	26.1	31.2	31.0	**27.0**	33.5	**32.9**	**38.8**	**38.6**	37.4	**36.6**	39.2	33.7	35.1	**36.2**	11.8	15.9	14.7

4.5.1.3 Absent Keyphrase Prediction

As stated, one important motivation for this work is that we are interested in the proposed model's capability for predicting absent keyphrases based on the "understanding" of content. It is worth noting that such prediction is a very challenging task. Therefore, we only provide the `RNN` performances in the discussion of the results of this task. Here, we evaluate the performance within the recall of the top 10 and top 50 results, to see how many absent keyphrases can be correctly predicted. We use the absent keyphrases in the testing datasets for evaluation.

Table 7 presents the recall scores of predicted absent keyphrases for `RNN` models, in which we observe that the best model (`One2One` with the copy attention) can, on average, recall 4.4%/9.0% of keyphrases at top 10/50 predictions. This indicates that, to some extent, the proposed models can capture the hidden semantics behind the textual content and make reasonable predictions. In addition, with the advantage of features from the source text, the variants with copy attention

Table 7: Performance (recall scores) of absent keyphrase prediction on five datasets. **Boldface** text indicates the best performance in corresponding columns.

	Average			Kp20K			Inspec			Krapivin			NUS			SemEval		
Model	**10**	**50**	$\mathcal{M}$	**10**	**50**	$\mathcal{M}$	**10**	**50**	$\mathcal{M}$	**10**	**50**	$\mathcal{M}$	**10**	**50**	$\mathcal{M}$	**10**	**50**	$\mathcal{M}$
RNN One2One	**4.4**	**9.0**	**14.0**	**6.8**	**13.2**	**19.7**	**7.9**	**13.9**	**20.9**	**4.5**	**9.3**	**15.0**	**4.7**	**11.3**	**17.6**	2.2	**6.1**	**10.0**
+COPY	1.2	3.5	6.2	2.4	5.8	9.8	1.2	4.0	8.5	1.4	2.3	3.3	1.5	5.6	9.6	0.4	3.5	6.1
RNN One2Seq	2.2	2.5	2.5	2.7	3.2	3.3	2.8	3.2	3.3	3.5	3.8	3.8	2.6	2.9	3.0	1.6	1.7	1.7
+COPY	3.1	4.1	4.2	4.9	6.7	6.7	4.6	6.3	6.4	1.7	2.0	2.0	4.5	6.2	6.2	**2.6**	3.6	3.6

outperform the vanilla ones by a significant margin, though it does not show as much improvement as in the present keyphrase setting. As the absent keyphrase prediction highly prefers recall-oriented models, therefore the One2One model with a large beam size (200) is innately proper for this task setting. Comparatively, One2Seq models are able to predict absent keyphrases, but they lag behind One2One counterparts by a large margin. I will discuss their differences in the later sections.

Table 8: Testing scores across different model architectures, training paradigms, and datasets. In which, D_0: in-distribution; D_1: out-of-distribution, and D_2: out-of-domain.

	Dataset	Present (**F**$_1$@$\mathcal{O}$)				Present (**F**$_1$**@10**)				Absent (**R@50**)			
		One2One		One2Seq		One2One		One2Seq		One2One		One2Seq	
		RNN	TRANS	RNN	TRANS	RNN	TRANS	RNN	TRANS	RNN	TRANS	RNN	TRANS
D_0	KP20K	35.3	37.4	31.2	36.2	27.9	28.9	26.1	29.0	13.1	22.1	3.2	15.0
	KRAPIVIN	35.5	33.0	33.5	36.4	27.0	26.4	26.9	28.1	13.7	23.8	3.3	16.6
	D_0 Average	35.4	35.2	32.3	**36.3**	27.4	27.7	26.5	**28.5**	13.4	**23.0**	3.2	15.8
D_1	INSPEC	33.7	32.6	38.8	36.9	32.5	30.8	38.7	36.6	8.2	9.2	3.7	6.7
	NUS	43.4	41.1	39.2	42.3	35.9	36.1	36.6	37.3	11.2	18.9	2.9	12.5
	SEMEVAL	35.2	35.1	36.2	34.8	34.6	33.0	35.0	34.2	6.1	18.9	1.7	12.5
	D_1 Average	37.4	36.3	**38.1**	38.0	34.4	33.3	**36.7**	36.0	8.5	**12.7**	2.8	9.2
D_2	DUC-2001	13.4	7.8	**15.0**	11.0	13.7	8.4	**16.0**	11.4	0.0	**0.2**	0.0	0.0
All	Average	32.8	31.2	32.3	**32.9**	28.6	27.3	**29.9**	29.4	8.7	**14.0**	2.5	9.8

4.5.2 Ablation Study

In this part, I will conduct comprehensive studies to analyze the important components inside **KPGEN** models and discuss their effects. Specifically, the components and factors I will be focusing on include: model architectures (i.e., RNN and TRANSFORMER), training paradigms (i.e., One2One or One2Seq), copy mechanism, beam width, and how to concatenate phrases for the One2Seq training.

To better understand model variants' generalization properties, I categorize the 6 testing sets into 3 classes according to their distribution similarity with the training data (KP20K), as shown in Table 8. Concretely, KP20K and KRAPIVIN are in-distribution test sets (denoted as D_0), since they both contain scientific paper abstracts paired with keyphrases provided by their authors. INSPEC, NUS and SEMEVAL are out-of-distribution test sets (denoted as D_1), they share same type of source text with D_0, but with additionally labeled keywords by third-party annotators. DUC-2001 is a special test set that uses news articles as its source text. Because it shares the least domain knowledge and vocabulary with all the other test sets, I call it an out-of-domain test set (denoted as D_2).

4.5.2.1 Model Architectures: RNN vs TRANSFORMER

The first thing to notice is that on the present KPG, the models show consistent trends between **F_1@10** and **$F_1@\mathcal{O}$**. I observe that TRANSFORMER models significantly outperform RNN models when trained with the One2Seq paradigm on D_0 test sets. However, when test data distribution shift increases, on D_1 test sets, RNN models starts to outperform TRANSFORMER; eventually, when dealing with D_2 test set, RNN outperforms TRANSFORMER by a large margin. On models trained with One2One paradigm, I observe a similar trend. On D_0 data, TRANSFORMER models achieve comparable **F_1@10** and **$F_1@\mathcal{O}$** scores with RNN, when data distribution shift increases, RNN models produce better results.

On the contrary, for absent KPG, TRANSFORMER outperforms RNN by a significant margin in all experiment settings. This is especially obvious when models are trained with One2Seq paradigm, where RNN models barely generalize to any of the testing data and produce an average **R@50** of 2.5. In the same setting, TRANSFORMER models get an average **R@50** of 9.8, which is $4\times$ higher than RNN.

To further study the different generation behaviors between RNN and TRANSFORMER, I investigate the average number of unique predictions generated by either of the models. As shown in Figure 11, comparing results of order **PRES–ABS** in sub-figure a/b (RNN) with sub-figure c/d (TRANSFORMER), I observe that TRANSFORMER is consistently generating more unique predic-

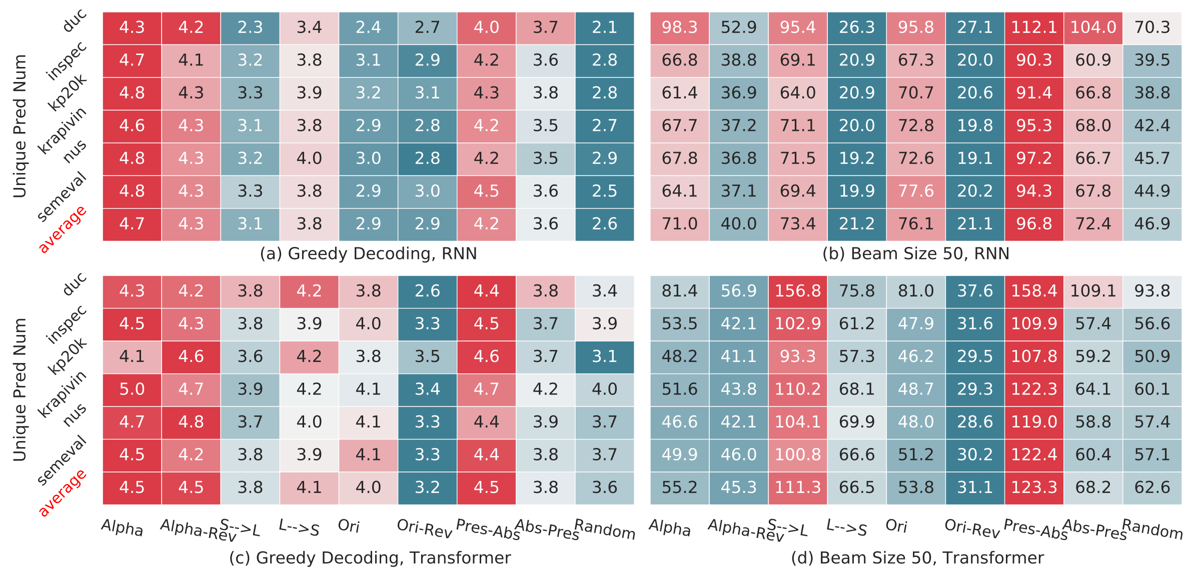


Figure 11: Unique number of keyphrases generated during the test. Colors represent the relative performance, normalized per row. Best checkpoints are selected by $\mathbf{F}_1@\mathcal{O}$ scores on KP20K-VALID.

tions than RNN, in both cases of greedy decoding (4.5 vs 4.2) and beam search (123.3 vs 96.8). I suspect that generating a more diverse set of keyphrases may have a stronger effect on in-distribution test data. The generated outputs during inference are likely to represent the distribution learned from the training data, when the test data share the same (or similar) distribution, a larger set of unique predictions leads to a higher recall — which further contributes to their F-scores. In contrast, on test sets which data distribution is far from training distribution, the extra predictions may not be as useful, and even hurts precision. Similarly, because I evaluate absent KPG by the models' recall, TRANSFORMER models — produce more unique predictions — can always outperform RNN models.

4.5.2.2 Training Paradigm: One2One vs One2Seq

I observe that on present KPG tasks, models trained with the One2Seq paradigm outperforms One2One in most settings, this is particularly clear on D_1 and D_2 test sets. I believe this is potentially due to the unique design of the One2Seq training paradigm where at every generation step,

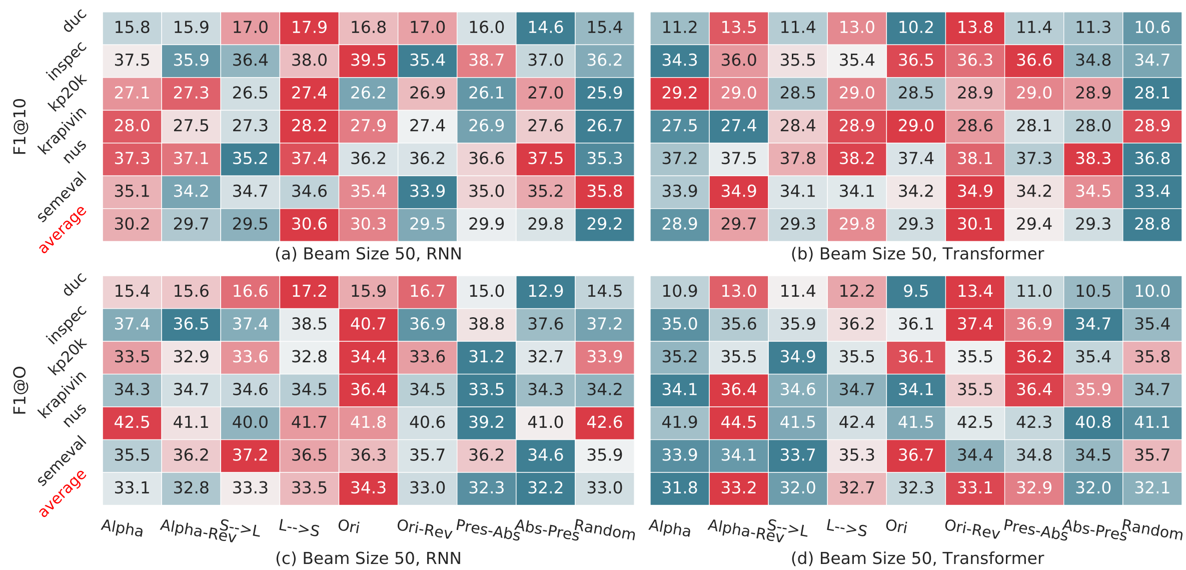


Figure 12: Present KPG testing scores (**top: $\mathbf{F_1@10}$**, **bottom: $\mathbf{F_1@\mathcal{O}}$**). Colors represent the relative performance, normalized per row.

the model conditions its decision making on all previously generated tokens (phrases). Compared to the `One2One` paradigm where multiple phrases can only be generated independently by beam search in parallel, the `One2Seq` paradigm can model the dependencies among tokens and the dependencies among phrases more explicitly.

However, on absent KPG, `One2One` consistently outperforms `One2Seq`. Furthermore, only when trained with `One2One` paradigm, an `RNN`-based model can achieve **R@50** scores close to TRANSFORMER-based models. This may because a `One2Seq` model tends to produce more duplicated predictions during beam search inference. By design, every beam is a string that contains multiple phrases that concatenated by the delimiter `<SEP>`, there is no guarantee that the phrase will not appear in multiple beams. In the example shown in Figure 8, "topic tracking" is such a duplicate prediction that appears in multiple beams. In fact, the proportion of duplicates in `One2Seq` predictions is more than 90%. This is in contrast with beam search on `One2One` models, where each beam only contains a single keyphrase thus has a much lower probability of generating duplication.[1]

4.5.2.3 Effect of Copy Mechanism

Copy mechanism [72] (also referred to as Pointer Generator [171] or Pointer Softmax [75]) has demonstrated to play a critical role in tasks where texts on the source and target side may overlap, such as summarization [171] and keyphrase generation [135]. Basically, it is an additional loss that enables models to extract information from the source side with the help of attention. Prior studies [135] have shown the importance of copy mechanism with `RNN+One2One`, but no further comparison has been made.

In Figure 13, I present the results of four KPG model variants, equipped with and without copy mechanism. The results show that copy mechanism leads to considerable improvements on the present KPG, especially for `RNN`. `TRANSFORMER` benefits less from the copy, which may be because its multi-head attentions behave similarly to the copy mechanism even without explicit training losses. With regard to the absent KPG results, copy mechanism only helps `RNN+One2One`. This suggests that `TRANSFORMER` can achieve consistently better abstractiveness (absent performance) by disabling the copy mechanism at the cost of weaker extractiveness. This dilemma cautions researchers to use copy mechanism more wisely according to specific applications.

4.5.2.4 Effect of Beam Width

One unique challenge of the KPG task is due to its multiplicity of target outputs. As a result, a common strategy is to take multiple beams during decoding in order to obtain more phrases (as opposed to greedy decoding). This choice is at times not only practical but in fact necessary: under the `One2One` paradigm, for example, it is crucial to have multiple beams in order to generate multiple keyphrases for a given input.

Generally speaking, KPG and its evaluation metrics are in general favors higher recall. It is thus not totally unexpected that the high precision scores of greedy decoding are often undermined by notable disadvantages in recall, which in turn leads to losing by large margins in F-scores when compared to results of beam search (with multiple beams). Empirically, as shown in Figure 14,

[1]Due to post-processing such as stemming, `One2One` model may still produce duplication.

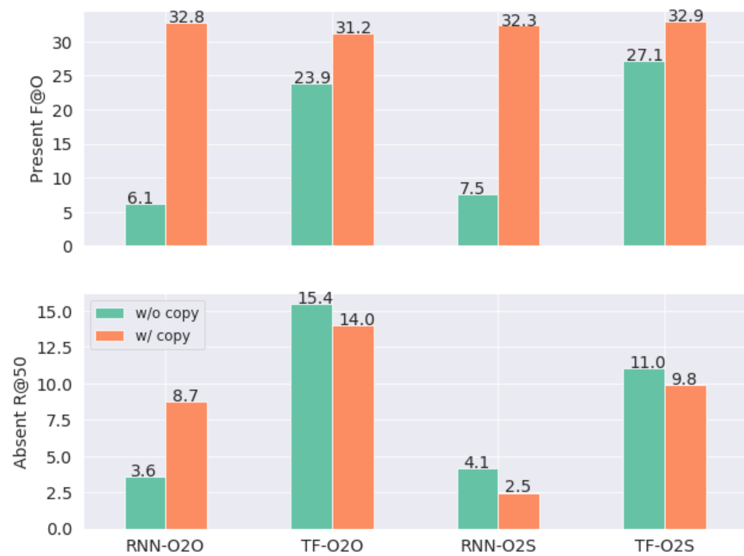


Figure 13: Averaged scores of models with and without utilizing copy mechanism.

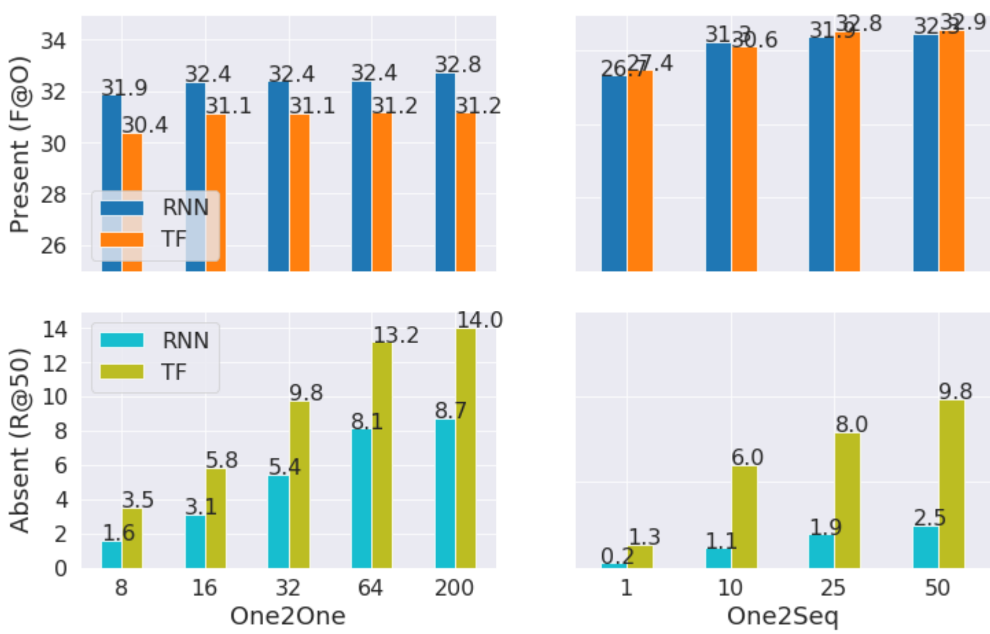


Figure 14: Averaged scores of models using different widths of Beam Search.

I observe that beam search can sometimes achieve a relative gain of more than 10% in present phrase generation , and a much larger performance boost in absent phrase generation, over greedy decoding.

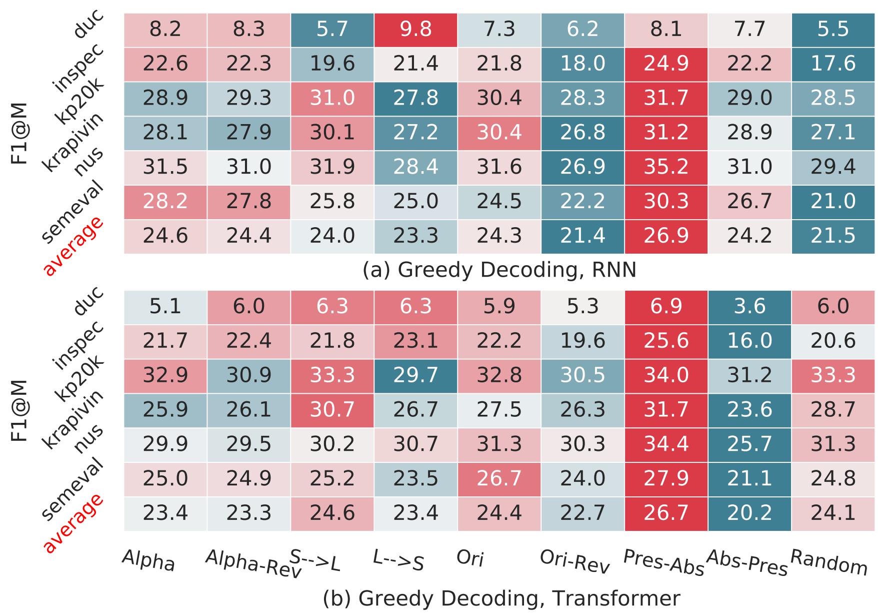
I am also interested in seeing if there exists an optimal beam width. In Figure 14, I show models' testing performance when various beam widths are used. In present KPG task with `One2One` (upper left), beam width of 16 already provides an optimal score, larger beam widths (even 200) do not show any further advantage. Replacing the training paradigm with `One2Seq` (upper right), I observe a positive correlation between beam width and testing score — larger beam widths lead to marginally better testing scores. However, the improvement (from beam size of 10 to 50) is not significant.

On absent KPG task (lower), both `One2One` and `One2Seq` paradigms seem to benefit from larger beam widths. Testing score shows strong positive correlation with beam width. I observe that this trend is consistent across different model architectures.

Overall, a larger beam width provides better scores in most settings, but the performance gain diminishes quickly towards very large beam width. In addition, it is worth noting that larger beam width also comes with more intense computing demands, for both space and time. As an example, in Figure 14 (top left), I observe that with the `One2One` training paradigm, a beam width of 200 does not show a significant advantage over 16, however, in terms of computation, beam width of 200 takes about $10\times$ of the resources compared to 16. There exists a trade-off between beam width and computational efficiency (e.g., carbon footprint). I thus hope the results can serve as a reference for researchers, to help them choose beam width more wisely depending on specific tasks.

4.5.2.5 Effects of Phrase Ordering in **`One2Seq`** Training

In the `One2One` paradigm (as shown in Figure 8), each data example is split into multiple equally weighted data pairs, thus it generates phrases without any prior on the order. In contrast, `One2Seq` training has the unique capability of generating a varying number of keyphrases in a single sequence. This inductive bias enables a model to learn dependencies among keyphrases, and also to implicitly estimate the number of target phrases conditioned on the source text. However,



(a) Greedy Decoding, RNN

(b) Greedy Decoding, Transformer

Figure 15: Present KPG testing scores ($\mathbf{F}_1@\mathcal{M}$). Colors represent the relative performance, normalized per row.

the `One2Seq` approach introduces a new complication. During training, the Seq2Seq decoder takes the concatenation of multiple target keyphrases as the target. As pointed out by [203], order matters in sequence modeling tasks; yet the ordering among the target keyphrases has not been fully investigated and its effect on the models' performance remains unclear. Several studies have noted this problem [226, 231] without further exploration.

Greedy Decoding

In Figure 15, I show the RNN and TRANSFORMER model's $\mathbf{F}_1@\mathcal{M}$ on present KPG task, equipped with greedy decoding. In this setting, the model simply chooses the token with the highest probability at every step, and terminates either upon generating the `<EOS>` token or reaching the maximum target length limit (40). This means the model predicts phrases solely relying on its innate distribution learned from the training data, and thus this performance could somewhat reflect to which degree the model fits the training distribution and understands the task.

Through this set of experiments, I first observe that each model demonstrates consistent performance across all six test datasets, indicating that ordering strategies play critical roles in training

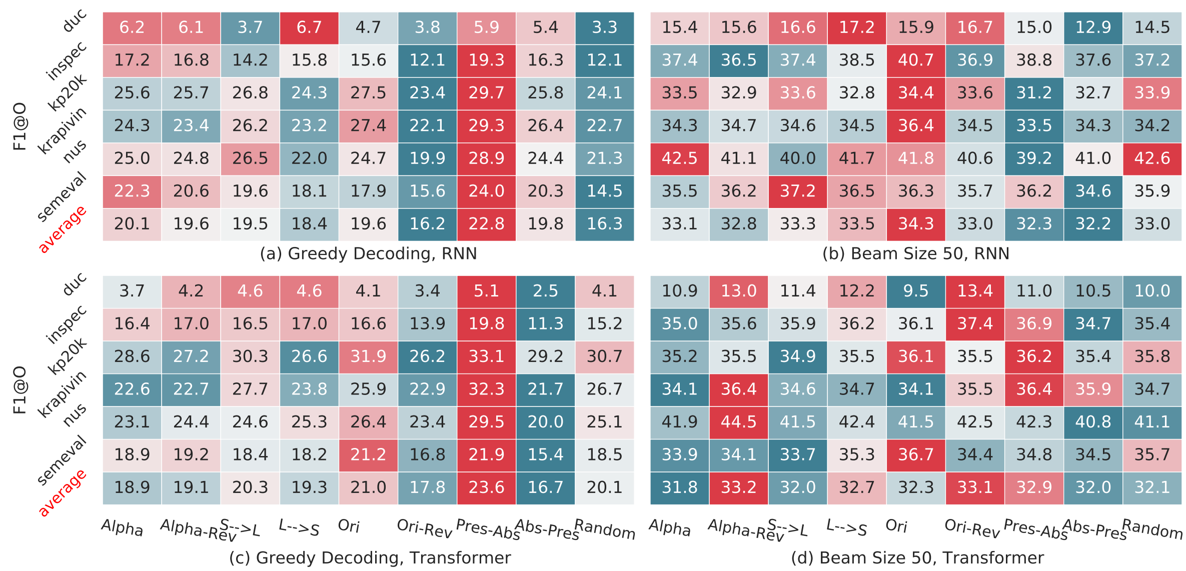


Figure 16: Present keyphrase generation testing scores (**F_1@$\mathcal{O}$**). Colors represent the relative performance, normalized per row.

One2Seq models when greedy decoding is applied. When using the RNN architecture, **RANDOM** consistently yields lower **F_1@$\mathcal{M}$** than other ordering strategies on all datasets. This suggests that a consistent order of the keyphrases is beneficial. However, TRANSFORMER models show a better resistance against randomly shuffled keyphrases and produce average tier performance with the **RANDOM** ordering. Meanwhile, I observe that **PRES–ABS** outperforms other ordering strategies by significant margins. A possible explanation is that with this order (of occurrences in the source text), the current target phrase is always to the right of the previous one, which can serve as an effective prior for the attention mechanism throughout the One2Seq decoding process. I can observe similar trends in greedy decoding models' **F_1@$\mathcal{O}$** and **F_1@10**.

Beam Search

Next, I show results obtained from the same set of models equipped with beam search (beam width is 50) in Figure 12 (a/b). Compared with greedy decoding, it can be clearly observed the overall **F_1@10** scores have a positive correlation with the beam width (greedy decoding is a special

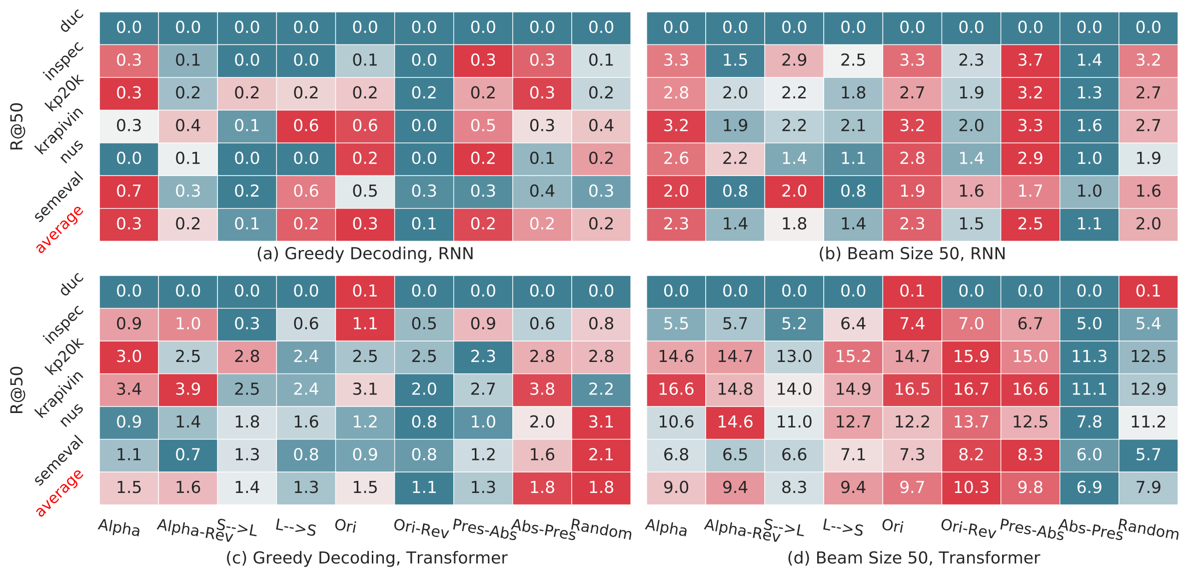


Figure 17: Absent keyphrase generation testing scores on **R@50**. Colors represent the relative performance, normalized per row.

case where beam width equals 1). I observe that compared to the greedy decoding case, the pattern among different ordering strategies appears to be less clear, with the scores distributed more evenly across different settings (concretely, the absolute difference between max average score and min average score is lower).

I suspect that the uniformity among different ordering strategies with beam search may be due to the limitation of the evaluation metric **F_1@10**. The metric **F_1@10** truncates a model's predictions to 10 top-ranked keyphrases. By investigation, I find that during greedy decoding, the number of predictions acts as a dominant factor, this number varies greatly among different ordering. With greedy decoding, **PRES–ABS** can generally predict more phrases than the others, which explains its performance advantage (Figure 18 (a/c)). However, as the beam width increases, all models can predict more than 10 phrases (Figure 18 (b/d)). In this case, the **F_1@10** is contributed more by a model' ability of generating more high quality keyphrases within its top-10 outputs, rather than the amount of predictions. Therefore, the performance gap among ordering strategies is gradually narrowed in beam search. For instance, I observe that the **F_1@10** difference between **PRES–ABS**

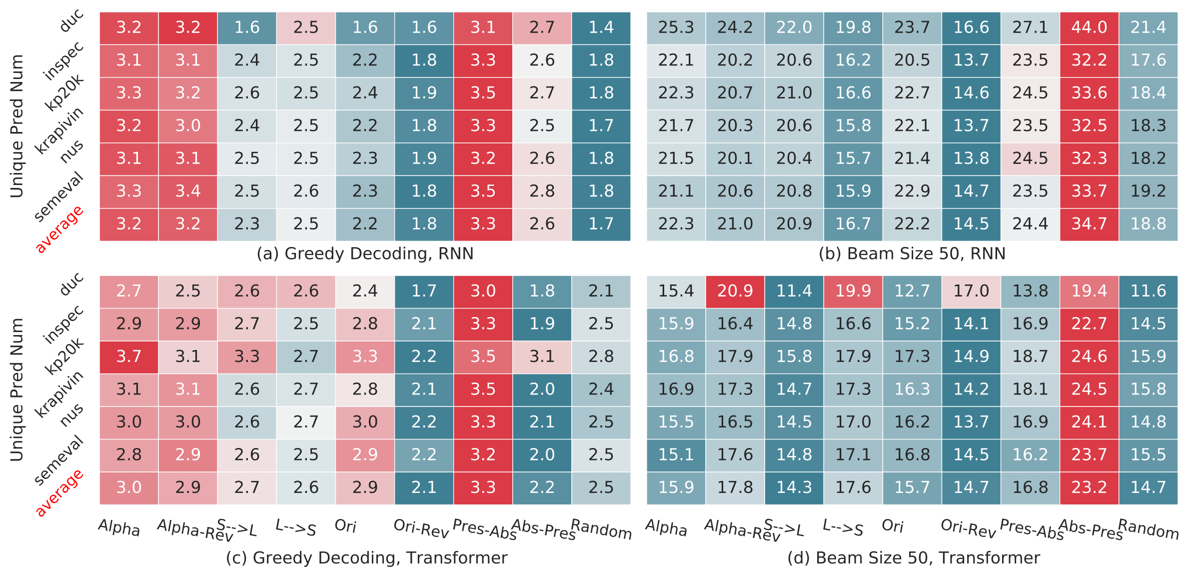


Figure 18: Unique number of present keyphrases generated during test. Colors represent the relative performance, normalized per row. Best checkpoints are selected by $\mathbf{F}_1@\mathcal{O}$ scores on KP20K-VALID.

and **S->L** produced by RNN is 3.5/2.0/1.0/0.2 when beam width is 1/10/25/50.

To validate our assumption, I further investigate the same set of models' performance on $\mathbf{F}_1@\mathcal{O}$, which strictly truncates the generated keyphrase list by the number of ground-truth keyphrases $\mathcal{O}$ (where in most cases $\mathcal{O} < 10$). Under this harsher criterion, a model is required to generate more high-quality keyphrases within its top-$\mathcal{O}$ outputs. From Figure 12 (c/d), I observe that the scores are less uniformly distributed, this indicates a larger difference between different order settings. Among all orders, **ORI** produces best average $\mathbf{F}_1@\mathcal{O}$ with RNN, whereas **ALPHA-REV** and **ORI-REV** produce best average $\mathbf{F}_1@\mathcal{O}$ with TRANSFORMER.

In our curated list of order settings, there are 3 pairs of orderings with reversed relationship (i.e., **S->L** vs **L->S**, **ALPHA** vs **ALPHA-REV**, **ORI** vs **ORI-REV**). Interestingly, I observe that when beam search is applied, these orderings often show a non-negligible score difference with their counterparts. This also suggests that order matters since specific model architecture and training paradigm often has its own preference for the phrase ordering.

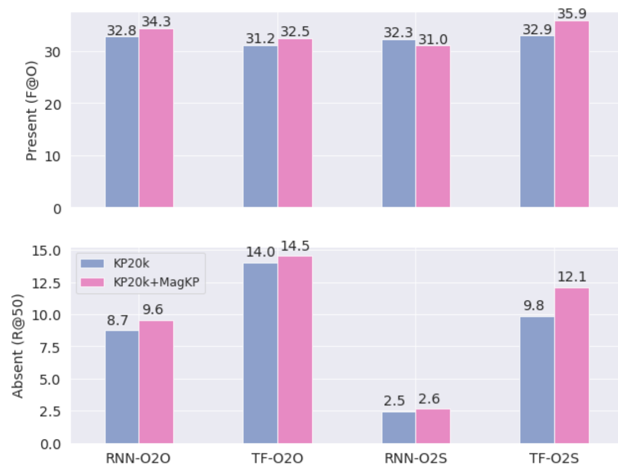


Figure 19: Comparing models trained solely with KP20K against with additional MAGKP data.

It is also worth mentioning that when I manually check the output sequences in the test set produced by **ALPHA** ordering, I notice that the model is actually able to retain alphabetical order among the predicted keyphrases, hinting that a Seq2Seq model might be capable of learning simple morphological dependencies even without access to any character-level representations.

Ordering in Absent KPG

I report the performance of the same set of models on the absent portion of data in Figure 17. Although achieving relatively low **R@50** in most settings, scores produced by various orderings show clear distinctions, normalized heat maps suggest that the rankings among different orderings tend to be consistent across all testing datasets. In general, **PRES–ABS** produces better absent keyphrases across different model architectures.

4.5.2.6 Training with More Data

In this section, we further explore the possibility of improving KPG performance by scaling up the training data. Data size has been shown as one of the most effective factors for training language models [161, 148] but it has yet to be discussed in the context of KPG.

MagKP Dataset

We construct a new dataset, namely MAGKP, based on Microsoft Academic Graph [176]. We filter the original MAG v1 dataset (166 million papers, multiple domains) and only keep papers in *Computer Science* and with at least one keyphrase. This results in 2.7 million data points ($5\times$ larger than KP20K). This dataset remains noisy despite the stringent filtering criteria, this is because 1) the data is crawled from the web and 2) some keywords are labeled by automatic systems rather than humans. This noisy nature brings many interesting observations.

Table 9: Statistics of training datasets KP20K and MAGKP.

Dataset	#Data	#kp	#unique kp	#(word) per kp
KP20K-TRAIN	514K	2.7M	700K	1.92
MAGKP	2.7M	41.6M	6.9M	3.42
MAGKP-LN	522K	2.3M	579K	2.73
MAGKP-Nlarge	1.5M	35.5M	5.8M	3.38
MAGKP-Nsmall	522K	12.2M	2.2M	3.37

General Observations

The first thing we try is to train a KPG model with both KP20K and MAGKP. During training, the two datasets are fed to the model in an alternate manner, we denote this data mixing strategy as ALT. In Figure 19, we compare models' performance when trained on both KP20K and MAGKP against solely on KP20K. We observe the extra MAGKP data brings consistent improvement across most model architecture and training paradigm variants. This suggests that the KPG models discussed in this work can benefit from additional training data. Among all the settings, $\mathbf{F_1@\mathcal{O}}$ of the TRANSFORMER+One2Seq is boosted by nearly 3 points on present KPG, the resulting score outperforms other variants by a significant margin and even surpass a host of state-of-the-art models. Again, the same setting obtains a 2.3 boost of **R@50** score on the absent KPG task, makes TRANSFORMER+One2Seq the setting that benefits the most from extra data. In contrast, the extra MAGKP data provide only marginal improvement to RNN-based models. On present KPG, RNN+One2Seq even has an $\mathbf{F_1@\mathcal{O}}$ drop when trained with more data.

Learning with Noisy Data

To further investigate the performance boost brought by the MAGKP dataset on TRANSFORMER+One2Seq, we are curious to know which portion of the noisy data helped the most. As a naturally way to cluster the MAGKP data, we define the noisiness by the number of keyphrases per data point. As shown in Figure 20, the distribution of MAGKP (black border) covers a much wider spectrum on the x-axis compared to KP20K (red). Because keyphrase labels are provided by human authors,

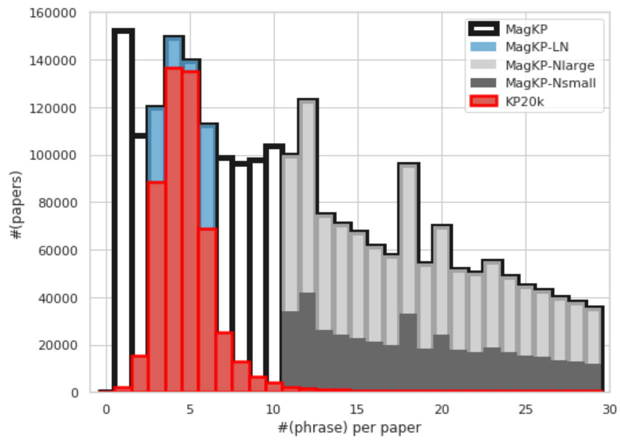


Figure 20: A histogram showing the distribution of #(kp per document) on KP20K, MAGKP and its subsets. Data points with more than 30 keyphrases are truncated.

a majority of its keyphrase numbers lie in the range of [3, 6]; however, only less than 20% of the MAGKP data overlaps with this number distribution.

We thus break MAGKP down into a set of smaller subset: 1) MAGKP-LN is a considerably **L**ess **N**oisy subset that contains data points that have 3~6 phrases. 2) MAGKP-Nlarge is the **N**oisy subset in which all data points have more than 10 keyphrases. 3) MAGKP-Nsmall is a randomly sampled subset of MAGKP-Nlarge with the same size as MAGKP-LN.

We also define a set of data mixing strategies to compare against ALT: ONLY: models are trained solely on a single set (or subset) of data; MX: KP20K and MAGKP (or its subset) are split into shards (10k each) and they are randomly sampled during training; FT: models are pre-trained on MAGKP (or its subset) and fine-tuned on KP20K.

In Figure 21, we observe that none of the MAGKP subsets can match KP20K's performance in the ONLY setting. Because MAGKP-LN and MAGKP-Nsmall share similar data size with KP20K, this suggest the distributional shift between MAGKP and the 6 testing sets is significant. In the MX setting where KP20K is mixed with noisy data, we observe a notable performance boost compared to ONLY (yet still lower than ALT), however, we do not see clear patterns among the 4 MAGKP subsets in this setting. In the FT setting, we observe a surge in scores across all MAGKP subsets. In present KPG, both MAGKP and MAGKP-Nlarge outperform the score achieved in the

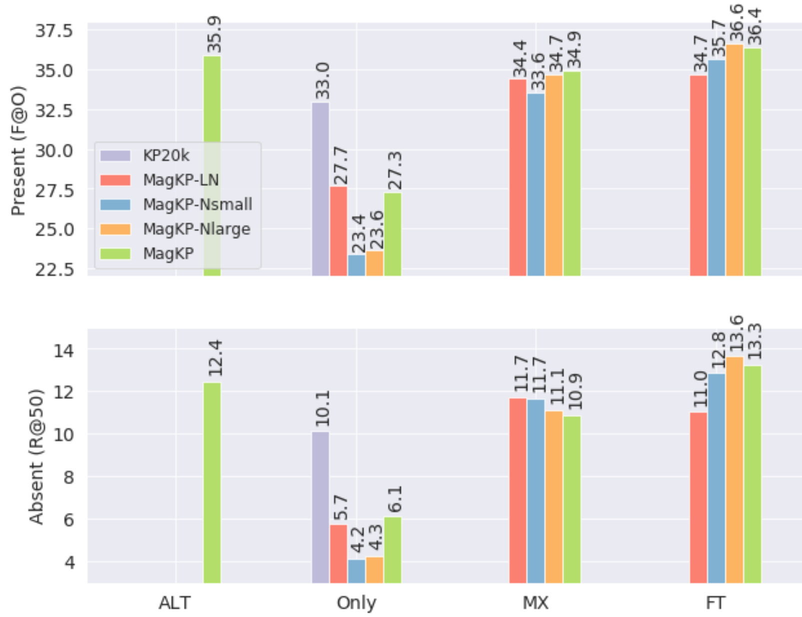


Figure 21: TRANSFORMER+One2Seq trained with KP20K and different subsets of MAGKP, using four data mixing strategy. Scores are averaged over all 6 test sets.

ALT setting; similarly, in absent KPG, MAGKP, MAGKP-Nlarge and MAGKP-Nsmall exceeds the ALT score. This is to our surprise that the subsets considered as noisy provide a greater performance boost, while they perform poorly if "ONLY" trained on these subsets.

To sum up, during our investigation on augmenting KP20K with the noisy MAGKP data, we obtain the best performance from a TRANSFORMER+One2Seq model that pre-trained on MAGKP and then fine-tuned on KP20K, and this performance has outperformed current state-or-the-art models. We conjecture that the performance gain may come from data diversity, because MAGKP contains a much wider distribution of data compared to the author keyword distribution as in KP20K. This inspires us to develop data augmentation techniques to exploit the diversity in unlabeled data.

4.6 SUMMARY

In this section, I introduce two training paradigms for adapting Sequence-to-Sequence Learning to keyphrasification and present empirical studies discussing neural KPG models from various aspects. Results suggest that language generation is a successful paradigm for keyphrasification, by outperforming previous state-of-the-art keyphrase extraction models by significant margins. Besides, the results have shown that, given a carefully chosen architecture and training strategy, a base RNN or TRANSFORMER model can perform comparably with fancy SOTA generation models.

In summary, I list some of the key takeaways as follows:

1. One2Seq excels at present KPG, while One2One performs better on absent KPG.
2. For present KPG, TRANSFORMER performs better on in-distribution data, when distribution or domain shift increases, RNN can outperform TRANSFORMER.
3. On absent KPG, TRANSFORMER is the clear winner.
4. For One2Seq, target ordering is important in greedy decoding (with **PRES-ABS** being an overall good choice).
5. The effect of target ordering tends to diminish when beam search is performed.
6. Copy mechanism helps present prediction while worsening absent performance.
7. Larger beam width is beneficial, especially for absent KPG. However, on present KPG tasks, the benefit is diminished past a certain point and thus computational efficiency needs to be carefully considered.

5.0 GENERATING KEYPHRASES WITH ENHANCED DIVERSITY (KPGEN-ONE2SEQ-DIV)

5.1 MOTIVATION

`One2Seq` models are expected to capture relationships among keyphrases automatically and avoid generating duplicate phrases. However, in practice, a particular issue observed during error analysis is that the model tends to produce identical tokens following the delimiter token. For instance, suppose a target sequence contains n delimiter tokens at time-steps $t_1, \ldots, t_n$. During training, the model is incentivized to generate the same delimiter token at these time-steps, resulting in significant homogeneity in the corresponding decoder states $h_d^{t_1}, \ldots, h_d^{t_n}$. Consequently, when these states are used as inputs at time-steps immediately following the delimiter, the decoder tends to produce highly similar distributions over the subsequent tokens, leading to the decoding of identical tokens.

In reality, for a given source text, it is expected that most keyphrases refer to specific semantics to avoid redundancy, as human annotators typically assign keyphrases that each represent distinct aspects or topics of the text. However, under the setting of basic `One2Seq` models, the information of individual keyphrases, as well as the global structural information including their correlations and the ending state, is mixed into the hidden state of the model. Therefore, it is essential to develop techniques to disentangle the encoded information in the model. These techniques would not only encourage the model to learn relatively isolated representations for individual phrases but also assist it in better controlling the generation process of both local (pertaining to individual phrases) and global structure (pertaining to multiple phrases as a set).

Furthermore, recalling the critical characteristic of keyphrasification discussed in Chapter 3, which is coverage, it is imperative that the generated phrases collectively cover the most important information points from the source text. Thus, additional model enhancements could be devised to further enhance the model's ability to understand the global structure.

In the following sections, I will propose two components for `One2Seq` models aimed at promoting the diversity of generated phrases.

5.2 METHODS

5.2.1 Semantic Coverage

I first propose a mechanism called *semantic coverage* that focuses on the semantic representations of generated phrases. Specifically, I introduce another uni-directional recurrent model $\mathrm{GRU}_{\mathrm{SC}}$ (dubbed *target encoder*) which encodes decoder-generated tokens y^{τ}, where $\tau \in [0, t)$, into hidden states h^{t}_{SC}. This state is then taken as an extra input to the decoder GRU, modifying equation of the decoder GRU to:

$$h^{t}_{dec} = \mathrm{GRU}_{dec}(\langle x^{t}_{dec}, h^{t}_{\mathrm{SC}} \rangle, h^{t-1}_{dec}). \tag{22}$$

If the target encoder were to be updated with the training signal from generation (i.e., backpropagating error from the decoder GRU to the target encoder), the resulting decoder is essentially a 2-layer GRU with residual connections. Instead, inspired by previous representation learning works [116, 198, 81], I train the target encoder in an self-supervised fashion (Figure 22). Specifically, due to the autoregressive nature of the RNN-based decoder, I follow Contrastive Predictive Coding (CPC) [198], where a Noise-Contrastive Estimation (NCE) loss is used to maximize a lower bound on mutual information. That is, the target encoder's final hidden state vector h^{M}_{SC} is extracted, where M is the length of target sequence, and taken as a general representation of the target phrases. Next, the model is trained by maximizing the mutual information between these phrase representations and the final state of the source encoder h^{T}_{enc} as follows. For each phrase representation vector h^{M}_{SC}, I take the encodings $H^{T}_{enc} = \{h^{T}_{enc,1}, \ldots, h^{T}_{enc,N}\}$ of N different source texts, where $h^{T}_{enc,true}$ is the encoder representation for the current source text, and the remaining $N-1$ are negative samples (sampled at random) from the training data. The target encoder is trained to minimize the classification loss:

$$\begin{aligned} \mathcal{L}_{\mathrm{SC}} &= -\log \frac{g(h^{T}_{enc,true}, h^{M}_{\mathrm{SC}})}{\sum_{i \in [1,N]} g(h^{T}_{enc,i}, h^{M}_{\mathrm{SC}})}, \\ g(h_a, h_b) &= \exp(h_a^{\top} B h_b) \end{aligned} \tag{23}$$

where B is bi-linear transformation.

The motivation here is to constrain the overall representation of generated keyphrase to be semantically close to the overall meaning of the source text. With such representations as input to the decoder, the semantic coverage mechanism can potentially help provide useful keyphrase information and guide generation.

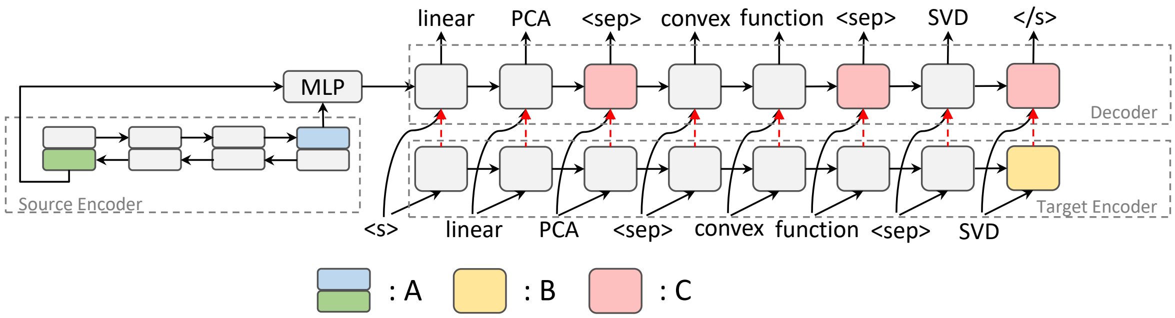


Figure 22: The architecture of the proposed model for improving keyphrase diversity. A represents last states of a bi-directional source encoder; B represents the last state of target encoder; C indicates decoder states where target tokens are either delimiters or end-of-sentence tokens. During orthogonal regularization, all C states are used; during target encoder training, the mutual information between states A with B is maximized. Red dash arrow indicates a detached path, i.e., no back-propagation through such path.

5.2.2 Orthogonal Regularization

I also propose orthogonal regularization, which explicitly encourages the delimiter-generating decoder states to be different from each other. This is inspired by [20], who use orthogonal regularization to encourage representations across domains to be as distinct as possible. Specifically, the decoder hidden states corresponding to delimiters are stacked together to form a matrix $H = \langle h_{dec}^{t_1}, \ldots, h_{dec}^{t_n} \rangle$ and use the following equation as the orthogonal regularization loss:

$$\mathcal{L}_{\text{OR}} = \left\| H^\top H \odot (\mathbf{1} - I_n) \right\|_2 , \tag{24}$$

where $H^\top$ is the matrix transpose of H, I_n is the identity matrix of rank n, $\odot$ indicates element wise multiplication, $\|M\|_2$ indicates L^2 norm of each element in a matrix M. This loss function prefers orthogonality among the hidden states $h_{dec}^{t_1}, \ldots, h_{dec}^{t_n}$ and thus improves diversity in the tokens following the delimiters.

5.2.3 Training Loss

I adopt the widely used negative log-likelihood loss for training the proposed models, denoted as $\mathcal{L}_{\text{NLL}}$. The overall loss I use for optimization is:

$$\mathcal{L} = \mathcal{L}_{\text{NLL}} + \lambda_{\text{OR}} \cdot \mathcal{L}_{\text{OR}} + \lambda_{\text{SC}} \cdot \mathcal{L}_{\text{SC}}, \tag{25}$$

where λ_{OR} and λ_{SC} are hyper-parameters.

5.3 IMPLEMENTATION DETAILS

Implementation details of the proposed models are as follows. In all experiments, the word embeddings are initialized with 100-dimensional random matrices. The number of hidden units in both the encoder and decoder GRU are 150. The number of hidden units in target encoder GRU is 150. The size of vocabulary is 50,000. In all experiments, I use a dropout rate of 0.1.

During negative sampling, 16 examples are randomly sampled from the same batch, thus target encoding loss in Equation 23 is a 17-way classification loss. Hyper-parameters λ_{OR} and λ_{SC} in Equation 25 are selected from [0.01, 0.03, 0.1, 0.3, 1.0] based on validation sets. I use *Adam* [99] as the step rule for optimization. The learning rate is $1e^{-3}$.

For exhaustive decoding, I use a beam size of 50 and a maximum sequence length of 40. Both lowercase and stemming are performed on the ground truth and generated keyphrases during evaluation. All the scores reported are from checkpoints with best performances ($\mathbf{F}_1@\mathcal{O}$) on the validation set.

5.4 RESULTS

5.4.1 Main Result

I conduct an ablation experiment to study the effects of orthogonal regularization and semantic coverage mechanism on `One2Seq-Div` ($\lambda_{\text{OR}} = 1.0, \lambda_{\text{SC}} = 0.03$). As shown in Table 10, semantic

Table 10: Ablation study with $\mathbf{F}_1@\mathcal{O}$ scores on five scientific publication datasets.

Model	KP20K	Inspec	Krapivin	NUS	SemEval
One2Seq	31.9	30.7	32.3	38.3	31.0
+ Orth. Reg.	31.1	29.3	31.0	36.5	29.5
+ Sem. Cov.	32.9	32.1	34.5	40.2	32.9
One2Seq-Div	**35.7**	**33.1**	**37.1**	**40.6**	**35.7**

coverage ($\lambda_{\text{SC}} = 0.03$) provides significant boost to One2Seq's performance on all datasets. Orthogonal regularization ($\lambda_{\text{OR}} = 0.3$), hurts performance when is solely applied to One2Seq model. Interestingly, when both components are enabled (One2Seq-Div), the model outperforms One2Seq by a noticeable margin on all datasets, this suggests the two components help keyphrase generation in a synergistic way. One future direction is to apply orthogonal regularization directly on target encoder, since the regularizer can potentially diversify target representations at phrase level, which may further encourage diverse keyphrase generation in decoder.

5.4.2 Visualizing Diversified Generation

To verify our assumption that target encoding and orthogonal regularization help to boost the diversity of generated sequences, I use two metrics, one quantitative and one qualitative, to measure the diversity of generation. I use the default parameters for t-SNE in sklearn (learning rate is 200.0, number of iterations is 1000).

First, I calculate the average *unique* predicted phrases produced by both One2Seq and One2Seq-Div (beam size is 50). The resulting numbers are 20.38 and 89.70 for One2Seq and One2Seq-Div respectively. Second, from the model running on the KP20K validation set, I randomly sample 2000 decoder hidden states at k steps following a delimiter ($k = 1, 2, 3$) and apply an unsupervised clustering method (t-SNE [199]) on them. From Figure 23 I can see that hidden states sampled from One2Seq-Div are easier to cluster while hidden states sampled from One2Seq yield one mass of vectors with no obvious distinct clusters. Results on both metrics suggest target encoding and orthogonal regularization help diversify the generation of the model.

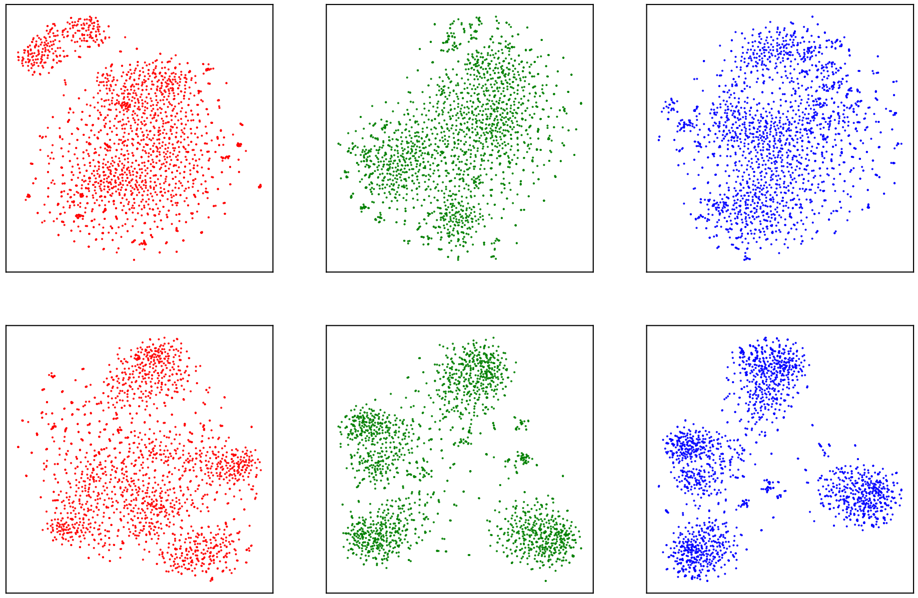


Figure 23: t-SNE results on decoder hidden states. Upper row: `One2Seq`; lower row: `One2Seq-Div`; column k shows hidden states sampled from tokens at k steps following a delimiter.

5.4.3 Qualitative Analysis

To illustrate the difference of predictions between our proposed models, I show an example chosen from the KP20K validation set. In this example, there are 29 ground-truth phrases. Neither of the models is able to generate all of the keyphrases, but it is obvious that the predictions from `One2Seq` all start with "test", while predictions from `One2Seq-Div` are diverse. This to some extent verifies our assumption that without the target encoder and orthogonal regularization, decoder states following delimiters are less diverse.

Table 11: Example from KP20K validation set, predictions generated by One2Seq and One2Seq-Div models.

Source	a visual test development environment for gui systems We have implemented an experimental test development environment (TDE) intended to raise the effectiveness of tests produced for GUI systems, and raise the productivity of the GUI system tester.The environment links a test designer , a test design library, and a test generation engine with a standard commercial capture/replay tool. These components provide a human tester the capabilities to capture sequences of interactions with the system under test (SUT), to visually manipulate and modify the sequences , and to create test designs that represent multiple individual test sequences . Test development is done using a high-level model of the SUT's GUI, and graphical representations of test designs. TDE performs certain test maintenance tasks automatically, permitting previously written test scripts to run on a revised version of the SUT.
One2Seq	test development ; test development environment ; test ; **test generation**
One2Seq-Div	**test generation** ; gui ; **tool** ; **version** ; **capabilities** ; **systems** ; **design** ; test ; **human** ; generation
Groundtruth	engine ; developer ; design ; human ; standardization ; tool ; links ; graphics ; model ; libraries ; replay ; component ; interaction ; product ; development environment ; script ; visualization ; capabilities ; systems ; experimentation ; test designer ; environments ; test generation ; testing ; maintenance ; test maintenance ; version ; effect ; sequence

Table 12: Example from KP20K validation set, and predictions generated by One2Seq and One2Seq-Div models.

Source	Integration of a Voice Recognition System in a Social Robot Human-robot interaction Human-robot interaction (HRI) (1) is one of the main fields in the study and research of robotics. Within this field, dialogue systems and interaction by voice play an important role. When speaking about human-robot natural dialogue we assume that the robot has the capability to accurately recognize what the human wants to transmit verbally and even its semantic meaning, but this is not always achieved. In this article we describe the steps and requirements that we went through in order to endow the personal social robot Maggie , developed at the University Carlos III of Madrid, with the capability of understanding the natural language spoken by any human. We have analyzed the different possibilities offered by current software/hardware alternatives by testing them in real environments. We have obtained accurate data related to the speech recognition capabilities in different environments, using the most modern audio acquisition systems and analyzing not so typical parameters such as user age, gender, intonation, volume, and language. Finally, we propose a new model to classify recognition results as accepted or rejected, based on a second automatic speech recognition (ASR) opinion.This new approach takes into account the precalculated success rate in noise intervals for each recognition framework, decreasing the rate of false positives and false negatives.
One2Seq	voice recognition system ; **social robot** ; **human robot interaction** ; **voice recognition** ; hri ; **speech recognition** ; **automatic speech recognition** ; noise intervals ; noise ; human robot ; automatic speech ; natural language
One2Seq-Div	**human robot interaction** ; **voice recognition** ; **social robotics** ; **social robots** ; integration ; **speech recognition** ; hri ; **social robot** ; robotics ; voice recognition system ; recognition ; **asr** ; **automatic speech recognition** ;
Ground Truth	asr ; automatic speech recognition ; dialogue ; human robot interaction ; maggie ; social robot ; speech recognition ; voice recognition ;

6.0 DOMAIN ADAPTABLE KEYPHRASE GENERATION

6.1 MOTIVATION

In the context of keyphrase generation, the term “domain” refers to the specific subject area or discipline that the text content belongs to. Domains can vary widely, such as computer science, medicine, finance, or literature, each with its own terminology, stylistic features, and thematic content. Domain adaptability in KPG involves the model’s ability to apply or adjust its keyphrase prediction capabilities to texts from different domains, effectively recognizing and generating relevant keyphrases that reflect the distinct characteristics and knowledge of each subject area.

The previously proposed neural models can predict relevant keyphrases, but their performance highly depends on the training on annotated keyphrase datasets. Since most of these datasets are limited to a single domain, it remains unclear how the trained models can be transferred to new domains, especially in a real-world setting. Some existing studies claim their models demonstrate a certain degree of transferability across domains. However, there is a lack of systematic studies on domain transferring KPG, and thus the observations reported in prior works do not support a comprehensive understanding of this topic.

In this section, I would like to look into the problem that how domain difference actually affects the performance of KPG models when transferred across domains. Moreover, if it does pose negative impacts, what is the implication of this observation and can we find methods to alleviate this issue accordingly?

6.2 DOMAIN GAP IN KEYPHRASE GENERATION

Previous studies [135, 221] have touched on how much KPG models can transfer their skills when applied across domains, but not in a systematic way. For instance, [135] show that models trained with scientific paper datasets can generate decent quality keyphrases from news articles, in a zero-shot manner. [221] present that training with open-domain web documents can improve the model's generalizability.

In this subsection, I revisit this topic and try to ground our discussion with thorough empirical results. Specifically, I consider four broadly used datasets in the KPG community: KP20K [135] contains scientific papers in computer science; OPENKP [221] is a collection of web documents; KPTIMES [59] contains a set of news articles; STACKEX [231] are community-based Q&A posts collected from StackExchange. All the four datasets are large enough to train KPG models from scratch. At the same time, the documents in these datasets cover a wide spectrum of domains. I report statistics of these four datasets in Table 13.

Table 13: Statistics of training/testing datasets used in this study. †Only 7.7m papers in MAG-CS 12M have keyphrases and are acutally used in this study.

	#doc	#words in doc	#kp	#unique kp	#kp per doc	#uni kp per doc	#present kp per doc	#absent kp per doc
Training Sets								
Wikipedia	3.2m	-	-	-	-	-	-	-
KP20K	514.2k	161	2.7m	680.1k	5.3	1.3	3.3	1.9
OPENKP	134.9k	1104	294.2k	206.8k	2.2	1.5	2.1	0.0
KPTIMES	259.9k	803	1.3m	104.8k	5.0	0.4	2.4	2.6
STACKEX	299.0k	207	803.9k	8.1k	2.7	0.0	1.6	1.1
MAG-CS 1M	1.0m	151	9.6m	1.7m	9.6	1.7	3.4	6.2
MAG-CS 12M†	12.1m	151	115.9m	14.3m	9.6	1.2	3.4	6.2
Test Sets								
KP20K	19,987	161	105.2k	55.9k	5.3	2.8	3.3	1.9
OPENKP	6,614	894	14.6k	13.6k	2.2	2.0	2.0	0.2
KPTIMES	10,000	804	50.4k	13.9k	5.0	1.4	2.4	2.6
STACKEX	16,000	205	43.1k	4.5k	2.7	0.3	1.6	1.1
JPTIMES	10,000	517	50.3k	9.0k	5.0	0.9	4.0	1.0
DUC-2001	308	701	2.5k	1.8k	8.1	6.0	7.9	0.2

On the model dimension, I consider two model architectures: TF-RAND, a 6-layer encoder-decoder Transformer with random initialization [202]; and TF-BART, a 12-layer Transformer initialized with BART-large [106]. I train the two models on the four datasets individually and subsequently evaluate all the resulting eight checkpoints on the test split of each dataset. As shown in Figure 24, in-domain scores (i.e., trained and tested on the same datasets) are placed along the diagonal, the other elements represent cross-domain testing scores. We can see that both

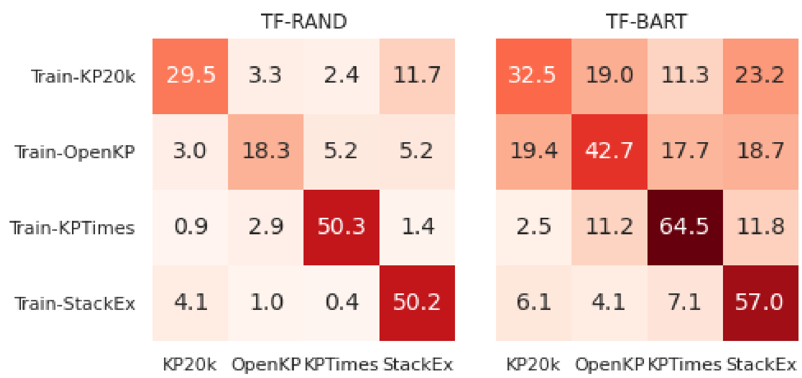


Figure 24: Cross-domain transfer performance of TF-RAND and TF-BART (F@O, the higher the better). Y-axis: training dataset; X-axis: test dataset.

models exhibit a large gap between in-domain and out-of-domain performance. Even though the initialization with BART can alleviate the gap to a certain degree, the difference remains significant.

Table 14: Overlap (%) of unique keyphrases between domains (train split). Numbers are normalized by dividing the diagonal element in its column. For example, the overlap keyphrases between KP20K and STACKEX make a proportion of 0.7% in KP20K and a 55.7% in STACKEX.

	KP20K	OPENKP	KPTIMES	STACKEX
KP20K	100	10.6	3.5	55.7
OPENKP	3.2	100	8.5	33.1
KPTIMES	0.5	4.3	100	6.9
STACKEX	0.7	1.3	0.5	100

Keyphrases are typically concepts or entities that represent important information of a document. The collection of keyphrases in a domain can also be deemed as a representation of domain knowledge. Therefore, to better investigate the domain gaps, I further look into the keyphrase overlap between datasets. As shown in Table 14, only a small proportion of phrases are in common between the four domains. I provide a T-SNE visualization of a set of phrases sampled from these dataset in Figure 25, the phrase clusters present clear domain gaps in their semantic space.

I hypothesize that the domain specific traits in annotated data make models difficult to learn keyphrase patterns in a domain-general sense. Furthermore, humans may label keyphrases under an application-oriented consideration and thus a one-size-fits-all standard for keyphrase annotation may not exist. For example, on StackExchange, users tend to assign common tags to better expose their questions to community experts, resulting in a small keyphrase vocabulary size. On the contrary, the topics are more specialized in scientific papers and authors would emphasize novel concepts in their studies. This may explain the large number of unique keyphrases found in `KP20K`.

6.3 DISENTANGLEMENT OF "KEY" AND "PHRASE"

In §6.2, I empirically show that KPG models do not adequately transfer to out-of-domain data, even initialized with pre-trained language models. However, data annotation for every single domain or application does not seem practical either, due to the high cost and the potential need of domain-specific annotators. This motivates me to look for novel methods that can adapt KPG models to new domains with limited cost (in terms of both data annotation and training cost).

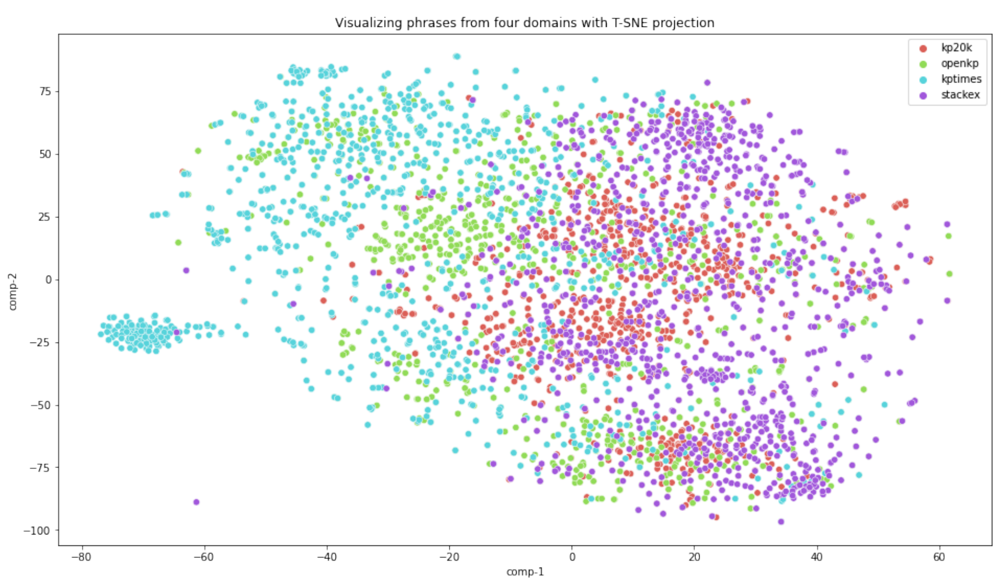


Figure 25: T-SNE visualization of keyphrase representations from four datasets.

Inspired by the understanding discussion in Chapter 3, I attempt to disentangle the properties of a keyphrase as *keyness* [17, 58] and *phraseness* [194]:

Keyness refers to the attribute that how well a phrase represents important information of a piece of text. The degree of keyness can be document dependent and domain dependent. For example, "cloud" is a common keyphrase in Computer Science papers, it is, in most cases, less likely to be important in Meteorology studies. Due to its high dependence on domain-specific information, we believe that the knowledge/notion of keyness is more likely to be acquired from in-domain data.

Phraseness, on the other hand, focuses more on the syntactical aspect. It denotes that given a short piece of text, without even taking into account its context, to what extent it can be grammatically functional as a meaningful unit. Although the majority of keyphrases in existing datasets are noun phrases [39], they can present in variant grammatical forms in the real world [187]. We believe that phraseness can be independent from domains and thus can be obtained from domain-general data.

6.4 METHODOLOGY

I propose a three-stage training procedure in which a model gradually moves its focus from learning domain-general phraseness towards domain-specific keyness, and eventually adapts to a new domain with only limited amount of annotated data. An overview of the proposed pipeline is illustrated in Figure 26. First, with a Pre-Training stage (**PT**), the model is trained with domain-general data to learn phraseness (§6.4.1). Subsequently, in the Domain Adaption stage (**DA**), the model is exposed with *unlabeled* in-domain data. Within a few iterations, the model labels the data itself and use them to gradually adapt to the new domain (§6.4.2). Lastly, in the Fine-Tuning stage (**FT**), the model fully adapts itself to the new domain by leveraging a limited amount of in-domain data with true annotations (§6.4.3). I will describe each of the three stages in detail.

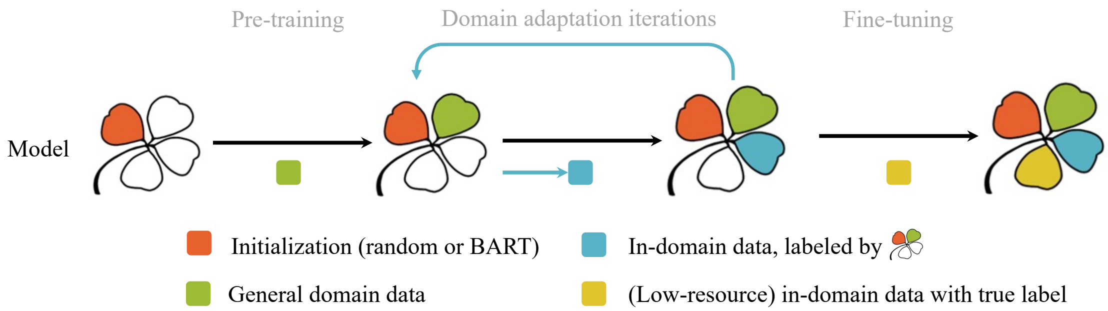


Figure 26: The proposed three-stage pipeline. A model is first pre-trained with general domain data and learns to generate syntactically correct phrases. In the domain adaptation stage, the model adapts to the target domain by training on domain-specific data, where the pseudo labels are generated by the model itself. Finally, I fine-tune the model with limited amount of target domain data with true label, to fully accomplish domain adaptation.

6.4.1 Domain-General Phrase Pre-training

The first training stage aims to capture the phraseness in general, I leverage the Wikipedia data and community labeled phrases from the text. Wikipedia is an open-domain knowledge base that contains rich entity-centric annotations, its articles cover a wide spectrum of topics and domains and thus it has been extensively used as a resource of distant supervision for NLP tasks related to entities and knowledge [67, 224, 222]. In this work, I consider four types of markup patterns in Wikipedia text to form distant keyphrase labels:

- in-text phrases with special formatting (italic, boldface, and quotation marks);
- in-text phrases with wikilinks (denoting an entity in Wikipedia);
- “see also” phrases (denoting related entities);
- “categories” phrases (denoting superordinate entities).

Although the constructed targets using the above heuristics can be noisy if considering the keyness aspect, I will show that they work sufficiently for training general phrase generation models.

Given a piece of Wikipedia text t and a set of community labeled phrases, I convert this data point to the format of `One2Seq` as described in §4.3.3. In practice, the number of phrases within t

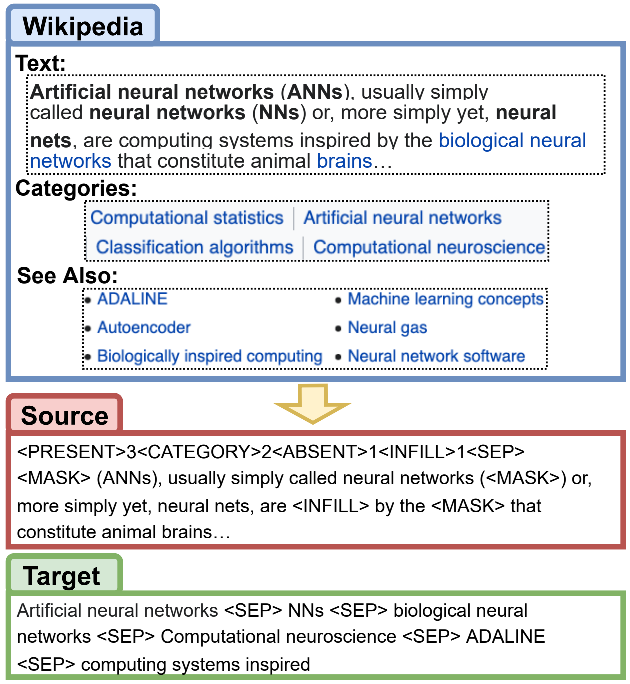


Figure 27: Illustration of processing Wikipedia to source-target pairs in domain general phrase pre-training.

can be large and thus I sample a subset from them to form the target. all the phrases appear in t are grouped as present candidates, the rest (e.g., "see-also" and categories) are grouped as absent candidates. Additionally, I take several random spans from t as infilling candidates (similar as [162]) for robustness. Finally, I sample a few candidates from each group and concatenate them as the final target sequence.

On the source side, I prepend a string suggesting the cardinality of phrases in each target group to the beginning of t. I also corrupt the source sequence by replacing a small proportion of present and infilling phrases with a special token `[MASK]`, expecting to improve models' robustness [162]. An example of a processed Wikipedia data instance is shown in Figure 27.

Trained with this data, I expect a model to become a general phrase generator — given a source text, the model can generate a sequence of phrases, regardless the specific domain a text belongs to.

6.4.2 Domain Adaption with Transfer Labeling

In the second stage, I aim to expose the model with data from a domain of interest, so it can learn the notion of domain-specific keyness. I propose a method, namely General-to-Specific Transfer Labeling , which does not require any in-domain annotated data. Transfer labeling can be considered as a special self-training method [225, 41, 143], where the key notion is to train a model with its own predictions iteratively.

Distinct from common practice of self-training where initial models are bootstrapped with annotated data, transfer labeling regards the domain-general model from the pre-training stage 6.4.1 as a qualified phrase predictor. I directly transfer the model to documents in a new domain to predict pseudo labels. The resulting phrases, paired with these documents, are used to tune the model so as to adapt it to the target domain distribution. Note that this process can be run iteratively, to gradually adapt models to target domains.

6.4.3 Low-resource Fine-Tuning

In the third stage, I expose the model to a small amount of in-domain data with annotated keyphrases. This aims to help the model fully adapt to the new domain and reduce the bias caused by noisy labels from previous stages.

6.5 EXPERIMENTS

I reuse the model architecture described in §6.2 throughout this paper. And most models apply a single iteration of transfer labeling. I will discuss the effect of multi-iteration transfer labeling in §6.5.3.5.

6.5.1 Implementation Details

Most experiments make use of four V100 GPUs. I elaborate the training hyper-parameters for reproducing our results in Table 15 and 16. I use beam search to produce multiple keyphrase predictions (beam width of 50, max length of 40 tokens). Testing scores are reported using the best checkpoints, which achieve best performance on valid set (2,000 data instances for all domains).

Phrase masking ratio denotes for *p%* of target phrases, replacing their appearances in the source text with a special token `[PRESENT]`.

Random span ratio denotes replacing *p%* of words in the source text with a special token `[MASK]`.

Table 15: Training hyperparameters for TF-RAND. *FT denotes the fine-tuning stage in cases of PT+FT or PT+DA+FT. Empty cell means it is the same as the leftmost value.

Hparam	**PT**	**DA**	**PT+DA/PT+DA$_{\text{MAG-CS}}$**	**FT 100/1k/10k**	***FT 100/1k/10k**
Max source length	512				
Max target length	128				
Max phrase number	16	8			
Max phrase length	16	8			
Phrase masking rate	0.1				
Random span ratio	0.05				
Batch size	≈80	100	100	100	100
Learning rate	3e-4	3e-4	1e-5	3e-4	1e-5
Number of steps	200k	40k	20k/200k	2k/4k/8k	1k/2k/4k
Warmup steps	10%				
Learning rate decay	linear				
Optimizer	Adam				
Adam β_1	0.9				
Adam β_2	0.998				
Adam epsilon	1e-6				
Max gradient norm	2.0	2.0	1.0	1.0	1.0
Dropout	0.1				
BPE Dropout	0.1	0.0	0.0	0.0	0.0
Label smoothing	0.1				
Save checkpoint freq	ending step	ending step	ending step	100/200/400	50/100/200

6.5.2 Datasets and Evaluation Metric

I consider the same four large-scale KPG datasets as described in §6.2, but instead of training models with all annotated document-keyphrases pairs, I take a large set of unannotated documents from each dataset for domain adaptation, and a small set of annotated examples for few-shot fine-tuning. Specifically, in the pre-training stage (**PT**), I use the 2021-05-21 release of English Wikipedia dump and process it with wikiextractor package, which results in 3,247,850 passages. In the domain adaptation stage (**DA**), for each domain, I take the first 100k examples from the training split (without keyphrases), and apply different strategies to produce pseudo labels and subsequently

Table 16: Training hyperparameters for TF-BART. Empty cell means it is the same as the leftmost value.

Hparam	PT	DA	PT+DA/PT+DA$_{\text{MAG-CS}}$	FT 100/1k/10k
Max source length	512			
Max target length	256	256	256	128
Max phrase number	16			
Max phrase length	6	8	8	8
Phrase masking rate	0.1			
Random span ratio	0.05			
Batch size	256	256	256	16
Learning rate	1e-5			
Number of steps	40k	5k	5k/20k	2k/4k/8k
Warmup steps	2.4k	300	300/1.2k	200/400/800
Learning rate decay	linear			
Optimizer	Adam			
Adam β_1	0.9			
Adam β_2	0.98	0.98	0.98	0.999
Adam epsilon	1e-8			
Weight decay	0.01			
Max gradient norm	1.0	-	-	0.1
Dropout	0.1			
Label smoothing	0.1			
Save checkpoint freq	ending step	ending step	100/200/400	50/100/200

train the models. In the fine-tuning stage (**FT**), I take the first 100/1k/10k annotated examples (document-keyphrases pairs) from the training split to train the models.

I follow previous studies to split training/validation/test sets, and report model performance on test splits of each dataset. A common practice in KPG studies is to evaluate the model performance on present/absent keyphrases separately. However, the ratios of present/absent keyphrases differ drastically among the four datasets (e.g. OPENKP is strongly extraction-oriented). Since I aim to improve the model's out-of-domain performance in general regardless of the keyphrases being present or absent, I follow [7] and simply evaluate present and absent keyphrases altogether. I report the F@O scores [231] between the generated keyphrases and the ground-truth. This metric requires systems to model the cardinality of predicted keyphrases themselves.

6.5.3 Results and Analyses

6.5.3.1 Zero-shot Performance

I first investigate how well models can perform after the pre-training stage, without utilizing any in-domain annotated data. Since Wikipedia articles contain a rather wide range of phrase types, I expect models trained on this data are capable of predicting relevant and well-formed phrases from documents in general. My models' testing scores are shown in the first row of Table 17 and 18, where only **PT** is checked. We can see that pre-training with Wikipedia data can provide decent zero-shot performance in both settings, i.e., model is initialized randomly (Table 17) and with pre-trained language models (18). Both settings achieve the same average F@O score of 12.2, which evinces the feasibility of using **PT** model to generate pseudo labels for further domain adaptation. The scores also suggest that at the pre-training stage, the BART model (with pre-trained initialization and more parameters) does not present an advantage in comparison to a smaller model trained from scratch.

6.5.3.2 Domain Adaptation Strategies

I compare transfer labeling (**TL**, proposed in §6.4.2) with two unsupervised strategies: (1) Noun Phrase (**NP**) and (2) Random Span (**RS**). For NP, I employ SpaCy [84] to POS-tag source texts and extract noun phrases based on regular expressions. For RS, I follow [162], extracting random spans as targets and masking them in the source text. For TL, all pseudo phrases are generated by a **PT** model in a zero-shot manner (with greedy decoding).

As shown in Figure 28, in the single strategy setting, RS performs the best among the three strategies and TL follows. I speculate that RS models are trained to predict randomly masked spans based on their context, and this results in the best generalization among the three. As for the NP strategy, since targets are only noun phrases appear in the source text, the models may have the risk of overfitting to recognize a subset of possible phrases. TL lies in between the two discussed strategies, the generated pseudo labels contain both present and absent phrases, and thanks to the **PT** model trained with Wikipedia data, the generated targets can contain many phrase types beyond noun phrases.

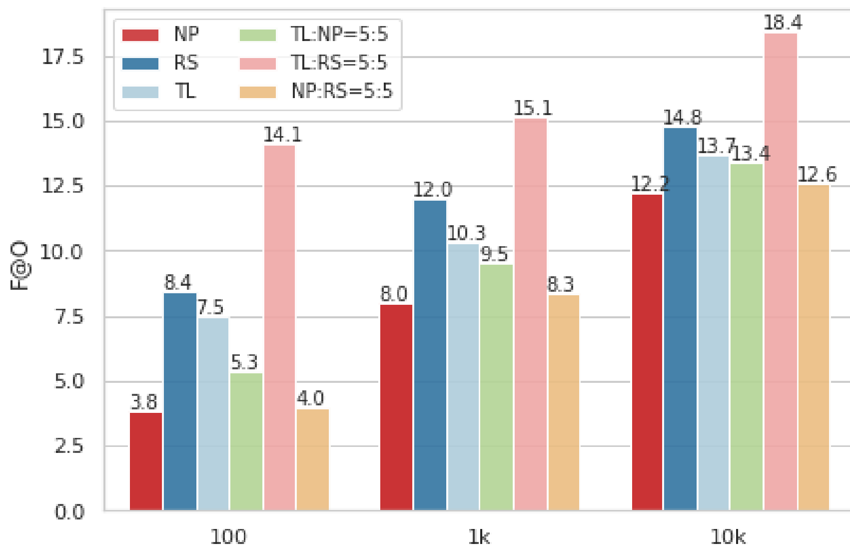


Figure 28: Comparison of different strategies for domain adaptation with TF-RAND. **TL**: Transfer Labeling. **NP**: Noun Phrases. **RS**: Random Span.

I further investigate the performance gap between RS and TL. On KP20K, the **PT** model can generate 5.1 present and 2.6 absent keyphrases on average. The generated pseudo labels, albeit of good quality, are always fixed during the training. This is due to the deterministic nature of the **PT** model, which may cause overfitting and limit the model's generalizability. In contrast, random spans in RS are dynamically generated, therefore a model can learn to generate different target phrases even the same documents appear multiple times during training. This motivates us to investigate if these strategies can be synergistic by combining them. As shown in Figure 28, I observe that combining TL and RS can lead to a significant improvement over all other strategies, indicating that these two strategies are somewhat complementary and thus can be used together in domain adaption. In the rest of the paper, I by default combine TL and RS in the domain adaptation stage, by taking equal amount of data from both sides.

It is worth noting that, if I apply domain adaptation with the TL+RS mixing strategy and evaluate models without any fine-tuning (2nd row in Table 17/18), we can observe a clear drop in the performance of randomly initialized model (Table 17). I believe it is because using random spans for targets worsens the phraseness of the predictions. BART initialized models, on the other hand, show robust performance against these noisy targets.

6.5.3.3 Performance in Low-Data Setting

As described in §6.5.2, I use 100/1k/10k in-domain examples with gold standard keyphrases to fine-tune the model. To investigate the necessity of the **PT** and **DA** stages given the **FT** stage, I conduct a set of ablation experiments, skipping some of the training stages in the full pipeline.

Table 17: Zero-shot and low-data results obtained by TF-RAND. The best average score in each column is **boldfaced**.

	PT	DA	FT	KP20K	OPENKP	KPTIMES	STACKEX	Avg
TF-IDF				6.3	-	-	5.2	-
0-shot	x			15.0	10.0	9.1	14.8	12.2
	x	x		11.2	4.6	7.7	4.3	6.9
100-shot			x	0.5	0.2	2.4	5.1	2.1
		x	x	14.1	5.6	5.3	11.7	9.2
	x		x	14.5	20.1	22.6	13.0	17.6
	x	x	x	16.7	24.4	22.0	18.4	20.4
1k-shot			x	0.5	0.6	5.4	7.0	3.4
		x	x	15.0	8.6	8.9	15.4	12.0
	x		x	17.6	25.5	30.5	21.1	23.7
	x	x	x	19.7	28.0	30.7	26.3	26.2
10k-shot			x	3.4	1.5	19.2	20.8	11.3
		x	x	16.5	13.1	13.4	23.4	16.6
	x		x	20.6	30.6	38.6	31.4	30.3
	x	x	x	22.1	31.6	36.7	34.7	31.3
Avg			x	1.5	0.8	9.0	11.0	5.6
		x	x	15.2	9.1	9.2	16.8	12.6
	x		x	17.6	25.4	**30.6**	21.8	23.8
	x	x	x	**19.5**	**28.0**	29.8	**26.5**	**25.9**

I start with discussing the results of randomly initialized models (Table 17). **FT-only**: in the case where models are only fine-tuned with a small subset of annotated examples, models perform rather poorly on all datasets, especially on KP20K and OPENKP, where more unique target phrases are involved. **DA+FT**: different from the previous setting, here all models are first trained with 100k pseudo labeled in-domain data points. I expect these pseudo labeled data to improve models on both phraseness and keyness dimensions. Indeed, I observe DA+FT leads to a large performance

Table 18: Zero-shot and low-data results of TF-BART model.

	PT	DA	FT	Avg
0-shot	x			12.2
	x	x		12.0
Average of few-shot (100/1k/10k)			x	36.2
		x	x	36.3
	x		x	**36.6**
	x	x	x	36.1

Table 19: Average scores (over 4 datasets) with different amount of transfer labeled data for domain adaptation. All models are trained through three stages. The best score in each block is **boldfaced**.

Model	DA Data	100-shot	1k-shot	10k-shot
TF-RAND	KP20K 100k	16.7	19.7	22.1
	MAG-CS 1m	16.8	19.4	21.8
	MAG-CS 12m	**17.6**	**20.4**	**22.8**
TF-BART	KP20K 100k	22.2	25.3	28.4
	MAG-CS 1m	22.3	**25.4**	28.4
	MAG-CS 12m	**22.5**	**25.4**	**28.6**

boost in almost all settings. This suggests the feasibility of leveraging unlabeled in-domain data using the proposed adaptation method (TL+RS). **PT+FT**: the pre-training stage provides a rather significant improvement in all settings, averaging over datasets and k-shot settings, PT+FT (23.8) nearly doubles the performance of DA+FT (12.6). This observation indicates that the large-scale pre-training with domain-general phrase data can be beneficial in various down-stream domains, which is consistent with prior studies for text generation pre-training. **PT+DA+FT**: I observe a further performance boost when both PT and DA stages are applied before FT. This to some extent verifies our design that PT and DA can guide the models to focus on different perspectives of KPG and thus work in a complementary manner.

I also investigate when the model is initialized with a pre-trained large language model, i.e., BART [106]. Due to space limit, I only report models' average scores (over the four datasets, and over the k-shot settings) in Table 18. I observe that in the pipeline, the fine-tuning stage provides TF-BART the most significant performance boost — the average score is tripled, compared to the

0-shot settings, even performing solely the fine-tuning stage. This may be because the BART model was trained on a much wider range of domains of data (compared to Wikipedia, which is already domain-general), so it may have already contained knowledge in our four testing domains. However, the auto-regressive pre-training of BART does not train particularly on the KPG task. This explains why it requires the BART model to fine-tune on KPG data to achieve higher performance. The above assumption can also be support by further observations in Table 18. Results suggest that the DA stage is not notably helpful to TF-BART's scores, and the PT stage, on the other hand, seems to contribute to a better score. I believe this is because the quality difference between labels used in these two stages: PT uses community-labeled phrases (high phrase quality but domain-general) and DA uses labels generated by the model itself (no guarantee on phrase quality but closer to target domains). Since TF-BART only needs specific knowledge about the KPG task, the PT stage can therefore be more helpful.

I run Wilcoxon signed-rank tests on the results of Table 17, and it shows all differences between neighboring experiments (e.g. PT+FT vs. PT+DA+FT, both trained with KP20k and 10k-shot) are significant ($p < 0.05$). For Table 18, the improvement of PT+FT over the other three settings is also significant.

6.5.3.4 Scaling the Domain Adaptation

One advantage of self-labeling is the potential to leverage large scale unlabeled data in target domains. I also investigate this idea and build a large domain adaptation dataset by pairing an unlabeled dataset with pseudo labels produced by a **PT** model. To this end, I resort to the MAG (Microsoft Academic Graph) dataset [177] and collect paper titles and abstracts from 12 million scientific papers in the domain of Computer Science, filtered by 'field of study'. The resulting subset MAG-CS is supposed to be in a domain close to KP20K, yet it may contain noisy data points due to errors in the MAG's data construction process. I follow the same experiment setting as reported in the above subsections, except that in the DA stage we either use 1 million or 12 million pseudo-labeled MAG data points for domain adaptation. I train the models with the PT+DA+FT pipeline and report models' scores on KP20K test split.

As shown in Table 19, compared to our default setting which uses 100k unlabeled KP20K data points for domain adaptation, larger scale domain adaptation data can indeed benefit model performance — models adapted with MAG-CS 12m documents show consistent improvements. However, the MAG-CS 1m data (still 10 times the size of KP20K) does not show clear evidence being helpful. I suspect the distribution gap between the domain adaptation data (i.e., MAG-CS) and the testing data (i.e., KP20K) may have caused the extra need of generalization. Therefore, the MAG-CS 12m data may represent a data distribution that has more overlap with KP20K and thus being more helpful. I also observe that models initialized with BART are more robust against such a distribution gap, on account of BART's pre-training with large scale of text in general domain.

6.5.3.5 Multi-iteration Domain Adaptation

Prior self-training studies have demonstrated the benefit of multi-iterations of label propagation [196, 107]. I conduct experiments to investigate its effects on KPG. Specifically, I first pre-train a TF-RAND model using Wikipedia data as in previous subsections. Then, I repeatedly perform the domain adaptation stage multiple times. In each iteration, the model produces pseudo labels from the in-domain documents and then train itself with this data. Finally, I fine-tune the model with 10k annotated data points, and report its test scores on KP20K. I consider two datasets, KP20K and MAG-CS 1m, as the in-domain data for domain adaptation. As illustrated in Figure 29, the TF-RAND model can gradually gain better test performance by iteratively performing domain adaptation using both datasets. Due to limited computing resources, I set the maximum number of iterations to 10. But the trend suggests that models may benefit from more DA iterations.

6.6 SUMMARY

In this study, I investigate domain gaps in the KPG task that hinder models from generalization. I attempt to alleviate this issue by proposing a three-stage pipeline to strategically enhance models' abilities on keyness and phraseness. Essentially, I consider phraseness as a domain-general property and can be acquired from Wikipedia data as distant supervision. Then I use self-labeling to distill the phraseness into data in a new domain, and the resulting pseudo labels are used for domain

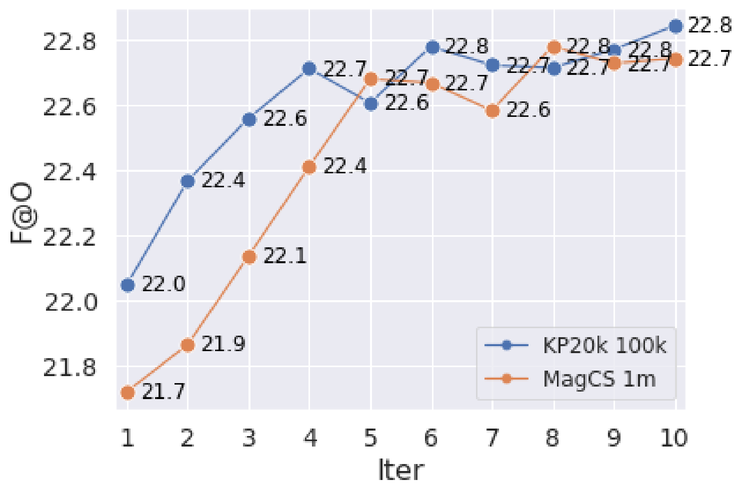


Figure 29: Trend of 10k-shot performance on KP20K with iterative self-labeling for 10 iterations.

adaptation, as the labels can reflect the keyness and phraseness of the new domain. Finally, I fine-tune the model with limited amount of target domain data with true labels. By taking the advantage of open-domain knowledge on the web, I believe this general-to-specific paradigm is generic and can be applied to a wide variety of machine learning tasks. As a next step, I plan to employ the proposed method for text classification and information retrieval, to see whether the domain-general phrase model can produce reliable class labels and queries for domain adaptation.

7.0 EXPLORING KEYPHRASIFICATION WITH LARGE LANGUAGE MODELS

7.1 MOTIVATION

The advent of large language models (LLMs) with billions of parameters, pre-trained on extensive datasets, has revolutionized the field of NLP, showcasing extraordinary capabilities and emergent behaviors unprecedented in earlier models [23, 195, 214]. Studies [16, 103] evaluating LLMs have demonstrated their potential to match or even surpass traditional models in tasks like question answering and summarization. This raises the intriguing question of whether LLMs can effectively generate keyphrases from text and if they exhibit advanced phrase understanding abilities akin to the Chain-of-Thought reasoning observed in complex tasks.

To investigate these aspects, I plan to conduct empirical research on the performance of OpenAI's LLMs in keyphrase generation, aiming to gauge their alignment with human-like understanding of keyphrases. This exploration will involve testing LLMs against datasets crafted to assess semantic understanding, keyness, and phraseness, as discussed in Chapter 3. The outcomes of this research will provide insights into LLMs' strengths and weaknesses in keyphrasification, guiding future directions in keyphrase generation studies.

7.2 EMPIRICAL ANALYSIS OF LLMS IN KEYPHRASE GENERATION

7.2.1 Experiment Setting

This section presents an evaluation of various large language models on three keyphrase generation datasets: `INSPEC`, `SEMEVAL`, and `DUC-2001`, focusing on the generation of present keyphrases to maintain a consistent comparison with extractive methods.

As multiple prior works [103, 180, 179, 123] have conducted similar empirical studies on OpenAI models, I mainly reference the results from [179], which mainly compares GPT-3.5-turbo to various models specifically trained for keyphrase prediction. Additionally, I incorporate results

Table 20: Prompts used to predict keyphrases from the text document.

Model	**Prompt**
`Prompt 1 (kw)`	`Extract keywords from this text:` [DOCUMENT]
`Prompt 2 (kp)`	`Extract keyphrases from this text:` [DOCUMENT]
`Prompt 3 (10kw)`	`Extract 10 keywords from the text below:` [DOCUMENT]
`Prompt 4 (10kp)`	`Extract 10 short keyphrases (no longer than 5 words) from the text below:` [DOCUMENT]

from two other LLMs: `text-davinci-003` and `GPT-3.5-turbo-instruct`, updating the performance metrics for `GPT-3.5-turbo` (API name *gpt-3.5-turbo-1106*, version *20231106*).

These models, fine-tuned on instructional data [149], excel in various language generation tasks, though they are not exclusively designed for conversational alignment. The prompts utilized across all models are detailed in Table 20.

A crucial aspect of the evaluation is data post-processing, as LLMs may present keyphrases in differing formats, such as starting with a phrase like "the keywords extracted from the text are:" or using various delimiters (e.g., commas, semicolons, new lines). Therefore, it is essential to normalize the outputs for a fair and consistent comparison across models.

7.2.2 Results

The detailed results are shown in Table **??**, comparing multiple extractive baselines and a generative TRANSFORMER model trained on KP20K and MAGKP. Overall, LLMs showed superior performance on the INSPEC dataset but were outpaced on DUC-2001 and SEMEVAL. This could be attributed to the extensive availability of scientific paper metadata online, likely making academic keyphrases a significant component of OpenAI LLMs' pre-training data. The varied use of delimiters like commas, semicolons, and line breaks in the models' outputs suggests a pre-training bias towards certain output formats.

The LLMs' better performance on INSPEC, which consists of Computer Science papers

Table 21: Performance on DUC-2001, INSPEC and SEMEVAL test sets. The best extraction and generation scores in each column are in bold.

MODEL	DUC2001			INSPEC			SEMEVAL2010		
	F1@5	F1@10	F1@15	F1@5	F1@10	F1@15	F1@5	F1@10	F1@15
STATISTICAL EXTRACTION MODELS									
TF-IDF [91]	9.21	10.63	11.06	11.28	13.88	13.83	2.81	3.48	3.91
YAKE [25]	12.27	14.37	14.76	18.08	19.62	20.11	11.76	14.4	15.19
GRAPH-BASED EXTRACTION MODELS									
TextRank [138]	11.80	18.28	20.22	27.04	25.08	36.65	3.80	5.38	7.65
SingleRank [206]	20.43	25.59	25.70	27.79	34.46	36.05	5.90	9.02	10.58
TopicRank [19]	21.56	23.12	20.87	25.38	28.46	29.49	12.12	12.90	13.54
PositionRank [53]	23.35	28.57	28.60	28.12	32.87	33.32	9.84	13.34	14.33
MultipartiteRank [18]	23.20	25.00	25.24	25.96	29.57	30.85	12.13	13.79	14.92
EMBEDDING-BASED EXTRACTION MODELS									
EmbedRankd2v [13]	24.02	28.12	28.82	31.51	37.94	37.96	3.02	5.08	7.23
EmbedRanks2v [13]	27.16	31.85	31.52	29.88	37.09	38.40	5.40	8.91	10.06
KeyGames [169]	24.42	28.28	29.77	32.12	**40.48**	40.94	11.93	14.35	14.62
SIFRank [189]	24.27	27.43	27.86	29.11	38.80	39.59	-	-	-
SIFRank+ [189]	**30.88**	33.37	32.24	28.49	36.77	38.82	-	-	-
JointGL [108]	28.62	**35.52**	**36.29**	**32.61**	40.17	**41.09**	**13.02**	**19.35**	**21.72**
GENERATION MODELS									
TRANSFORMER,KP20K + MAGKP [134]	16.50	20.23	-	34.74	41.46	-	**36.30**	**36.23**	-
text-davinci-003 - kw	14.05	19.32	20.64	**38.39**	**47.90**	48.45	24.80	30.10	30.45
GPT-3.5-turbo-instruct - kw	11.92	16.74	19.83	23.98	33.16	36.80	16.15	20.92	23.42
GPT-3.5-turbo - kw	18.02	24.06	**25.38**	35.27	46.83	**48.63**	22.88	30.38	31.19
GPT-3.5-turbo - kp	**19.63**	23.80	24.26	34.60	43.12	44.28	20.77	27.46	28.49
GPT-3.5-turbo - 10kw	18.87	**25.10**	25.09	32.22	44.71	44.73	26.10	33.99	**33.99**
GPT-3.5-turbo - 10kp	6.42	7.89	7.91	18.32	19.85	19.88	8.48	8.72	8.72

Table 22: Number of keyphrases predicted by different models. Numbers in parentheses suggest the number of total/present/absent ground-truth keyphrases in each dataset.

MODEL	DUC2001 (8.1/7.9/0.2)			INSPEC (9.8/7.8/2.0)			SEMEVAL2010 (15.1/6.7/8.3)		
	#pred	#pre_pred	#ab_pred	#pred	#pre_pred	#ab_pred	#pred	#pre_pred	#ab_pred
TRANSFORMER,KP20K + MAGKP [134]	134.5	34.5	100.0	143.1	21.9	121.2	169.7	21.6	148.1
text-davinci-003 - kw	13.6	11.7	1.9	8.8	8.2	0.6	9.1	8.5	0.6
GPT-3.5-turbo-instruct - kw	21.6	19.5	2.1	16.8	15.4	1.5	20.6	18.6	2.0
GPT-3.5-turbo - kw	12.6	11.5	1.2	10.6	9.8	0.7	11.6	10.6	0.97
GPT-3.5-turbo - kp	12.2	9.5	2.7	9.4	8.4	1.0	11.2	9.7	1.5
GPT-3.5-turbo - 10kw	10.0	9.5	0.5	10.0	9.4	0.6	10.0	9.3	0.7
GPT-3.5-turbo - 10kp	10.2	2.5	7.7	9.5	4.0	5.5	10.0	3.9	6.1

with author-assigned keywords, suggests that this task aligns well with the models' training data. However, potential data contamination could also influence this performance, as INSPEC papers may have been included in their pre-training. The datasets for DUC-2001 and SEMEVAL, annotated under distinct guidelines [97, 206], did not show strong results, indicating a need for further fine-tuning or instruction-specific training for these tasks.

Comparatively, GPT-3.5-turbo and text-davinci-003 exhibited similar performances on scientific datasets, with GPT-3.5-turbo showing excellence in DUC-2001. As the successor of text-davinci-003, GPT-3.5-turbo-instruct, however, presented weaker performance across all datasets, underscoring the impact of pretraining on LLMs' effectiveness in keyphrase tasks. Despite this, the chat-aligned GPT-3.5-turbo demonstrated improved keyphrase generation capabilities.

The experiment revealed the significance of prompt design, as shown by the varying performances of GPT-3.5-turbo with different prompts. By explicitly defining keyphrase number or length in prompts like 10kw and 10kp, I could steer the model's output more precisely. For instance, the 10kw prompt led to improved scores on SEMEVAL, aligning the model output closer to the dataset's average keyphrase count. This suggests that prompt engineering could act as hyperparameter tuning, optimizing model performance for specific tasks. However, 10kp is a number-controlled variant of kp, but the model's behavior has drastically changed and results in much worse performance. I investigated the outputs and noticed that the model generates phrases that are absent from the text, instead of following the prompt of “extract”. While prompt engineering allows for substantial model control, it doesn’t always guarantee adherence to instructions. This inconsistency highlights the potential for optimization in future model iterations. More details can be seen in the number of predicted keyphrases as shown in Table 22. By explicitly specifying the number to predict in the prompt, both `10kw` and `10kp` output keyphrases of the desired number, whereas the number by the other models is subject to the prior knowledge learned during training.

7.3 LLMS ON PHRASE SEMANTIC UNDERSTANDING

7.3.1 Motivation

Phrase semantics present unique challenges due to natural language's compositionality and ambiguity. The same set of words can yield phrases with vastly different meanings, exemplified by "traffic light" versus "light traffic." Understanding phrase semantics [100, 197, 3, 153] involves tasks that assess a model's ability to discern the semantic relationships between phrases, such as identifying semantic similarities or differences between phrase pairs.

Traditional approaches to these tasks typically employ either embedding-based or fine-tuning strategies. Embedding-based methods [210, 94, 164, 61, 229] generate vector representations for phrases, using vector similarity as a proxy for semantic relatedness. However, this approach often oversimplifies the complex relationships between phrases. On the other hand, fine-tuning methods [210, 156] adapt a pretrained model (like BERT) to a specific task, such as semantic classification or relatedness regression, but this requires substantial task-specific data, which may not be readily available.

LLMs stand out for their ability to generalize to new tasks using just a few instructions, a significant advantage in data-sparse situations. Additionally, in-context few-shot learning and chain-of-thought prompting have demonstrated effectiveness in transferring knowledge to unfamiliar tasks and enhancing reasoning capabilities. The potential of LLMs in phrase semantic understanding remains underexplored and warrants investigation.

In this section, I will investigate how well LLMs understand phrase semantics by instruction-following.

7.3.2 Methods

We compare three types of models to evaluate their performance in understanding phrase semantics:

- **Embedding similarity based methods**: These methods encode phrases into dense vectors and measure semantic relatedness through vector similarity. For classification, the phrase with the highest vector similarity is selected, while for regression tasks, the similarity score itself is used as the prediction.

- **Fine-tuning based methods**: This approach involves adding a prediction head to a pretrained model, fine-tuning it to capture the nuances of specific semantic tasks. In this study, results are reported for a classification dataset, PiC-PS, where a training split is available.
- **Natural language instruction based methods**: Leveraging instruction-tuned models like GPT-3 [24] and FLAN [40], this method provides task details in natural language as part of the input, guiding the model to generate the required output. Techniques such as in-context few-shot learning [23] and chain-of-thought prompting [215, 214] are employed to enhance the model's task-specific predictions and reasoning capabilities.

7.3.3 Experiment Setting

To evaluate phrase semantics understanding, three datasets were utilized:

- **Turney** [197]: Comprising 2,180 examples derived from WordNet, this dataset pairs one bigram phrase with five unigrams, where one unigram is a synonym of the bigram, and the others are related distractors. As a multiple-choice classification task, it challenges the model to identify the synonym from the unigrams, with performance measured by accuracy.
- **BiRD** [3]: This is a bigram relatedness dataset using the Best–Worst Scaling (BWS) annotation. In contrast to common annotation techniques that use a discrete 0 to 5 scale, BWS is expected to provide more reliable and discriminating annotations. Specifically, it employs comparative annotations, where annotators are given multiple items at a time and asked to select which item is the best and worst. BiRD consists of 3,345 pairs of bigram phrases, each including a pair of phrases and a human rating of similarity between 0 and 1. The Pearson correlation coefficient between the model predictions and human judgments is reported.
- **PiC-PS** (Phrase-in-Context - Phrase Similarity) [156]: PiC-PS is a binary classification task and it contains 2,000 examples for testing. The model is tasked to predict whether two noun phrases are semantically similar or not in the same context. The challenge of this task is that, without taking the contextual information into account, very likely the two phrases are interpreted as synonyms. The accuracy score is reported.

For **Turney** and **BiRD**, I follow the setting by [210] to reproduce baselines. For **PiC-PS**, I adopt the setting by [156] and reuse their reported scores. I run `GPT-3.5-turbo` (API name *gpt-3.5-turbo-1106*, version *20231106*) and `GPT-4-turbo` (API name *gpt-4-1106-preview*, version *20231106*) on these datasets. Sample data and prompts are detailed in Table 23. In instances where the OpenAI API response does not yield a valid output, a default similarity score of -1 is assigned. The potential of few-shot learning to enhance LLM performance was also considered. Given the absence of a development set for **Turney** and **BiRD**, 10 examples from each test split were allocated as a makeshift development set, leaving 2,170 and 3,335 examples for final testing, respectively.

Table 23: Prompts used to predict keyphrases from the text document.

Task	**Prompt**	**Ground-truth**	**Output**
Turney	Given 5 word candidates (delimited by |), you are tasked to answer which word is most semantically similar to a given phrase. Please respond with the word only, without any explanation. Among 5 words "proverb | sabre | stockpiling | moonlighter | jigsaw", which word is most similar to the phrase "saber saw"?	jigsaw	jigsaw
BiRD	Given two phrases, you are tasked to predict the semantic relatedness of them. Now, given two "access road" and "road access", what is their semantic relatedness? Please only output a number within the range [0, 100].	0.684	The semantic relatedness between "access road" and "road access" is 100. These phrases are semantically identical, as they convey the same meaning with the words in a different order.
PiC	Given a pair of phrases sharing the same context, you are tasked to answer whether the two phrases have the same semantic meaning. Now here is the test, in the two sentences below: - This was also the first address of his sister Hannah's shop where she sold James Gillray's prints before moving to Bond Street. - This was also the inaugural speech of his sister Hannah's shop where she sold James Gillray's prints before moving to Bond Street. The phrase pair "first address" and "inaugural speech" has the same semantic meaning is (only output Yes or No):	yes	yes
PiC+CoT	Given a pair of phrases sharing the same context, you are tasked to answer whether the two phrases have the same semantic meaning. Now here is the test, in the two sentences below: - On the local circuit, Faiz emerged champion in 3 events and obtained medals in both the Foil and Sabre U-12 events. - On the community competitions, Faiz emerged champion in 3 events and obtained medals in both the Foil and Sabre U-12 events. Please answer whether the phrase pair "local circuit" and "community competitions" has the same semantic meaning. Let's think step by step, briefly explain the similarity/difference between the two phrases, and end the respond with only Yes or No).	yes	The phrases "local circuit" and "community competitions" have a similar semantic meaning. Both phrases refer to events or competitions that take place within a specific local or community setting. Therefore, the answer is Yes.

7.3.4 Results

Table 24: The scores on two phrase semantic similarity datasets. The numbers in parentheses indicate how many times the model failed/refused to return a valid answer.

Model	**Turney** (2,170) **Accuray**	**BiRD** (3,335) **PCC**
Baselines (Embedding+Cosine)		
BERT	42.5	0.445
BERT-Large	42.5	0.476
SpanBERT	20.8	0.257
SpanBERT-Large	41.0	0.330
Phrase-BERT	57.1	0.689
LLMs (Instruction)		
`GPT-3.5-Turbo`, 0-shot	86.2	**0.762** (4)
`GPT-3.5-Turbo`, 2-shot	86.4	0.698 (36)
`GPT-3.5-Turbo`, 4-shot	**87.4**	0.685 (54)
`GPT-4-Turbo`, 0-shot	85.9	0.727 (3)
`GPT-4-Turbo`, 2-shot	87.0	0.587 (1309)

Table 25: The scores on PiC-PS.

Model	**PiC-PS Accuracy**	
Baselines	**M1: Embedding+Cosine**	**M2: Fine-tuned Classifiers**
BERT	64.10	68.85
SpanBERT	64.00	66.85
SpanBERT-Large	66.30	69.25
SentenceBERT	60.30	62.55
PhraseBERT	63.40	66.10
SimCSE	62.50	66.65
LLMs	**M3: w/o CoT**	**M4: w/ CoT**
`GPT-3.5-Turbo`, 0-shot	60.80	65.85
`GPT-3.5-Turbo`, 1-shot	67.95	67.45
`GPT-3.5-Turbo`, 2-shot	70.00	61.70
`GPT-3.5-Turbo`, 4-shot	69.80	64.50
`GPT-4-Turbo`, 0-shot	**73.85**	63.05
`GPT-4-Turbo`, 2-shot	73.15	69.60

The results of phrase semantic similarity are presented in Table 24 and 25. The evaluation of phrase semantic similarity across the datasets shows that instruction-based methods employing

LLMs set new benchmarks (Turney from 57.1 to 87.4, BiRD from 0.689 to 0.762, PiC-PS from 69.25 to 73.85). Interestingly, the highest scores varied across different models and configurations. For instance, despite GPT-4-Turbo being an advancement over GPT-3.5-Turbo, it only scored higher on PiC-PS without Chain-of-Thought (CoT) prompting.

In Turney's benchmark, the previous best accuracy score (57.1) was achieved by an embedding-based method utilizing Phrase-BERT and cosine similarity. However, cosine similarity, as a proxy measure, may not sufficiently capture the intricate semantic relationship between two phrases. Large Language Models (LLMs) significantly outperform this previous score on Turney's benchmark by a substantial margin (+30.3 points), primarily due to their enhanced language understanding capabilities. Additionally, their ability to follow instructions enables adaptation to new tasks without requiring any training data. Furthermore, it's worth noting that further improvements can be attained by providing a few-shot examples within the context, allowing the model to glean more task-specific information for refining its predictions.

BiRD directly evaluates models' similarity predictions by Pearson correlation coefficient, which may favor embedding-based methods as they were trained by optimizing the similarities between phrases [210], whereas LLMs output similarities in natural language, similar to the way that humans annotate the data [3]. The result shows that LLMs are still able to outrun baselines by a clear margin. However, the performance deteriorates when few-shot examples are provided. One possible reason is the values provided in the examples potentially bias the models' prediction. Another non-trivial impact of the few-shot prompt is that, it significantly increases the chance that OpenAI APIs refuse to answer as instructed. The frequency of failure responses is shown in parentheses in Table 24, and we can see that it has a more severe influence on `GPT-4-Turbo`, leading to 1,309 refusal responses, and one example is "As an AI language model, I don't have the capability to calculate semantic relatedness with a precise numerical score. Semantic relatedness is typically determined using complex algorithms or models that analyze the meanings of words and phrases in context. However, I can provide a qualitative assessment." Further prompt tuning or repeated sampling may alleviate this problem.

PiC-PS is a context-dependent test, asking the model if two phrases carry distinct meanings in the same context. The task is arguably more challenging as the difference between them can be subtle and models need to comprehend the context for support. The previous best score (69.25) is achieved by SpanBERT-Large with a fine-tuned classifier. Surprisingly, the best LLM only attains a minor improvement (about 4 points). The few-shot demonstration benefits `GPT-3.5-Turbo` the most when two examples are used, but it worsens the performance of `GPT-4-Turbo`. Additionally, the result shows that the chain-of-thought prompt leads to degradation in general. The models appear to give a very reasonable interpretation and few invalid responses are detected. A response from `GPT-4-Turbo` is shown as follow:

Input:
Given a pair of phrases sharing the same context, you are tasked to answer whether the two phrases have the same semantic meaning.
Here are 2 examples:

Example #1, in the two sentences below:
- Carty attended a Saturday morning acting class at Phildene Stage School from the age of four, which had an attached agency.
- Carty attended a Saturday morning acting class at Phildene Stage School from the age of four, which had an accompanying business that represents actors.

Whether the phrase pair "attached agency" and "accompanying business that represents actors" has the same semantic meaning? Let's think step by step: Both phrases describe a business connected to the Stage School, with the primary function of representing actors.
So the answer is: Yes

Example #2, in the two sentences below:
- after theo's apparent death, she decides to leave first colony and ends up traveling with the apostles.
- after theo's apparent death, she decides to leave original settlement and ends up traveling with the apostles.

Whether the phrase pair "first colony" and "original settlement" has the same semantic meaning? Let's think step by step: Both phrases refer to an initial establishment or habitation, but "first colony" carries specific connotations of being part of a series of colonies or a larger colonization effort, whereas "original settlement" is a more neutral term for any first habitation.
So the answer is: No

Now here is the test, in the two sentences below:
- The software was ported to a wide set of Unix and Linux variants.
- The software was ported to a spacious collection of Unix and Linux variants.

Please answer whether the phrase pair "wide set" and "spacious collection" has the same semantic meaning.
Let's think step by step, briefly explain the similarity/difference between the two phrases, and end the response with only Yes or No)."

Ground-truth: No.

Response:
The phrase "wide set" refers to a broad range or assortment, emphasizing the diversity or number within a group. The phrase "spacious collection" is not commonly used in this context; "spacious"

typically describes a physical space that is large or roomy. However, if we interpret "spacious collection" as an attempt to convey a large assortment, then it could be seen as a less conventional way of expressing a similar idea to "wide set," with the focus being on the size of the collection rather than its diversity.

So, while the phrases are not identical in their usual contexts and connotations, in the context of describing the extent of software compatibility across different Unix and Linux variants, they could be understood to mean a similar thing: a large and varied group.
The answer is: Yes.

7.4 LLMS ON KEYNESS AND PHRASENESS UNDERSTANDING

7.4.1 Motivation

Previous sections have highlighted the strong performance of Large Language Models (LLMs) in keyphrase generation and phrasal semantic understanding, as demonstrated through benchmarking on various datasets. However, these models have not been evaluated specifically on their abilities to understand keyphrase-related factors, namely keyness and phraseness. To address this gap, I propose constructing two datasets aimed at testing LLMs' capability in discerning phrases of different degrees of keyness or phraseness.

7.4.2 Dataset Construction

Inspired by existing datasets on phrasal semantics, I intend to develop a multi-choice dataset to assess models' understanding of keyness and phraseness. This dataset consists of 500 papers selected from the `KP20K` test split, each with precisely four author keywords (5,371 papers meet this criterion out of 19,987). Adhering to the definitions provided in [194], I outline the process for constructing distractors below:

- **Keyness/Informativeness**:
  - Definition: Measures how effectively a phrase captures or illustrates the key ideas in a set of documents.
  - Data construction: The original authors' annotated keywords serve as positive labels, while four random keyphrases are sampled from other documents, deemed irrelevant to the source document. This evaluation can be conducted in four settings based on whether the target is a positive keyphrase or a negative one (identifying non-keyphrases vs. identifying keyphrases) and whether the source context is provided to the model (with context vs. without context). The absence of context significantly raises the difficulty level, as the model must discern, from five phrase candidates, which four are likely from the same document and semantically related.
- **Phraseness**:
  - Definition: Measures the degree to which a word sequence is considered a phrase.

- Data construction: The original authors' annotated keywords are positive labels, while four n-grams from the same document, excluding all noun phrases, are randomly selected as negatives. To reduce false negatives, the length of sampled n-grams ranges from 2 to 4 words. Phraseness is considered a context-free characteristic, leading to two evaluation settings: (1) Identifying non-keyphrases by spotting one non-phrase n-gram (negative) among five candidates, with the remaining four being well-formed phrases (positives). (2) Identifying keyphrases by identifying one well-formed phrase (positive) among five candidates, with the remaining four being non-phrases (negatives).

7.4.3 Results

Since there are no suitable baseline methods available for keyness and phraseness prediction, I exclusively evaluate the performance of two OpenAI Large Language Models (LLMs) using the prompts outlined in Table 26.

Table 26: Accuracy scores of keyness/phraseness prediction by various LLMs.

	Identify Non-keyphrase			Identify Keyphrase	
	Keyness w/ context	Keyness w/o context	Phraseness	Keyness w/ context	Phraseness
GPT-3.5-Turbo	**96.6**	66.2	88.8	96.6	79.6
GPT-4-Turbo	95.8	**70.0**	**97.8**	**97.4**	**89.2**

The performance scores are summarized in Table 26. Both LLMs exhibit excellent keyness prediction when provided with context. However, when the context is absent, the scores notably decrease, indicating the difficulty of linking abstract keyphrases without the source text as a reference point. Regarding phraseness, typically viewed as a simpler task of selecting well-formed phrases from random n-grams and vice versa, **GPT-3.5-Turbo** surprisingly makes incorrect predictions in over 10% of test cases, highlighting the challenge even in this seemingly straightforward task. On the other hand, **GPT-4-Turbo** significantly improves phraseness scores. Comparing the settings of identifying non-keyphrases and identifying keyphrases, the models show similar performance in keyness prediction. It seems easier to predict a random n-gram from several well-formed phrases

(97.8 vs. 89.2 accuracy), although more false negatives may occur in the identifying keyphrase setting.

7.4.4 Summary

In this chapter, I conduct a comprehensive series of experiments to assess the performance of Large Language Models (LLMs) in keyphrase generation and understanding.

Initially, I conduct an empirical study that evaluates multiple LLMs on three widely recognized keyphrase benchmarks, comparing their performance against previous state-of-the-art keyphrase models. The results on in-domain datasets highlight the impressive performance of LLMs, attributed to their larger model size and exposure to extensive pretraining data. Furthermore, this study underscores the critical importance of using appropriate prompts to achieve optimal results.

Following that, I delve into analyzing the performance of LLMs in phrasal semantic understanding, employing instruction-following methods and comparing them against various baseline models. While LLMs exhibit superior performance across three benchmarks, the utilization of few-shot demonstration and chain-of-thought prompts yields mixed results. This prompts further exploration and analysis to ascertain why chain-of-thought prompts may struggle in semantic reasoning tasks.

Additionally, I introduce a novel dataset designed to evaluate LLMs' proficiency in understanding keyness and phraseness, which are fundamental characteristics of keyphrases. Through this preliminary evaluation, it becomes evident that current LLMs possess the capability to differentiate keyphrases from other phrases. However, certain challenges persist in specific evaluation settings, indicating areas for improvement.

Overall, LLMs demonstrate remarkable capabilities in keyphrase-related tasks. Future studies should focus on exploring additional applications of LLMs in keyphrase tasks, including automatic evaluation methods and other potential use cases.

Table 27: Data examples of keyness/phraseness prediction under various settings and the corresponding prompts for LLMs.

Setting		Prompt	Ground-truth	Model Output
Identify Non-keyphrase	Keyness w/ context	Given a list of 5 phrases that will be used as keyphrases to describe the core topics of a document, there is one irrelevant phrase among them. You are tasked to identify which one is the irrelevant phrase (not a keyphrase). Please respond with the phrase directly, without any explanation. The document is as follows: Search strategies on a new health information retrieval system . Purpose - The goals of this study are: to evaluate the merits of a newly developed health information retrieval system; to investigate users' search strategies when using the new search system; and to study the relationships between users' search strategies and their prior topic knowledge. Design/methodology/approach - The paper developed a new health information retrieval system called MeshMed. A term browser and a tree browser are included in the new system in addition to the traditional search box. The term browser allows a user to search Medical Subject Heading (MeSH) terms using natural language... Here are 5 keyphrase candidates: ["information systems", "hospitals", "knowledge management", "cost-effective design", "information retrieval"]. Now tell me which phrase is unlikely to be a keyphrase (irrelevant)? Answer:	cost-effective design	hospitals
	Keyness w/o context	Given a list of 5 phrases that will be used as keyphrases to describe the core topics of a document, there is one irrelevant phrase among them. You are tasked to identify which one is the irrelevant phrase (not a keyphrase). Please respond with the phrase directly, without any explanation. Here are 5 keyphrase candidates: ["content dissemination", "wireless lan", "friendship relations", "user characteristics", "social media website"]. Now tell me which phrase is unlikely to be a keyphrase (irrelevant)? Answer:	wireless lan	friend-ship relations
	Phrase-ness	Given a list of 5 keyphrase candidates, most of them are well-formed noun phrases (keyphrases for a document) and one is likely to be a random phrase. You are tasked to identify which one is not a well-formed phrase (not a noun phrase nor keyphrase). Please respond with the phrase directly, without any explanation. Here are 5 keyphrase candidates: ["image-based rendering", "lumiproxy", "surface fitting", "as Trefftz methods", "sampling"]. Now tell me which one is not a well-formed phrase? Answer:	as trefftz methods	lu-miproxy
Identify Keyphrase	Keyness w/ context	Given a list of 6 phrases that will be used as keyphrases to describe the core topics of a document, there is only one phrase considered relevant. You are tasked to identify which one is the relevant phrase (a keyphrase). Please respond with the phrase directly, without any explanation. The document is as follows: distributed multivariate regression based on influential observations . Large-scale data sets are sometimes logically and physically distributed in separate databases. The issues of mining these data sets are not just their sizes, but also the distributed nature. The complication is that communicating all the data to a central database would be too slow. To reduce communication costs, one could compress the data during transmission. Another method is random sampling. We propose an approach for distributed multivariate regression based on sampling and discuss its relationship with the compression method. The central idea is motivated by the observation that, although communication is limited, each individual site can still scan and process all the data it holds. Thus it is possible for the site to communicate only influential samples without seeing data in other sites. We exploit this observation and derive a method that provides tradeoff between communication cost and accuracy. Experimental results show that it is better than the compression method and random sampling.. Here are 6 keyphrase candidates: ["reliability function", "learning curve", "unfolding", "optimal scheduling", "electre", "range-reduction"], Now tell me which phrase is likely to be a keyphrase (relevant)? Answer:	learning curve	multi-variate regres-sion
	Phrase-ness	Given a list of 6 keyphrase candidates, most of them are random n-grams and only one is a well-formed noun phrase (likely to be a keyphrase for a document). You are tasked to identify which one is the well-formed noun phrase. Please respond with the phrase directly, without any explanation. Here are 6 keyphrase candidates: ["30] using fundamental", "solutions (", "and the", "of the MFS over", "s2-continuous poset", "(TM) ["], Now tell me which phrase is likely to be the well-formed noun phrase (a keyphrase)? Answer:	s2-continuous poset	and the

8.0 CONCLUSION AND FUTURE WORKS

The preceding chapters have addressed each of the research questions outlined in the Introduction (Chapter 1). This concluding chapter aims to consolidate the key findings and insights derived from the research conducted in this dissertation. Additionally, it will explore potential avenues for future research.

8.1 CONCLUSION

As an extensively studied natural language processing task, traditional methods for automatic keyphrase assignment, namely keyphrase extraction and automatic tagging, have demonstrated limitations in replicating human-like keyphrase prediction capabilities. Motivated by these shortcomings, I reexamined the constraints of previous studies and introduced a more comprehensive formulation of this task, which I termed **Keyphrasification**. This task is conceptually defined as the process of summarizing source content and conveying condensed information through phrases. To validate this conceptualization, I conducted an interview study involving 15 participants to understand human perceptions and behaviors regarding keyphrases. The results confirmed that most participants adhere to the proposed formulation of keyphrasification when assigning keyphrases. Additionally, a set of properties related to keyphrasification was discussed and assessed by the participants, shedding light on their significance. These findings are anticipated to deepen our understanding of keyphrase-related tasks and facilitate the design of future models in this domain.

Building on the framework discussed earlier, I propose an approach to automatic keyphrasification, namely **Keyphrase Generation**, employing neural language generation techniques. This innovative modeling approach addresses the limitations of prior methods by learning to mimic human preferences for assigning keyphrases in a data-driven manner. Several key techniques are explored and thoroughly evaluated to tackle the unique challenges of this task, including different model architectures, training strategies, beam search, and the copy mechanism. The keyphrase generation method presented here outperforms traditional keyphrase extraction benchmarks and uniquely predicts keyphrases that do not appear in the source text. This method combines the strengths of keyphrase extraction and automatic tagging, showcasing its advanced capability. Moreover, I endeavored to incorporate the specific keyphrase properties defined in the framework into modeling, resulting in the **KPGen-One2Seq-Div** variant. The results demonstrate that deliberate measures aimed at introducing semantic coverage and uniqueness significantly enhance the diversity in the generated phrases.

Nevertheless, previous methods heavily rely on annotated data and often exhibit weak generalization across domains. To address this issue, I proposed a three-stage pipeline aimed at enhancing models' abilities in both keyness and phraseness. This approach involves strategically distilling domain-general phrase knowledge and domain-specific knowledge through a **Phrase-oriented Generative Pre-training**. Experimental results demonstrate that the proposed method is effective in generating high-quality keyphrases in new domains, even with limited in-domain annotated data.

Lastly, In a series of experiments, I explore the performance of **Large Language Models (LLMs)** in keyphrase generation and understanding. Initially, I assess multiple LLMs on three keyphrase benchmarks, finding that on in-domain datasets, LLMs excel due to their larger model size and extensive pretraining data, underscoring the importance of appropriate prompts. Next, I investigate LLMs' performance in phrasal semantic understanding via instruction-following, noting their superiority on three benchmarks but mixed outcomes with few-shot demonstration and chain-of-thought prompts, suggesting the need for further prompt exploration and analysis. Finally, I introduce a new dataset to evaluate LLMs' comprehension of keyness and phraseness, revealing their capability to differentiate keyphrases from other phrases, albeit with some remaining challenges under certain settings.

Throughout this study, the investigation into automatic keyphrasification evolves across **three generations of neural language modeling techniques**. It begins with the initial models, which are trained from scratch on task-specific data, progresses through models pre-trained on a vast corpus of domain-general text and fine-tuned for specific tasks or domains with minimal additional training, and culminates with the latest large language models (LLMs) capable of adapting to a wide array of NLP tasks through instruction following alone. As the scale of both model size and data volume increases, we witness an enhancement in the capabilities of language models, alongside a trend towards performance saturation on established keyphrase benchmarks. While current models seem to effectively address the keyphrase assignment challenge, I argue that LLMs pave the way for novel applications related to keyphrases. This advancement necessitates the development of more refined task designs to fully assess the potential of LLMs in understanding and assigning keyphrases.

8.2 DISCUSSION

8.2.1 Misunderstandings in Keyphrasification

The findings discussed in Chapter 3 indicate that individuals typically select keyphrases based on their personal criteria, which are shaped by practice and experience, and often lack formal instruction. Furthermore, the practical utility of keyphrases, as discussed in the literature [76], may not be widely recognized by actual users. These factors contribute to a general ambiguity and misunderstanding regarding the purpose and importance of keyphrasification, resulting in a tendency for keyphrases to be assigned haphazardly. Consequently, it is imperative to define the task of keyphrasification clearly, elucidate the value of keyphrases, and provide specific guidelines for their assignment.

I think the significance of keyphrases varies across different applications and scenarios, which can be classified into three main categories:

- **Indexing Terms**: In an ideal setup, when a user employs a specific term in a search query, documents tagged with that term should appear prominently in the search results. However, the effectiveness of this system diminishes if the same term is assigned across numerous documents. Therefore, content-specific phrases, which are often "extractable" from the content, are marked as keyphrases. With advancements in language understanding technologies, neural search models are increasingly capable of identifying significant terms within content, including non-textual data [213], seemingly reducing the reliance on traditional keyword-based indexing. Exploring more sophisticated methods to integrate keyphrases and additional metadata seamlessly into neural search represents a valuable direction for future research.
- **Social Tagging or Folksonomy**: This involves users within a community assigning tags or labels to content to facilitate organization, exemplified by hashtags on Twitter, topic labels

on StackOverflow, and the selection of academic paper reviewers by keywords. Although there is no strict rule for tag assignment, each tag aims to represent a resource group, creating a community-based taxonomy. Thus these tags should be recognizable to a segment of the community, representing a category, topic, or interest. While automatic models can predict tags from historical data, a notable challenge is the dynamic nature of tagging trends, which these models might not always capture. Investigating the evolution and the taxonomy of tags presents an interesting research avenue.

- **Keyphrases for readers**: Analogous to structured abstracts [147, 158, 132], structured keyphrases [90] can offer a clear outline, aiding readers, particularly professionals like academic researchers and patent attorneys, in comprehending the main themes of extensive texts. A systematic format not only assists readers in swiftly locating relevant content but also guides authors in crafting meaningful keyphrases. Moreover, search engines can leverage this structured data for enhanced functionality, e.g. filtering results by keyword types. Unlike structured abstracts, which often follow the IMRAD format (Introduction, Methods, Results, and Discussion), a distinct annotation framework for keyphrases should be developed to elucidate the connection between the document and its keyphrases. The interview study in Chapter 3 suggests possible keyphrase functions/roles such as method, problem, solution, goal, focus, and domain, which could structure this annotation. However, the norm of keyword types should be adaptable to the conventions and practices of different research fields.

The exploration of keyphrasification practices reveals the need for a structured and deliberate approach to keyphrase assignment. As we have seen, the selection and application of keyphrases are influenced by a variety of factors, from individual discretion to broader community norms. The diversity in the use and recognition of keyphrases across different contexts highlights the complexity of this problem. Moving forward, it is crucial to cultivate a more systematic and informed methodology in keyphrasification to ensure that keyphrases fulfill their potential as pivotal navigational and informational tools in the digital information landscape. Furthermore, a more precise understanding of keyphrase assignment practices, coupled with well-defined keyphrase frameworks, will lay the groundwork for advancing research in automated keyphrasification methods.

8.2.2 Emerging Opportunities with Large Language Models (LLMs)

The central focus of this dissertation has been the investigation of automated keyphrasification solutions, traversing through three generations of neural language modeling. The journey begins with neural network models trained from scratch on task-specific and domain-specific data, advances through models pre-trained on extensive domain-general text and fine-tuned for particular tasks with minimal training, and culminates with Large Language Models (LLMs) that adapt to new tasks and domains through instruction following only. The evolution of language modeling has consistently elevated the performance of automatic keyphrasification across various domains and scenarios, as demonstrated in Chapter 7, and showcased their effectiveness on unseen tasks without extensive model tuning. This progress supersedes the traditional modeling approaches discussed in earlier chapters, which typically rely on considerable amounts of task-specific training data (e.g., KP20K and StackEx) and intricate modeling techniques (e.g., copying attention, semantic coverage).

However, I contend that LLMs unlock additional avenues for keyphrase research and application. As identified earlier, numerous real-world applications for keyphrases are still underexplored. Leveraging LLMs' advanced capabilities opens up new possibilities for investigating the untapped potential of keyphrases. For example, LLMs' ability to identify crucial phrases, evidenced in Chapter 7, suggests their utility in classifying keyphrases according to their functions or roles, thus facilitating further applications. Another intriguing prospect is automating keyphrase evaluation using LLMs. Recent studies have shown GPT-4's adeptness in evaluating tasks like summarization and long-form question answering through specialized prompts, such as chain-of-thought and form-filling. Capitalizing on LLMs' capacity to discern semantically similar phrases could lead to the creation of an automated evaluation framework, with its effectiveness measured against human judgments. Insights from Chapter 3 could guide the formulation of such an evaluative mechanism.

8.3 FUTURE WORKS

In this section, I will discuss several research directions that are worth exploring in the future.

8.3.1 Enhancing User Interaction through Keyphrasification

Investigating how keyphrases influence user-system interactions is a fascinating yet complex area. Keyphrases, as concise summaries of detailed content, can significantly reduce the cognitive burden on users who engage with extensive information resources, like searching or reviewing lengthy documents. The development of interactive systems that leverage generated keyphrases to facilitate user interactions with data, and evaluating their impact, can offer valuable perspectives on the practical utility of keyphrases in real-world scenarios. This exploration can enrich our understanding of keyphrases' role in improving information accessibility and user experience.

8.3.2 Keyphrase Generation for Non-Textual Content

The extension of keyphrasification techniques to encompass non-textual media, including images, videos, and audio files, represents a pioneering frontier in information accessing. Investigating the generation and application of keyphrases for such multimodal content offers a pathway to enhance indexing and retrieval processes, enabling more nuanced and efficient searches. This approach could transform how users interact with and extract value from diverse media types, providing a unified framework for summarizing and navigating complex information resources. By developing algorithms capable of discerning and encapsulating the essence of visual and auditory information into concise keyphrases, we can facilitate a more intuitive and accessible retrieval experience, bridging the gap between various content forms and their textual representations.

8.3.3 Integration of Keyphrasification with Other Tasks

Examining how keyphrasification can be combined with other NLP tasks like document summarization, topic modeling, or information retrieval to enhance overall text understanding and information extraction capabilities.

Currently, keyphrases are often utilized in a simplistic, non-hierarchical manner, overlooking the potential depth of their interrelations. A promising avenue for enhancing various NLP tasks is to employ a keyphrase-driven taxonomy, structuring information in a hierarchy that reflects the underlying knowledge relationships encapsulated by keyphrases. This approach not only organizes data more effectively but also facilitates more logical and user-friendly access to large volumes of information.

Further research should also focus on developing sophisticated techniques for embedding keyphrases into information retrieval systems, thereby augmenting search relevance and accuracy, and enriching the user experience with features like personalized search outcomes tailored to individual user-generated keyphrases.

8.3.4 Other Directions

I would like to highlight several promising research areas that merit further investigation. Some of these topics have been touched upon in my previous work but are not extensively covered in this dissertation:

- **Cross-domain and Cross-lingual Keyphrasification**: This area focuses on assessing the adaptability of keyphrasification models across various domains and languages, aiming to understand their flexibility in different textual environments. In Chapter 6, we introduced a method for distilling 'keyness' and 'phraseness' through pseudo data labeling. The study in [62] proposed a retrieval-augmented approach to mitigate data scarcity in low-resource languages. Further research is needed to evaluate the disparities between domains and languages comprehensively. Additionally, the extraction of bilingual phrase pairs represents a fertile ground for future exploration.
- **Keyphrasification Models with Limited or No Supervision**: The extensive need for annotated training data is a significant bottleneck in the advancement of keyphrasification models. Studies like [174] and [133] have introduced data-efficient methodologies to lessen this dependency. Developing robust methods that can intuitively guide models to grasp the inductive biases of specific tasks continues to be a formidable challenge, despite the rise of LLMs. Data synthesis and augmentation leveraging language models have yielded promising outcomes in some domains [208, 130], suggesting that this approach warrants deeper exploration.

8.4 RELATED PUBLICATIONS

Throughout my doctoral studies, I authored 10 publications connected to the theme of keyphrasification, each contributing directly or indirectly to the substance of this dissertation. A comprehensive list of these publications is detailed in Table 28. These papers are arranged not by chronological order, but by their respective research objectives and contributions. This organization aims to provide readers with a clearer understanding of the research trajectory undertaken in this endeavor.

Table 28: A list of publications related to this dissertation.

Title •Author List •Conference / Journal	Chapter	Core Contribution
1. Deep Keyphrase Generation [135] Meng, R., Zhao, S., Han, S., He, D., Brusilovsky, P., & Chi, Y. ACL, 2017.	Chapter 4	First study of keyphrase generation.
2. An Empirical Study on Neural Keyphrase Generation [134] Meng, R., Yuan, X., Wang, T., Zhao, S., Trischler, A., & He, D. NAACL, 2021	Chapter 4	Comprehensive empirical study.
3. One Size Does Not Fit All: Generating and Evaluating Variable Number of Keyphrases [231] Yuan, X., Wang, T., Meng, R., Thaker, K., Brusilovsky, P., He, D., & Trischler, A. ACL, 2020.	Chapter 5	Proposed semantic coverage and orthogonal regularization.
4. General-to-Specific Transfer Labeling for Domain Adaptable Keyphrase Generation [133] Meng, R., Wang, T., Yuan, X., Zhou, Y., & He, D. Findings of ACL, 2023.	Chapter 6	Low-resource KPG.
5. Unsupervised Deep Keyphrase Generation [174] Shen, X., Wang, Y., Meng, R., & Shang, J. AAAI, 2022.	Not included	Unsupervised KPG.
6. Generating Keyphrases for Readers: A Controllable Keyphrase Generation Framework [90] Jiang, Y., Meng, R., Huang, Y., Lu, W., & Liu, J. Journal of the Association for Information Science and Technology, 2023.	Not included	Structured KGP.
7. Bringing Structure into Summaries: a Faceted Summarization Dataset for Long Scientific Documents [132] Meng, R., Thaker, K., Zhang, L., Dong, Y., Yuan, X., Wang, T., & He, D. ACL 2021.	Not included	Structured summarization.
8. Retrieval-Augmented Multilingual Keyphrase Generation with Retriever-Generator Iterative Training [62] Gao, Y., Yin, Q., Li, Z., Meng, R., Zhao, T., Yin, B., King, I, & Lyu, M. R. Findings of NAACL 2022.	Not included	Cross-lingual KPG.
9. AugTriever: Unsupervised Dense Retrieval by Scalable Data Augmentation [130] Meng, R., Liu, Y., Yavuz, S., Agarwal, D., Tu, L., Yu, N., Zhang, J, Bhat, M., & Zhou, Y. arXiv:2212.08841.	Not included	KPG as a data augmentation for dense retrieval.
10. From Fundamentals to Recent Advances: A Tutorial on Keyphrasification [131] Meng, R., Mahata, D., & Boudin, F. , Y. ECIR Tutorial 2022.	Chapter 3	Tutorial on Keyphrasification.

Appendix A INTERVIEW PROTOCOL

Purpose of the Study

We are interested in how humans evaluate the keyphrases that are used to describe the key content of the text.

Part 0: Introduction

- Start recording. Avoid using identifying names.
- Warm-up and background information of the participant:
  - Which department are you from? What year are you in your current program?
  - Tell me a bit about your research interests.

Definition: *A keyphrase or a keyword is a short sequence of one or multiple words that describes the key content of a longer text. The common forms and use cases of keyphrases include author-assigned keywords to articles, user-generated tags to blogs, or online content...I try not to describe more because that might hijack your answer…*

Part 1: Keyphrase Use Cases

- From your experience, in what scenarios have you ever used keyphrases (This can be keyphrases created by you or you use keyphrases created by others)?
- How do you use keyphrases in this scenario?
- To what extent keyphrases are important to you in this scenario, why?
- Is there any other case in which no keyphrase is provided, but you would like to have keyphrase, why?
- Talking about keypharses in research articles, when you search for papers, do you read the author-assigned keyphrases?
- When you make sense of a newly found paper, do you read the author-assigned keyphrases?
  - If not, how do you quickly get the key content of a newly found paper?

- If so, do you think keyphrase is useful for understanding the content of the paper? How useful?
- To what extent title, abstract, keywords help you to get the idea of the paper?

- As an author, do you expect readers to read the keyphrases of your papers? What do you expect them to get/obtain from your keyphrases?

For the next part, we want to focus on the author-assigned keyphrases in the research articles. We found three of your most recent published articles (use identifiers, original names of the articles will be removed). Next, we want to talk about the keyphrases in each article one by one together with you. We know there is no right or wrong answer when authors assign keyphrases, so what we are interested in are your considerations and reasons behind each keyphrase.

[For each article of the three ask the following questions:] Ask the participants to generate a set of keyphrases that best represent the article. [remove the original keyphrases if available]

Part 2-1: Keyphrase Evaluation - Single Keyphrase

For article XXX, keyphrase YYY...

- How did you come up with this keyphrase?
- How important is this keyphrase in representing the article (scale of 1-5)? Why?
- What is the role (or function) of this keyphrase in representing this article?
- (Prompts: e.g., "area," "method," "data," "problem," "solution," "goal," "technology," "focus," "domain," etc.)
- How do you expect this keyphrase to be used?

Part 2-2: Keyphrase Evaluation - A Set of Keyphrases

- What's your strategy in composing this set of keyphrases that represent this article?
- If there any relationships between the keyphrases in one set?
- How did you decide the number of keywords to include in this article?
- Does the sequence/order of keyphrases in this article matter to you? If not, why? If so, how do you decide on the ordering?

- Is there any other alternative keyphrases you considered but not included in your final set? What are they, and why?
- Why do you use noun/verbal phrases?

Part 2-3: Keyphrase Evaluation - General Strategy

- Do you have any general strategy in composing a set of keyphrases that represent your article? [Prompts: What is your preference in keyword selection: general terms or field-specific ones?]
- Is there any part of the article that you typically refer to when you assign keyphrases? [What is the role of title, abstract, different sections, in your keyphrase assignment process?]
- Does your strategy vary by where you publish?
- Any barriers that you could think about when you assigning keyphrases? (If so, what would help you to deal with them?) Verb phrase/noun phrase
- Have you ever read any instruction/guidance provided by any publisher on how you should choose keywords for your articles?
- If not constrained by the publisher, how many keyphrases you would select to represent your articles?

Part 3-1: Abstract & Keyphrase Evaluation - General Strategy

- Do you have any general strategy in writing the abstract of your articles?
- Is there any part of the article that you typical refer to when you write abstract?
- Does your strategy vary by where you publish?
- Any barriers that you could think about when you write abstract?
- Have you ever read any instruction/guidance how you should write abstract?
- Is there any alignment or relationship between your keyphrase selection strategy and abstract writing strategy?

Part 3-2: Abstract & Keyphrase evaluation - Priority

Below is a tentative keyphrase evaluation framework. For each criterion listed, please tell us:

- Do the criterion and the description make sense to you?
- From 1 to 5, please rate how important it is in your keyphrase evaluation and selection practice. 1 means not important at all, and 5 means very important.
- Does the criterion apply to abstract summarization?

Table 29: Keyness/phraseness properties that are discussed in this study.

Category	Property	Description
Keyness - keyphrases as a whole	Coverage	Whether the set of keyphrases contains a range of topics that the source text covers
	Low-redundancy	Whether the semantic overlap among keyphrases in a set is low
Keyness - individual keyphrases	Importance	Whether the keyphrase represents the essence of the source text
	Relevance	Whether the keyphrase is accurate and contains the same information in the source text
	Specificity	Whether the keyphrase is specific to the content of the source text, in comparison to general knowledge
	Uniqueness	Whether the keyphrase captures unique points of the source text
Phraseness	Conciseness	Whether the keyphrase is short and concise rather than long or descriptive
	Synonymity	Whether the keyphrases contain multiple synonymous phrase candidates that correspond to one semantic meaning

Part 4: Debriefing

- If not talking about any technical constraints, what functions or features would help you with your keyphrase selection?
- Is there anything else that you want to add?
- Thank you so much for your time! If you have any questions or concerns, please feel free to contact [investigators].
- Stop recording. Who else do you think we should talk to?

Appendix B COMPLETE EMPIRICAL RESULTS

In this section, we report the full set of our experimental results pertaining to the discussion in Chapter 4.

In Table 30, we report all the testing scores on present keyphrase generation tasks. For all experiments, we use $\mathbf{F_1@5}$, $\mathbf{F_1@10}$, $\mathbf{F_1}@\mathcal{O}$ and $\mathbf{F_1}@\mathcal{M}$ to evaluate a model's performance. Additionally, we provide an average score for each of the 4 metrics over all datasets (over each row in Table 30).

In Table 31, we report all the testing scores on absent keyphrase generation tasks. For all experiments, we use **R@10** and **R@50** to evaluate a model's performance. Additionally, we provide an average score for each of the 2 metrics over all datasets (over each row in Table 31).

In table 32, we report detailed present testing scores when model trained with `One2Seq` paradigm, using different ordering strategies. For all experiments, we use $\mathbf{F_1@5}$, $\mathbf{F_1@10}$, $\mathbf{F_1}@\mathcal{O}$ and $\mathbf{F_1}@\mathcal{M}$ to evaluate a model's performance.

In table 33, we report detailed absent testing scores when the model trained with `One2Seq` paradigm, using different ordering strategies. For all experiments, we use **R@10** and **R@50** to evaluate a model's performance.

Table 30: Detailed performance (F_1-score) of present keyphrase prediction on six datasets. **Boldface**/<u>Underline</u> text indicates the best/2nd-best performance in corresponding columns.

	Average				Kp20K				Krapivin				Inspec				NUS				SemEval				DUC			
Model	5	10	$\mathcal{O}$	$\mathcal{M}$	5	10	$\mathcal{O}$	$\mathcal{M}$	5	10	$\mathcal{O}$	$\mathcal{M}$	5	10	$\mathcal{O}$	$\mathcal{M}$	5	10	$\mathcal{O}$	$\mathcal{M}$	5	10	$\mathcal{O}$	$\mathcal{M}$	5	10	$\mathcal{O}$	$\mathcal{M}$
One2One variants																												
RNN-O2O-KP20k	30.0	28.7	32.7	15.6	33.1	27.9	35.5	11.4	32.1	26.9	36.0	11.5	29.0	32.4	33.6	23.5	40.2	36.2	43.2	18.1	34.2	35.2	34.8	19.6	11.3	13.9	13.2	9.7
RNN-O2O-KP20k-nocopy	6.2	6.5	6.1	6.6	8.3	8.5	8.4	8.5	5.6	5.8	5.0	5.8	4.9	5.3	5.3	5.4	10.3	10.9	10.4	11.1	8.4	8.4	7.5	9.0	0.0	0.0	0.0	0.0
RNN-O2O-KP20k+MagKP-ALT	31.2	30.6	34.3	13.3	32.4	27.4	34.7	9.0	32.4	28.2	35.6	9.4	32.3	37.8	38.4	21.0	40.2	38.2	43.4	15.1	35.4	35.4	37.4	16.6	14.2	16.4	16.4	8.8
BIGRNN-O2O-KP20k	31.9	31.1	35.3	13.4	35.5	29.5	<u>38.1</u>	9.2	34.3	29.1	38.8	9.4	31.1	36.4	37.2	20.6	42.6	39.3	45.8	15.5	34.5	36.2	37.0	17.4	13.4	16.2	15.0	8.2
BIGRNN-O2O-magkp20k-ALT	31.5	30.7	34.5	12.0	33.2	27.8	35.3	8.1	32.4	28.2	36.5	8.3	32.4	37.1	37.9	19.1	41.3	38.1	44.8	13.6	35.7	35.8	36.0	15.4	14.3	17.3	16.2	7.8
TF-O2O-KP20k	28.3	27.3	31.3	15.2	34.5	28.9	37.4	11.8	29.7	26.5	33.2	11.3	28.1	30.8	32.6	23.7	37.7	36.1	41.3	17.7	33.0	33.0	35.3	19.4	6.9	8.3	7.8	7.1
TF-O2O-KP20k-nocopy	22.0	20.5	23.9	17.4	28.5	24.0	31.2	19.0	24.7	20.9	29.0	16.0	16.6	18.2	18.4	19.0	33.7	30.2	35.3	24.0	25.3	25.9	25.8	22.1	3.4	4.1	3.7	4.2
TF-O2O-KP20k+MagKP-ALT	29.9	29.0	32.5	14.0	33.8	28.2	36.5	10.1	32.8	27.6	35.0	9.7	29.0	33.5	34.0	22.5	39.6	37.9	42.0	15.9	33.7	34.8	<u>35.7</u>	18.3	10.8	12.2	11.9	7.7
One2Seq variants (trained in PRES-ABS)																												
RNN-O2S-KP20k	29.7	29.9	32.2	22.4	31.2	26.1	31.2	16.4	31.0	27.0	33.5	18.2	32.9	38.8	38.6	32.8	37.4	36.6	39.2	25.6	33.7	35.1	36.2	26.7	11.8	15.9	14.7	14.5
RNN-O2S-KP20k-nocopy	7.1	7.0	7.5	7.0	10.4	10.2	11.1	10.2	8.1	7.9	9.9	7.9	4.5	4.5	4.5	4.5	11.0	10.5	10.9	10.6	8.8	8.6	8.8	8.6	0.1	0.1	0.1	0.1
RNN-O2S+KP20k+MagKP-ALT	28.1	29.1	31.0	22.0	28.3	23.9	28.2	15.3	28.1	25.8	30.7	17.1	32.9	<u>40.3</u>	39.6	33.4	35.2	33.4	36.4	23.8	30.9	33.2	34.2	26.2	13.3	18.3	17.2	<u>16.1</u>
BIGRNN-O2S-KP20k	28.9	29.1	31.7	22.7	30.2	25.7	30.4	16.7	29.9	26.4	32.4	18.2	31.9	37.7	37.9	32.5	37.6	35.7	39.8	27.1	32.6	33.7	35.5	27.3	11.5	15.0	14.1	14.5
BIGRNN-O2S-KP20k+MagKP-ALT	29.1	29.6	31.6	21.3	28.2	23.7	28.2	15.0	29.0	25.7	31.0	16.9	**34.8**	**41.2**	**40.1**	32.2	36.0	34.5	37.8	24.3	32.1	33.8	34.8	24.0	14.3	<u>18.6</u>	17.4	15.3
TF-O2S-KP20k	30.6	29.5	33.0	25.6	34.6	29.0	36.2	21.6	32.6	28.2	36.4	21.9	31.7	36.8	37.1	34.7	40.3	37.4	42.2	32.2	33.9	34.1	34.8	30.9	10.2	11.5	11.1	12.1
TF-O2S-KP20k-nocopy	25.7	24.5	27.2	23.8	32.4	29.0	33.9	**28.0**	28.5	25.1	31.6	**24.2**	23.4	24.7	25.2	24.8	37.1	34.6	37.5	**33.6**	27.4	28.4	29.5	26.9	5.3	5.1	5.2	5.2
TF-O2S-KP20k+MagKP-ALT	<u>32.8</u>	32.2	35.9	25.3	36.8	30.2	37.7	19.3	35.3	30.0	37.6	20.6	32.4	38.9	39.4	36.1	41.8	39.3	44.2	29.7	<u>35.7</u>	36.6	<u>39.0</u>	30.1	14.9	18.5	<u>17.6</u>	16.0
TRANSFORMER+One2Seq (trained in PRES-ABS)																												
MagKP-LN-ONLY	26.2	26.8	27.7	25.4	28.1	25.2	28.0	21.1	27.9	26.4	28.7	<u>24.1</u>	29.7	34.5	34.3	35.8	33.7	34.2	35.0	32.0	29.0	30.3	30.4	29.4	8.6	10.0	9.4	10.2
MagKP-Nsmall-ONLY	22.4	23.3	23.4	22.7	20.8	20.0	20.9	18.9	25.3	24.3	26.0	23.3	31.0	34.5	34.1	33.7	26.3	27.2	27.1	26.3	24.1	26.3	24.9	26.4	6.9	7.6	7.5	7.8
MagKP-Nlarge-ONLY	22.0	23.3	23.6	22.6	20.5	19.7	21.1	18.5	24.8	23.7	25.6	22.5	32.9	36.7	36.4	35.8	26.1	26.8	28.2	26.4	21.4	25.0	23.2	24.4	6.1	7.8	7.3	7.9
MagKP-ONLY	25.0	26.9	27.3	25.0	25.3	22.4	25.5	18.9	26.1	25.2	27.8	22.1	31.5	39.0	37.4	37.6	29.9	31.1	31.4	28.8	26.5	30.3	29.2	28.7	11.0	13.3	12.6	13.6
MagKP-LN-MX	31.5	30.8	34.4	24.8	35.6	29.3	36.9	19.9	34.3	28.6	37.9	20.5	31.6	38.0	37.8	35.0	41.7	38.7	44.4	30.0	32.7	35.1	34.4	29.0	13.5	15.4	14.8	14.6
MagKP-Nsmall-MX	31.1	30.8	33.6	26.5	35.4	29.2	36.5	<u>21.4</u>	34.2	28.8	37.0	23.0	31.9	38.4	37.3	37.5	40.6	38.7	42.6	<u>32.8</u>	33.2	36.1	35.4	30.7	11.1	13.4	12.5	13.3
MagKP-Nlarge-MX	31.8	31.1	34.7	26.5	36.3	30.0	37.1	21.1	35.0	29.7	37.0	23.3	31.9	37.9	38.3	<u>38.2</u>	42.0	39.7	44.9	31.8	34.3	35.2	37.6	30.7	11.4	13.9	13.4	13.8
MagKP-MX	32.2	32.1	34.9	**26.9**	36.3	29.8	37.4	20.2	35.1	30.1	37.2	23.0	32.1	39.8	38.5	**38.9**	41.4	<u>40.0</u>	44.8	31.5	34.9	<u>36.7</u>	36.8	<u>31.4</u>	13.5	16.0	14.8	<u>16.1</u>
MagKP-LN-FT	32.1	31.3	34.7	25.9	36.2	29.8	37.7	21.0	35.1	29.6	36.2	21.5	32.3	38.1	38.3	36.8	41.5	39.5	44.9	31.0	35.5	36.2	36.8	31.0	11.9	14.8	14.1	14.4
MagKP-Nsmall-FT	32.7	31.8	35.7	26.5	36.4	30.1	37.7	20.6	35.8	30.6	**39.4**	22.5	32.6	38.4	38.5	37.6	43.2	39.7	45.5	31.3	35.1	35.7	37.5	31.1	13.3	16.5	15.4	<u>16.1</u>
MagKP-Nlarge-FT	**33.6**	**32.9**	**36.6**	26.0	<u>37.0</u>	<u>30.3</u>	37.9	19.9	**36.7**	**30.8**	<u>39.0</u>	22.0	<u>33.5</u>	39.6	<u>39.8</u>	36.9	<u>44.0</u>	**40.2**	**47.8**	30.2	34.4	36.5	35.9	31.0	**16.0**	**19.7**	**19.2**	**16.2**
MagKP-FT	**33.6**	<u>32.3</u>	<u>36.4</u>	<u>26.7</u>	**37.1**	**30.5**	**38.3**	20.6	<u>36.3</u>	<u>30.7</u>	38.5	22.8	32.6	38.1	38.5	36.5	**44.1**	**40.2**	<u>46.1</u>	32.6	**36.7**	**37.1**	**39.3**	**31.9**	<u>15.1</u>	17.4	17.4	15.8

Table 31: Detailed performance (recall scores) of absent keyphrase prediction on six datasets. **Boldface**/<u>Underline</u> text indicates the best/2nd-best performance in corresponding columns.

	Average			**Kp20K**			**Inspec**			**Krapivin**			**NUS**			**SemEval**			**DUC**		
Model	**10**	**50**	$\mathcal{M}$	**10**	**50**	$\mathcal{M}$	**10**	**50**	$\mathcal{M}$	**10**	**50**	$\mathcal{M}$	**10**	**50**	$\mathcal{M}$	**10**	**50**	$\mathcal{M}$	**10**	**50**	$\mathcal{M}$
`One2One variants`																					
RNN-O2O-KP20k	4.4	9.0	14.0	6.8	13.2	19.7	7.9	13.9	20.9	4.5	9.3	15.0	4.7	11.3	17.6	2.2	6.1	10.0	0.0	0.0	<u>0.6</u>
RNN-O2O-KP20k-nocopy	1.2	3.5	6.2	2.4	5.8	9.8	1.2	4.0	8.5	1.4	2.3	3.3	1.5	5.6	9.6	0.4	3.5	6.1	0.0	0.0	0.0
RNN-O2O-KP20k+MagKP-ALT	5.2	9.9	15.2	6.6	13.3	19.5	8.4	16.3	23.9	5.7	10.6	18.1	7.0	13.8	18.8	3.2	5.2	10.0	0.0	0.1	**0.8**
BIGRNN-O2O-KP20k	7.6	14.2	19.7	11.5	20.4	28.0	13.0	22.7	31.7	6.0	12.6	18.9	10.1	18.6	24.9	4.3	<u>10.1</u>	13.7	**0.4**	**0.8**	**0.8**
BIGRNN-O2O-magkp20k-ALT	6.7	12.3	18.2	9.1	16.7	23.5	12.5	20.9	29.1	7.1	12.4	<u>19.7</u>	7.6	15.1	23.5	3.6	8.2	12.8	<u>0.3</u>	0.3	0.4
TF-O2O-KP20k	7.6	14.3	20.9	12.7	<u>22.3</u>	<u>30.9</u>	13.0	<u>24.0</u>	33.3	4.7	10.4	17.6	10.1	18.9	27.3	4.8	9.9	15.7	0.0	0.2	0.2
TF-O2O-KP20k-nocopy	8.9	**15.5**	**22.5**	14.0	**24.4**	**33.5**	15.2	**25.9**	**39.0**	5.1	10.4	14.1	**13.2**	**22.1**	**30.9**	6.1	**10.5**	**17.2**	0.0	0.0	0.1
TF-O2O-KP20k+MagKP-ALT	8.3	<u>14.9</u>	<u>21.7</u>	12.5	21.7	30.4	13.2	23.9	<u>34.7</u>	6.2	13.3	**21.1**	11.4	<u>20.2</u>	<u>27.4</u>	6.1	<u>10.1</u>	<u>16.1</u>	0.1	0.2	0.2
`One2Seq variants` (trained in PRES-ABS)																					
RNN-O2S-KP20k	2.2	2.5	2.5	2.7	3.2	3.3	2.8	3.2	3.3	3.5	3.8	3.8	2.6	2.9	3.0	1.6	1.7	1.7	0.0	0.0	0.0
RNN-O2S-KP20k-nocopy	3.1	4.1	4.2	4.9	6.7	6.7	4.6	6.3	6.4	1.7	2.0	2.0	4.5	6.2	6.2	2.6	3.6	3.6	0.0	0.0	0.0
RNN-O2S+KP20k+MagKP-ALT	2.0	2.7	2.8	2.6	3.2	3.3	2.4	3.2	3.4	3.7	5.1	5.1	1.4	2.5	2.6	2.0	2.0	2.1	0.0	0.0	0.0
BIGRNN-O2S-KP20k	1.7	1.8	1.8	1.9	2.0	2.0	2.6	2.6	2.6	2.5	2.6	2.6	2.2	2.3	2.3	1.0	1.0	1.0	<u>0.3</u>	0.3	0.3
BIGRNN-O2S-KP20k+MagKP-ALT	2.7	3.1	3.1	3.2	3.7	3.8	4.6	5.4	5.4	3.7	4.2	4.2	2.9	3.6	3.6	1.8	1.9	1.9	0.0	0.0	0.0
TF-O2S-KP20k	7.2	10.1	10.5	11.1	15.1	15.6	12.2	16.8	17.7	5.0	8.1	8.6	9.0	12.5	12.7	5.9	8.3	8.4	0.0	0.0	0.0
TF-O2S-KP20k-nocopy	8.1	11.2	11.7	13.2	18.1	18.5	14.5	20.0	21.1	5.3	7.4	7.5	10.5	15.1	15.8	5.2	6.9	7.2	0.0	0.0	0.0
TF-O2S-KP20k+MagKP-ALT	9.7	12.4	13.2	13.8	16.5	17.1	15.9	19.3	20.7	**10.8**	**15.9**	16.5	<u>12.2</u>	15.3	16.4	5.4	7.4	8.1	0.1	0.3	0.3
`TRANSFORMER+One2Seq +MAGKP` (trained in PRES-ABS)																					
MagKP-LN-ONLY	4.1	5.7	6.9	6.0	8.3	9.6	6.4	9.6	12.0	2.9	5.0	6.2	7.2	8.7	10.4	1.9	2.8	3.3	0.0	0.0	0.0
MagKP-Nsmall-ONLY	3.2	4.2	4.9	3.6	4.5	5.2	7.3	8.8	10.4	4.7	6.2	7.5	1.8	3.0	3.8	1.8	2.5	2.6	0.0	0.0	0.0
MagKP-Nlarge-ONLY	3.3	4.3	4.8	3.4	4.6	5.1	8.3	10.4	11.1	4.4	5.7	6.5	2.2	2.9	3.2	1.6	2.1	2.7	0.0	0.0	0.0
MagKP-ONLY	4.5	6.1	7.5	4.5	6.0	7.4	9.6	12.9	15.2	6.8	9.4	11.9	3.5	4.9	6.4	2.5	3.3	3.8	0.1	0.2	0.2
MagKP-LN-MX	8.9	11.7	12.4	13.0	16.8	17.4	15.3	19.4	19.9	6.7	10.6	11.7	12.0	15.6	17.2	<u>6.3</u>	8.1	8.4	0.0	0.0	0.1
MagKP-Nsmall-MX	8.9	11.7	12.8	13.1	16.8	18.0	15.2	18.7	20.7	8.1	12.7	14.5	<u>12.2</u>	15.2	16.2	4.3	6.1	6.9	0.1	<u>0.4</u>	0.4
MagKP-Nlarge-MX	9.1	11.1	12.1	13.3	15.7	16.6	15.7	19.2	21.5	8.9	12.4	13.8	11.5	13.8	14.5	5.2	5.7	6.1	0.0	0.0	0.0
MagKP-MX	9.1	10.9	12.0	12.4	14.4	15.4	15.9	17.5	19.9	<u>10.6</u>	14.3	16.1	10.5	12.5	13.8	5.2	6.2	6.7	0.2	0.3	0.3
MagKP-LN-FT	8.3	11.0	11.5	13.0	16.4	16.9	13.7	16.9	17.9	7.5	11.1	11.5	10.2	14.4	14.6	5.4	7.3	7.8	0.0	0.0	0.0
MagKP-Nsmall-FT	<u>10.0</u>	12.8	14.3	14.3	18.1	19.4	16.9	20.1	23.0	<u>10.6</u>	15.3	16.6	11.9	15.5	17.4	6.0	7.9	9.2	0.2	0.2	0.2
MagKP-Nlarge-FT	**10.2**	13.6	14.7	**15.1**	19.1	20.5	**17.7**	21.4	23.8	**10.8**	<u>15.4</u>	16.5	11.2	16.8	17.5	**6.4**	8.8	9.5	<u>0.3</u>	0.3	0.4
MagKP-FT	9.9	13.3	14.1	<u>14.8</u>	19.0	19.9	<u>17.4</u>	21.6	23.3	8.9	14.0	15.3	11.5	16.3	17.0	<u>6.3</u>	8.5	8.8	0.2	0.2	0.2

Table 32: Detailed present keyphrase prediction performance (F_1-score) of `One2Seq` trained with different orders.

	Average				Kp20K				Krapivin				Inspec				NUS				SemEval				DUC			
Model	5	10	$\mathcal{O}$	$\mathcal{M}$	5	10	$\mathcal{O}$	$\mathcal{M}$	5	10	$\mathcal{O}$	$\mathcal{M}$	5	10	$\mathcal{O}$	$\mathcal{M}$	5	10	$\mathcal{O}$	$\mathcal{M}$	5	10	$\mathcal{O}$	$\mathcal{M}$	5	10	$\mathcal{O}$	$\mathcal{M}$
RNN Greedy																												
RANDOM	21.6	21.6	21.5	21.6	28.6	28.6	28.5	28.6	27.2	27.2	27.0	27.2	17.7	17.7	17.8	17.7	29.2	29.2	28.7	29.2	21.1	21.1	21.3	21.1	5.4	5.4	5.4	5.4
ALPHA	24.7	24.7	23.9	24.7	29.0	29.0	27.3	29.0	28.4	28.4	26.2	28.4	22.6	22.6	22.4	22.6	31.9	31.8	31.0	31.8	28.2	28.2	28.3	28.2	8.1	8.2	8.2	8.2
ALPHA-REV	24.5	24.5	23.4	24.5	29.4	29.4	27.5	29.4	27.9	28.0	25.5	28.0	22.4	22.3	21.9	22.3	31.1	31.1	30.6	31.1	28.0	27.9	26.4	27.9	8.2	8.2	8.2	8.2
S->L	24.1	24.1	23.9	24.1	31.0	31.0	29.8	31.0	30.2	30.2	29.3	30.2	19.6	19.6	19.4	19.6	32.0	32.0	33.1	32.0	25.8	25.8	26.0	25.8	5.6	5.6	5.6	5.6
L->S	23.6	23.6	23.1	23.6	28.1	28.1	27.0	28.1	27.7	27.7	26.3	27.7	21.7	21.7	21.7	21.7	28.9	28.9	28.7	28.9	25.2	25.2	24.9	25.2	9.8	9.8	9.8	9.8
ORI	24.4	24.4	24.5	24.4	30.5	30.5	30.7	30.5	30.6	30.6	31.0	30.6	22.0	22.0	21.9	22.0	31.8	31.8	31.7	31.8	24.5	24.5	24.6	24.5	7.3	7.3	7.3	7.3
ORI-REV	21.5	21.5	21.3	21.5	28.4	28.4	27.7	28.4	26.9	26.9	26.0	26.9	18.1	18.1	18.1	18.1	27.2	27.2	27.3	27.2	22.5	22.5	22.5	22.5	6.2	6.2	6.2	6.2
PRES-ABS	26.9	26.9	26.7	26.9	31.8	31.8	31.4	31.8	31.3	31.4	31.4	31.4	24.7	24.8	24.8	24.8	35.1	35.2	34.6	35.2	30.3	30.4	30.0	30.4	8.1	8.1	8.1	8.1
ABS-PRES	24.3	24.3	24.2	24.3	29.0	29.0	28.2	29.0	29.1	29.1	29.3	29.1	22.4	22.3	22.3	22.3	31.1	31.1	30.9	31.1	26.7	26.7	26.8	26.7	7.7	7.7	7.7	7.7
TRANSFORMER Greedy																												
RANDOM	24.2	24.2	24.3	24.2	33.3	33.3	33.1	33.3	28.9	28.9	29.6	28.9	20.6	20.5	20.6	20.5	31.6	31.6	31.8	31.6	24.9	24.9	25.0	24.9	5.9	5.9	5.9	5.9
ALPHA	23.5	23.5	22.4	23.5	32.8	32.9	29.8	32.9	25.9	25.8	24.2	25.8	21.8	21.8	21.8	21.8	30.2	30.2	28.9	30.2	24.9	25.0	24.9	25.0	5.2	5.2	5.2	5.2
ALPHA-REV	23.4	23.4	22.8	23.4	30.9	30.9	29.1	30.9	26.1	26.1	24.4	26.1	22.5	22.5	22.5	22.5	29.8	29.7	29.8	29.7	25.0	25.0	24.9	25.0	6.0	6.0	6.0	6.0
S->L	24.7	24.7	24.3	24.7	33.3	33.3	31.9	33.3	31.0	31.0	30.0	31.0	21.9	21.9	22.0	21.9	30.4	30.4	30.8	30.4	25.2	25.2	24.8	25.2	6.3	6.3	6.3	6.3
L->S	23.5	23.5	23.3	23.5	29.8	29.8	28.9	29.8	26.9	26.9	26.2	26.9	23.2	23.2	23.2	23.2	31.0	30.9	31.5	30.9	23.6	23.6	23.8	23.6	6.3	6.3	6.3	6.3
ORI	24.5	24.5	24.7	24.5	32.8	32.8	33.4	32.8	27.6	27.6	27.7	27.6	22.3	22.3	22.3	22.3	31.6	31.6	31.9	31.6	26.7	26.7	27.1	26.7	5.9	5.9	5.9	5.9
ORI-REV	22.7	22.7	22.5	22.7	30.6	30.6	29.5	30.6	26.4	26.4	26.0	26.4	19.7	19.7	19.8	19.7	30.2	30.2	30.5	30.2	24.1	24.1	23.8	24.1	5.3	5.3	5.3	5.3
PRES-ABS	26.8	26.8	27.5	26.8	34.0	34.0	34.7	34.0	31.8	31.9	33.8	31.9	25.8	25.8	25.7	25.8	34.2	34.3	35.2	34.3	28.0	28.0	28.4	28.0	6.9	7.0	7.0	7.0
ABS-PRES	20.3	20.3	20.3	20.3	31.2	31.3	31.0	31.3	23.8	23.8	24.3	23.8	16.1	16.1	16.1	16.1	25.8	25.8	25.6	25.8	21.2	21.2	21.3	21.2	3.7	3.7	3.7	3.7
RNN beam50																												
RANDOM	30.3	29.3	33.0	24.4	31.9	25.9	33.9	19.6	32.0	26.8	34.3	20.5	31.9	36.3	37.0	33.2	39.2	35.4	42.7	28.9	34.5	35.9	35.8	29.8	12.4	15.6	14.5	14.3
ALPHA	30.8	30.2	33.2	22.6	32.6	27.1	33.5	17.4	33.0	28.1	34.4	18.9	31.8	37.5	37.5	31.3	39.6	37.4	42.7	26.7	34.5	35.2	35.8	26.9	13.5	15.9	15.4	14.1
ALPHA-REV	30.3	29.7	32.7	23.3	32.6	27.2	33.0	18.5	32.1	27.6	34.0	19.5	30.9	36.1	37.1	31.8	39.8	37.4	41.6	27.9	33.3	33.9	35.6	28.4	13.2	16.0	14.9	13.6
S->L	30.8	29.6	33.3	21.2	32.6	26.6	33.6	17.0	33.4	27.3	34.7	18.3	32.5	36.6	37.5	29.5	38.4	35.3	40.2	24.3	33.9	34.7	37.3	25.4	14.2	17.0	16.7	13.0
L->S	31.4	30.6	33.4	27.7	32.4	27.4	32.8	22.6	33.5	28.2	34.6	23.9	33.3	38.1	38.4	36.1	38.5	37.5	41.5	33.0	35.4	34.4	36.4	33.3	15.1	17.9	17.1	17.2
ORI	31.6	30.4	34.4	23.2	32.4	26.2	34.4	17.2	33.1	28.0	36.5	19.2	34.5	39.8	41.1	34.3	40.0	36.3	42.0	26.8	35.5	35.3	36.5	27.4	14.0	16.8	15.8	14.6
ORI-REV	31.1	29.6	33.1	26.6	32.0	26.9	33.6	22.9	32.3	27.5	34.6	23.5	33.0	35.5	37.1	32.4	39.6	36.4	40.8	32.8	35.2	34.1	35.9	31.8	14.7	17.0	16.7	16.4
PRES-ABS	29.7	29.9	32.2	22.4	31.2	26.1	31.2	16.4	31.0	27.0	33.5	18.2	32.9	38.8	38.6	32.8	37.4	36.6	39.2	25.6	33.7	35.1	36.2	26.7	11.8	15.9	14.7	14.5
ABS-PRES	29.1	29.9	32.1	19.7	31.7	27.0	32.7	13.8	31.0	27.7	34.3	15.5	31.7	37.1	37.6	29.9	38.3	37.5	40.9	22.8	31.5	35.3	34.6	23.5	10.5	14.5	12.8	12.7
TRANSFORMER beam50																												
RANDOM	30.3	28.8	32.1	25.4	34.4	28.2	35.8	22.8	33.9	29.0	34.9	23.8	31.3	34.9	35.4	32.9	38.9	36.9	41.3	32.4	34.0	33.4	35.5	29.8	9.3	10.5	9.9	10.7
ALPHA	30.4	28.9	31.9	25.2	34.7	29.2	35.2	22.8	33.4	27.6	34.2	21.5	30.9	34.3	35.0	33.2	39.9	37.4	42.1	32.4	33.8	33.9	33.9	30.6	9.9	11.2	10.8	11.1
ALPHA-REV	30.3	29.8	33.2	25.4	34.7	29.0	35.5	21.8	32.7	27.5	36.5	21.7	30.4	36.1	35.6	34.0	40.4	37.6	44.3	31.4	31.6	34.8	34.0	30.6	12.0	13.8	13.1	12.7
S->L	30.4	29.3	32.0	25.6	34.4	28.5	34.9	22.2	33.7	28.5	34.6	23.5	31.1	35.6	35.8	33.1	40.5	37.9	41.6	32.3	32.6	34.0	33.7	31.2	10.1	11.6	11.6	11.4
L->S	30.8	29.9	32.7	25.9	34.7	29.0	35.5	22.6	33.5	29.0	34.7	22.7	31.8	35.6	36.3	33.6	40.7	38.4	42.4	32.5	33.5	34.3	35.3	31.5	10.6	12.9	12.2	12.7
ORI	30.2	29.4	32.4	25.3	34.4	28.5	36.1	21.8	33.6	29.2	34.2	23.2	31.9	36.6	35.9	34.1	38.5	37.6	41.7	31.9	34.3	34.1	36.8	29.9	8.7	10.2	9.6	10.6
ORI-REV	31.0	30.1	33.2	27.1	34.6	28.9	35.5	24.1	33.9	28.7	35.5	24.5	32.7	36.3	37.5	33.8	40.1	38.3	42.7	34.3	32.5	34.8	34.5	32.3	11.9	13.8	13.4	13.5
PRES-ABS	30.6	29.5	33.0	25.6	34.6	29.0	36.2	21.6	32.6	28.2	36.4	21.9	31.7	36.8	37.1	34.7	40.3	37.4	42.2	32.2	33.9	34.1	34.8	30.9	10.2	11.5	11.1	12.1
ABS-PRES	29.7	29.3	32.0	22.3	34.3	28.9	35.4	17.9	32.5	28.0	35.9	18.0	30.7	34.9	34.8	32.5	38.4	38.4	40.9	26.9	33.1	34.5	34.4	27.9	9.4	11.1	10.6	10.3

Table 33: Detailed absent keyphrase prediction performance (recall scores) of `One2Seq` trained with different orders.

	Average			Kp20K			Inspec			Krapivin			NUS			SemEval			DUC		
Model	**10**	**50**	$\mathcal{M}$	**10**	**50**	$\mathcal{M}$	**10**	**50**	$\mathcal{M}$	**10**	**50**	$\mathcal{M}$	**10**	**50**	$\mathcal{M}$	**10**	**50**	$\mathcal{M}$	**10**	**50**	$\mathcal{M}$
RNN Greedy																					
RANDOM	0.2	0.2	0.2	0.2	0.2	0.2	0.4	0.4	0.4	0.1	0.1	0.1	0.2	0.2	0.2	0.3	0.3	0.3	0.0	0.0	0.0
ALPHA	0.2	0.2	0.2	0.3	0.3	0.3	0.3	0.3	0.3	0.2	0.2	0.2	0.0	0.0	0.0	0.7	0.7	0.7	0.0	0.0	0.0
ALPHA-REV	0.2	0.2	0.2	0.2	0.2	0.2	0.4	0.4	0.4	0.1	0.1	0.1	0.1	0.1	0.1	0.3	0.3	0.3	0.0	0.0	0.0
S->L	0.1	0.1	0.1	0.2	0.2	0.2	0.1	0.1	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.2	0.2	0.0	0.0	0.0
L->S	0.2	0.2	0.2	0.2	0.2	0.2	0.6	0.6	0.6	0.0	0.0	0.0	0.0	0.0	0.0	0.6	0.6	0.6	0.0	0.0	0.0
ORI	0.3	0.3	0.3	0.2	0.2	0.2	0.6	0.6	0.6	0.1	0.1	0.1	0.2	0.2	0.2	0.5	0.5	0.5	0.0	0.0	0.0
ORI-REV	0.1	0.1	0.1	0.2	0.2	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.3	0.3	0.3	0.0	0.0	0.0
PRES-ABS	0.2	0.2	0.2	0.2	0.2	0.2	0.5	0.5	0.5	0.3	0.3	0.3	0.2	0.2	0.2	0.3	0.3	0.3	0.0	0.0	0.0
ABS-PRES	0.2	0.2	0.2	0.3	0.3	0.3	0.4	0.4	0.4	0.3	0.3	0.3	0.0	0.0	0.0	0.4	0.4	0.4	0.0	0.0	0.0
TRANSFORMER Greedy																					
RANDOM	1.9	1.9	1.9	2.8	2.8	2.8	2.4	2.4	2.4	0.8	0.8	0.8	3.1	3.1	3.1	2.1	2.1	2.1	0.0	0.0	0.0
ALPHA	1.6	1.6	1.6	3.0	3.0	3.0	3.6	3.6	3.6	0.8	0.8	0.8	0.9	0.9	0.9	1.1	1.1	1.1	0.0	0.0	0.0
ALPHA-REV	1.6	1.6	1.6	2.5	2.5	2.5	3.9	3.9	3.9	1.1	1.1	1.1	1.4	1.4	1.4	0.7	0.7	0.7	0.0	0.0	0.0
S->L	1.5	1.5	1.5	2.8	2.8	2.8	2.5	2.5	2.5	0.4	0.4	0.4	1.8	1.8	1.8	1.3	1.3	1.3	0.0	0.0	0.0
L->S	1.3	1.3	1.3	2.4	2.4	2.4	2.4	2.4	2.4	0.6	0.6	0.6	1.6	1.6	1.6	0.8	0.8	0.8	0.0	0.0	0.0
ORI	1.5	1.5	1.5	2.5	2.5	2.5	3.3	3.3	3.3	1.1	1.1	1.1	1.2	1.2	1.2	0.9	0.9	0.9	0.1	0.1	0.1
ORI-REV	1.1	1.1	1.1	2.5	2.5	2.5	2.2	2.2	2.2	0.5	0.5	0.5	0.8	0.8	0.8	0.8	0.8	0.8	0.0	0.0	0.0
PRES-ABS	1.3	1.3	1.3	2.3	2.3	2.3	2.7	2.7	2.7	0.8	0.8	0.8	1.0	1.0	1.0	1.2	1.2	1.2	0.0	0.0	0.0
ABS-PRES	1.8	1.8	1.8	2.8	2.8	2.8	4.0	4.0	4.0	0.7	0.7	0.7	2.0	2.0	2.0	1.6	1.6	1.6	0.0	0.0	0.0
RNN beam50																					
RANDOM	1.9	2.1	2.1	2.4	2.7	2.7	2.4	2.7	2.7	3.3	3.6	3.6	1.5	1.9	1.9	1.5	1.6	1.6	0.0	0.0	0.0
ALPHA	2.1	2.4	2.4	2.5	2.7	2.7	2.9	3.1	3.1	3.4	3.8	3.8	2.4	2.6	2.6	1.8	2.0	2.0	0.0	0.0	0.0
ALPHA-REV	1.4	1.5	1.5	2.0	2.1	2.1	2.1	2.2	2.2	1.7	1.7	1.7	2.1	2.2	2.2	0.6	0.8	0.8	0.0	0.0	0.0
S->L	1.7	1.9	1.9	2.1	2.2	2.2	2.1	2.2	2.2	2.8	3.6	3.6	1.3	1.4	1.4	1.9	2.0	2.0	0.0	0.0	0.2
L->S	1.4	1.4	1.4	1.8	1.8	1.8	2.1	2.1	2.1	2.5	2.5	2.5	1.1	1.1	1.1	0.8	0.8	0.8	0.0	0.0	0.0
ORI	1.9	2.3	2.3	2.3	2.7	2.7	2.7	3.2	3.2	3.0	3.6	3.6	2.3	2.8	2.8	1.2	1.9	1.9	0.0	0.0	0.0
ORI-REV	1.5	1.6	1.6	1.9	2.0	2.0	2.0	2.0	2.0	2.5	2.5	2.5	1.3	1.4	1.4	1.5	1.6	1.6	0.0	0.0	0.0
PRES-ABS	2.2	2.5	2.5	2.7	3.2	3.3	2.8	3.2	3.3	3.5	3.8	3.8	2.6	2.9	3.0	1.6	1.7	1.7	0.0	0.0	0.0
ABS-PRES	1.1	1.1	1.1	1.2	1.3	1.3	1.5	1.5	1.5	1.6	1.6	1.6	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0
TRANSFORMER beam50																					
RANDOM	6.7	8.0	8.2	10.2	12.5	12.8	11.5	13.1	13.3	3.8	5.7	5.8	10.0	11.2	11.5	4.9	5.7	6.0	0.1	0.1	0.1
ALPHA	7.3	9.6	9.7	11.8	14.8	14.9	12.9	17.2	17.3	5.0	7.2	7.5	9.1	11.5	11.5	5.2	6.7	6.7	0.0	0.1	0.1
ALPHA-REV	6.5	9.7	9.8	10.3	14.3	14.6	10.2	15.4	15.6	3.8	5.7	5.9	8.9	15.6	15.8	5.7	7.2	7.2	0.0	0.0	0.0
S->L	6.5	8.5	8.6	10.3	13.1	13.2	11.0	14.2	14.3	3.7	6.0	6.0	9.0	11.0	11.3	5.0	6.6	6.9	0.0	0.0	0.0
L->S	6.8	9.5	9.8	11.1	15.3	15.7	11.6	15.1	15.6	4.2	6.7	7.4	9.2	12.7	12.8	4.9	7.1	7.2	0.0	0.0	0.0
ORI	7.0	10.0	10.4	11.2	15.3	15.7	12.3	16.9	17.6	4.1	6.4	6.7	9.9	14.4	14.9	4.5	7.0	7.3	0.1	0.1	0.1
ORI-REV	7.2	10.4	10.8	11.3	16.0	16.6	12.2	16.9	17.3	4.6	7.2	7.9	9.8	13.7	14.3	5.3	8.2	8.3	0.0	0.3	0.3
PRES-ABS	7.2	10.1	10.5	11.1	15.1	15.6	12.2	16.8	17.7	5.0	8.1	8.6	9.0	12.5	12.7	5.9	8.3	8.4	0.0	0.0	0.0
ABS-PRES	6.2	7.0	7.1	10.2	11.4	11.4	10.6	11.3	11.3	4.2	5.7	5.9	7.5	7.8	7.8	4.9	6.0	6.0	0.0	0.0	0.0

BIBLIOGRAPHY

[1] M. Allamanis, H. Peng, and C. Sutton. A Convolutional Attention Network for Extreme Summarization of Source Code. *ArXiv e-prints*, February 2016.

[2] Rabah Alzaidy, Cornelia Caragea, and C Lee Giles. Bi-lstm-crf sequence labeling for keyphrase extraction from scholarly documents. In *The world wide web conference*, pages 2551–2557, 2019.

[3] Shima Asaadi, Saif Mohammad, and Svetlana Kiritchenko. Big bird: A large, fine-grained, bigram relatedness dataset for examining semantic composition. 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)*, pages 505–516, 2019.

[4] American Medical Informatics Association. Amia 2023 annual symposium call for participation. `https://amia.org/education-events/amia-2023-annual-symposium/call-participation`.

[5] Prafulla Bafna, Dhanya Pramod, Shailaja Shrwaikar, and Atiya Hassan. Semantic key phrase-based model for document management. *Benchmarking: An International Journal*, 2019.

[6] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[7] Hareesh Bahuleyan and Layla El Asri. Diverse keyphrase generation with neural unlikelihood training. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5271–5287, 2020.

[8] Jie Bao, Prithwish Basu, Mike Dean, Craig Partridge, Ananthram Swami, Will Leland, and James A Hendler. Towards a theory of semantic communication. In *2011 IEEE Network Science Workshop*, pages 110–117. IEEE, 2011.

[9] Christos Baziotis, Barry Haddow, and Alexandra Birch. Language model prior for low-resource neural machine translation. *arXiv preprint arXiv:2004.14928*, 2020.

[10] Slobodan Beliga, Ana Meštrović, and Sanda Martinčić-Ipšić. An overview of graph-based keyword extraction methods and approaches. *Journal of information and organizational sciences*, 39(1):1–20, 2015.

[11] Iz Beltagy, Arman Cohan, and Kyle Lo. Scibert: Pretrained contextualized embeddings for scientific text. *arXiv preprint arXiv:1903.10676*, 2019.

[12] Fabiano Belém, Jussara Almeida, and Marcos Gonçalves. *Tagging and Tag Recommendation*. 09 2019.

[13] Kamil Bennani-Smires, Claudiu Musat, Andreea Hossmann, Michael Baeriswyl, and Martin Jaggi. Simple unsupervised keyphrase extraction using sentence embeddings. In Anna Korhonen and Ivan Titov, editors, *CoNLL*, pages 221–229. Association for Computational Linguistics, 2018.

[14] Gábor Berend. Opinion expression mining by exploiting keyphrase extraction. In *IJCNLP*, pages 1162–1170. Citeseer, 2011.

[15] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[16] Rishi Bommasani, Percy Liang, and Tony Lee. Holistic evaluation of language models. *Annals of the New York Academy of Sciences*, 2023.

[17] Marina Bondi and Mike Scott. *Keyness in texts*, volume 41. John Benjamins Publishing, 2010.

[18] Florian Boudin. Unsupervised keyphrase extraction with multipartite graphs. In Marilyn A. Walker, Heng Ji, and Amanda Stent, editors, *NAACL-HLT (2)*, pages 667–672. Association for Computational Linguistics, 2018.

[19] Adrien Bougouin, Florian Boudin, and Béatrice Daille. Topicrank: Graph-based topic ranking for keyphrase extraction. In *IJCNLP*, pages 543–551. Asian Federation of Natural Language Processing / ACL, 2013.

[20] Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. Domain separation networks. *CoRR*, abs/1608.06019, 2016.

[21] Laurel J Brinton and Donna Brinton. *The linguistic structure of modern English*. John Benjamins Publishing, 2010.

[22] Christopher H Brooks and Nancy Montanez. Improved annotation of the blogosphere via autotagging and hierarchical clustering. In *Proceedings of the 15th international conference on World Wide Web*, pages 625–632, 2006.

[23] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[24] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc.

[25] Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Mário Jorge, Célia Nunes, and Adam Jatowt. Yake! collection-independent automatic keyword extractor. In Gabriella Pasi, Benjamin Piwowarski, Leif Azzopardi, and Allan Hanbury, editors, *ECIR*, volume 10772 of *Lecture Notes in Computer Science*, pages 806–810. Springer, 2018.

[26] Erion Çano and Ondřej Bojar. Keyphrase generation: A multi-aspect survey. *arXiv preprint arXiv:1910.05059*, 2019.

[27] Sérgio D Canuto, Fabiano M Belém, Jussara M Almeida, and Marcos A Gonçalves. A comparative study of learning-to-rank techniques for tag recommendation. *Journal of Information and Data Management*, 4(3):453–453, 2013.

[28] Hou Pong Chan, Wang Chen, Lu Wang, and Irwin King. Neural keyphrase generation via reinforcement learning with adaptive rewards. *arXiv preprint arXiv:1906.04106*, 2019.

[29] Yves Chauvin and David E Rumelhart. *Backpropagation: theory, architectures, and applications*. Psychology press, 2013.

[30] Jun Chen, Xiaoming Zhang, Yu Wu, Zhao Yan, and Zhoujun Li. Keyphrase generation with correlation constraints. *CoRR*, abs/1808.07185, 2018.

[31] Minmin Chen, Alice Zheng, and Kilian Weinberger. Fast image tagging. In *International conference on machine learning*, pages 1274–1282, 2013.

[32] Wang Chen, Hou Pong Chan, Piji Li, Lidong Bing, and Irwin King. An integrated approach for keyphrase generation via exploring the power of retrieval and extraction. *arXiv preprint arXiv:1904.03454*, 2019.

[33] Wang Chen, Hou Pong Chan, Piji Li, and Irwin King. Exclusive hierarchical decoding for deep keyphrase generation. *arXiv preprint arXiv:2004.08511*, 2020.

[34] Wang Chen, Yifan Gao, Jiani Zhang, Irwin King, and Michael R. Lyu. Title-guided encoding for keyphrase generation. *CoRR*, abs/1808.08575, 2018.

[35] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[36] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[37] Keunwoo Choi, George Fazekas, and Mark Sandler. Automatic tagging using deep convolutional neural networks. *arXiv preprint arXiv:1606.00298*, 2016.

[38] Jishnu Ray Chowdhury, Seo Yeon Park, Tuhin Kundu, and Cornelia Caragea. Kpdrop: Improving absent keyphrase generation. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 4853–4870, 2022.

[39] Jason Chuang, Christopher D Manning, and Jeffrey Heer. "without the clutter of unimportant words" descriptive keyphrases for text visualization. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 19(3):1–29, 2012.

[40] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022.

[41] Mark Culp and George Michailidis. An iterative algorithm for extending learners to a semi-supervised setting. *Journal of Computational and Graphical Statistics*, 17(3):545–571, 2008.

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

[43] Oxford English Dictionary. Iphrasify, v., 2020.

[44] Oxford English Dictionary. keyword, n., 2020.

[45] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. Unified language model pre-training for natural language understanding and generation. *Advances in neural information processing systems*, 32, 2019.

[46] Ernesto D'Avanzo, Bernardo Magnini, and Alessandro Vallin. Keyphrase extraction for summarization purposes: The lake system at duc-2004. In *Proceedings of the 2004 document understanding conference*, 2004.

[47] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[48] Günes Erkan and Dragomir R Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research*, 22:457–479, 2004.

[49] Umer Farooq, Yang Song, John M Carroll, and C Lee Giles. Social bookmarking for scholarly digital libraries. *IEEE Internet Computing*, 11(6):29–35, 2007.

[50] Flavio Figueiredo, Henrique Pinto, Fabiano BeléM, Jussara Almeida, Marcos GonçAlves, David Fernandes, and Edleno Moura. Assessing the quality of textual features in social media. *Information Processing & Management*, 49(1):222–247, 2013.

[51] Katja Filippova, Enrique Alfonseca, Carlos Colmenares, Lukasz Kaiser, and Oriol Vinyals. Sentence compression by deletion with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP'15)*, 2015.

[52] Nazanin Firoozeh, Adeline Nazarenko, Fabrice Alizon, and Béatrice Daille. Keyword extraction: Issues and methods. *Natural Language Engineering*, 26(3):259–291, 2020.

[53] Corina Florescu and Cornelia Caragea. Positionrank: An unsupervised approach to keyphrase extraction from scholarly documents. In Regina Barzilay and Min-Yen Kan, editors, *ACL (1)*, pages 1105–1115. Association for Computational Linguistics, 2017.

[54] Luciano Floridi. Philosophical conceptions of information. In *Formal theories of information*, pages 13–53. Springer, 2009.

[55] Association for Computing Machinery. How to classify works using acm's computing classification system. `https://www.acm.org/publications/computing-classification-system/how-to-use`.

[56] Kenneth I Forster and Elizabeth S Bednall. Terminating and exhaustive search in lexical access. *Memory & Cognition*, 4(1):53–61, 1976.

[57] Eibe Frank, Gordon W Paynter, Ian H Witten, Carl Gutwin, and Craig G Nevill-Manning. Domain-specific keyphrase extraction. 1999.

[58] Costas Gabrielatos. Keyness analysis: Nature, metrics and techniques. In *Corpus approaches to discourse*, pages 225–258. Routledge, 2018.

[59] Ygor Gallina, Florian Boudin, and Béatrice Daille. Kptimes: A large-scale dataset for keyphrase generation on news documents. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 130–135, 2019.

[60] Ygor Gallina, Florian Boudin, and Béatrice Daille. Large-scale evaluation of keyphrase extraction models. *arXiv preprint arXiv:2003.04628*, 2020.

[61] Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, 2021.

[62] Yifan Gao, Qingyu Yin, Zheng Li, Rui Meng, Tong Zhao, Bing Yin, Irwin King, and Michael Lyu. Retrieval-augmented multilingual keyphrase generation with retriever-generator iterative training. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1233–1246, Seattle, United States, July 2022. Association for Computational Linguistics.

[63] Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. Allennlp: A deep semantic natural language processing platform. *arXiv preprint arXiv:1803.07640*, 2018.

[64] Krishna Garg, Jishnu Ray Chowdhury, and Cornelia Caragea. Keyphrase generation beyond the boundaries of title and abstract. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5809–5821, 2022.

[65] Krishna K Garg, Jishnu Ray Chowdhury, and Cornelia Caragea. Data augmentation for low-resource keyphrase generation. In *The 61st Annual Meeting Of The Association For Computational Linguistics*, 2023.

[66] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv:1811.12231*, 2018.

[67] Abbas Ghaddar and Philippe Langlais. Winer: A wikipedia annotated corpus for named entity recognition. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 413–422, 2017.

[68] Sujatha Das Gollapalli and Cornelia Caragea. Extracting keyphrases from research papers using citation networks. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, pages 1629–1635. AAAI Press, 2014.

[69] Sujatha Das Gollapalli, Xiao-Li Li, and Peng Yang. Incorporating expert knowledge into keyphrase extraction. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[70] Sujatha Das Gollapalli, Xiaoli Li, and Peng Yang. Incorporating expert knowledge into keyphrase extraction. In *AAAI*, pages 3180–3187. AAAI Press, 2017.

[71] Maria Grineva, Maxim Grinev, and Dmitry Lizorkin. Extracting key terms from noisy and multitheme documents. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 661–670, New York, NY, USA, 2009. ACM.

[72] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. Incorporating copying mechanism in sequence-to-sequence learning. *CoRR*, abs/1603.06393, 2016.

[73] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393*, 2016.

[74] Çaglar Gülçehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. Pointing the unknown words. *CoRR*, abs/1603.08148, 2016.

[75] Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. Pointing the unknown words. *arXiv preprint arXiv:1603.08148*, 2016.

[76] James Hartley and Ronald N Kostoff. How useful are 'keywords' in scientific journals? *Journal of Information Science*, 29(5):433–438, 2003.

[77] Kazi Saidul Hasan and Vincent Ng. Conundrums in unsupervised keyphrase extraction: making sense of the state-of-the-art. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 365–373. Association for Computational Linguistics, 2010.

[78] Kazi Saidul Hasan and Vincent Ng. Automatic keyphrase extraction: A survey of the state of the art. In *Proceedings of the 52nd Annual Meeting of the Association for Computational*

Linguistics (Volume 1: Long Papers), pages 1262–1273, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[79] Robert Hecht-Nielsen. Theory of the backpropagation neural network. In *Neural networks for perception*, pages 65–93. Elsevier, 1992.

[80] Paul Heymann, Daniel Ramage, and Hector Garcia-Molina. Social tag prediction. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 531–538, 2008.

[81] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018.

[82] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[83] Thomas Hofmann. Probabilistic latent semantic analysis. *arXiv preprint arXiv:1301.6705*, 2013.

[84] Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spaCy: Industrial-strength Natural Language Processing in Python. 2020.

[85] Anette Hulth. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 216–223. Association for Computational Linguistics, 2003.

[86] Anette Hulth. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 216–223. Association for Computational Linguistics, 2003.

[87] Anette Hulth and Beáta B Megyesi. A study on automatically extracted keywords in text categorization. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 537–544. Association for Computational Linguistics, 2006.

[88] Jacob Jacoby. Information load and decision quality: Some contested issues. *Journal of Marketing Research*, 14(4):569–573, 1977.

[89] Christian Jacquemin and Didier Bourigault. Term extraction and automatic indexing. In *The Oxford handbook of computational linguistics*. 2003.

[90] Yi Jiang, Rui Meng, Yong Huang, Wei Lu, and Jiawei Liu. Generating keyphrases for readers:: A controllable keyphrase generation framework. *Journal of the Association for Information Science and Technology*, 74(7):759–774, 2023.

[91] Karen Spärck Jones. A statistical interpretation of term specificity and its application in retrieval. *J. Documentation*, 60(5):493–502, 2004.

[92] Steve Jones and Mark S Staveley. Phrasier: a system for interactive document retrieval using keyphrases. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 160–167. ACM, 1999.

[93] Michael I Jordan. Serial order: A parallel distributed processing approach. In *Advances in psychology*, volume 121, pages 471–495. Elsevier, 1997.

[94] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77, 2020.

[95] Vidur Joshi, Matthew Peters, and Mark Hopkins. Extending a parser to distant domains using a few dozen partially annotated examples. *arXiv preprint arXiv:1805.06556*, 2018.

[96] Daniel Kelleher and Saturnino Luz. Automatic hypertext keyphrase detection. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, IJCAI'05, pages 1608–1609, San Francisco, CA, USA, 2005. Morgan Kaufmann Publishers Inc.

[97] Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. Semeval-2010 task 5 : Automatic keyphrase extraction from scientific articles. In Katrin Erk and Carlo Strapparava, editors, *SemEval@ACL*, pages 21–26. The Association for Computer Linguistics, 2010.

[98] Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 21–26. Association for Computational Linguistics, 2010.

[99] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[100] Ioannis Korkontzelos, Torsten Zesch, Fabio Massimo Zanzotto, and Chris Biemann. Semeval-2013 task 5: Evaluating phrasal semantics. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 39–47, 2013.

[101] Mikalai Krapivin, Aliaksandr Autayeu, and Maurizio Marchese. Large dataset for keyphrases extraction. Technical Report DISI-09-055, DISI, Trento, Italy, May 2008.

[102] Ralf Krestel, Peter Fankhauser, and Wolfgang Nejdl. Latent dirichlet allocation for tag recommendation. In *Proceedings of the third ACM conference on Recommender systems*, pages 61–68, 2009.

[103] Md Tahmid Rahman Laskar, M Saiful Bari, Mizanur Rahman, Md Amran Hossen Bhuiyan, Shafiq Joty, and Jimmy Xiangji Huang. A systematic study and comprehensive evaluation of chatgpt on benchmark datasets. *arXiv preprint arXiv:2305.18486*, 2023.

[104] Tho Thi Ngoc Le, Minh Le Nguyen, and Akira Shimazu. *Unsupervised Keyphrase Extraction: Introducing New Kinds of Words to Keyphrases*, pages 665–671. Springer International Publishing, Cham, 2016.

[105] Debra A Lelewer and Daniel S Hirschberg. Data compression. *ACM Computing Surveys (CSUR)*, 19(3):261–296, 1987.

[106] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, 2020.

[107] Xinzhe Li, Qianru Sun, Yaoyao Liu, Qin Zhou, Shibao Zheng, Tat-Seng Chua, and Bernt Schiele. Learning to self-train for semi-supervised few-shot classification. *Advances in Neural Information Processing Systems*, 32, 2019.

[108] Xinnian Liang, Shuangzhi Wu, Mu Li, and Zhoujun Li. Unsupervised keyphrase extraction by jointly modeling local and global context. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 155–164, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[109] Junwei Liao, Sefik Emre Eskimez, Liyang Lu, Yu Shi, Ming Gong, Linjun Shou, Hong Qu, and Michael Zeng. Improving readability for automatic speech recognition transcription. *arXiv preprint arXiv:2004.04438*, 2020.

[110] Jialu Liu, Xiang Ren, Jingbo Shang, Taylor Cassidy, Clare R Voss, and Jiawei Han. Representing documents via latent keyphrase inference. In *Proceedings of the 25th international conference on World wide web*, pages 1057–1067, 2016.

[111] Rui Liu, Zheng Lin, and Weiping Wang. Keyphrase prediction with pre-trained language model. *arXiv preprint arXiv:2004.10462*, 2020.

[112] Yixin Liu, Alexander R Fabbri, Pengfei Liu, Yilun Zhao, Linyong Nan, Ruilin Han, Simeng Han, Shafiq Joty, Chien-Sheng Wu, Caiming Xiong, et al. Revisiting the gold standard: Grounding summarization evaluation with robust human evaluation. *arXiv preprint arXiv:2212.07981*, 2022.

[113] Zhiyuan Liu, Xinxiong Chen, Yabin Zheng, and Maosong Sun. Automatic keyphrase extraction by bridging vocabulary gap. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 135–144. Association for Computational Linguistics, 2011.

[114] Zhiyuan Liu, Wenyi Huang, Yabin Zheng, and Maosong Sun. Automatic keyphrase extraction via topic decomposition. In *Proceedings of the 2010 conference on empirical methods in natural language processing*, pages 366–376. Association for Computational Linguistics, 2010.

[115] Zhiyuan Liu, Peng Li, Yabin Zheng, and Maosong Sun. Clustering to find exemplar terms for keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 257–266. Association for Computational Linguistics, 2009.

[116] Lajanugen Logeswaran and Honglak Lee. An efficient framework for learning sentence representations. *CoRR*, abs/1803.02893, 2018.

[117] Patrice Lopez and Laurent Romary. Humb: Automatic key term extraction from scientific articles in grobid. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval '10, pages 248–251, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[118] Yi Luan, Mari Ostendorf, and Hannaneh Hajishirzi. Scientific information extraction with semi-supervised neural tagging. *CoRR*, abs/1708.06075, 2017.

[119] Debanjan Mahata, Navneet Agarwal, Dibya Gautam, Amardeep Kumar, Swapnil Parekh, Yaman Kumar Singla, Anish Acharya, and Rajiv Ratn Shah. Ldkp: A dataset for identifying keyphrases from long scientific documents. *arXiv preprint arXiv:2203.15349*, 2022.

[120] Debanjan Mahata, John Kuriakose, Rajiv Shah, and Roger Zimmermann. Key2vec: Automatic ranked keyphrase extraction from scientific articles using phrase embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 634–639, 2018.

[121] Sumit Chopra Marc'Aurelio Ranzato, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. *ICLR, San Juan, Puerto Rico*, 2016.

[122] Leandro Balby Marinho and Lars Schmidt-Thieme. Collaborative tag recommendations. In *Data Analysis, Machine Learning and Applications*, pages 533–540. Springer, 2008.

[123] Roberto Martínez-Cruz, Alvaro J López-López, and José Portela. Chatgpt vs state-of-the-art models: A benchmarking study in keyphrase generation task. *arXiv preprint arXiv:2304.14177*, 2023.

[124] Luis Marujo, Wang Ling, Isabel Trancoso, Chris Dyer, Alan W Black, Anatole Gershman, David Martins de Matos, Joao P Neto, and Jaime G Carbonell. Automatic keyword extraction on twitter. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 637–643, 2015.

[125] Yutaka Matsuo and Mitsuru Ishizuka. Keyword extraction from a single document using word co-occurrence statistical information. *International Journal on Artificial Intelligence Tools*, 13(01):157–169, 2004.

[126] Ernest J. Mccormick. Human factors engineering. 1985.

[127] Olena Medelyan, Eibe Frank, and Ian H Witten. Human-competitive tagging using automatic keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3*, pages 1318–1327. Association for Computational Linguistics, 2009.

[128] Olena Medelyan, Eibe Frank, and Ian H Witten. Human-competitive tagging using automatic keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3*, pages 1318–1327. Association for Computational Linguistics, 2009.

[129] Olena Medelyan, Eibe Frank, and Ian H. Witten. Human-competitive tagging using automatic keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3 - Volume 3*, EMNLP '09, pages 1318–1327, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[130] Rui Meng, Ye Liu, Semih Yavuz, Divyansh Agarwal, Lifu Tu, Ning Yu, Jianguo Zhang, Meghana Bhat, and Yingbo Zhou. Augtriever: Unsupervised dense retrieval by scalable data augmentation. *arXiv preprint arXiv:2212.08841*, 2022.

[131] Rui Meng, Debanjan Mahata, and Florian Boudin. From fundamentals to recent advances: A tutorial on keyphrasification. In *European Conference on Information Retrieval*, pages 582–588, 2022.

[132] Rui Meng, Khushboo Thaker, Lei Zhang, Yue Dong, Xingdi Yuan, Tong Wang, and Daqing He. Bringing structure into summaries: a faceted summarization dataset for long scientific documents. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 1080–1089, 2021.

[133] Rui Meng, Tong Wang, Xingdi Yuan, Yingbo Zhou, and Daqing He. General-to-specific transfer labeling for domain adaptable keyphrase generation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1602–1618, 2023.

[134] Rui Meng, Xingdi Yuan, Tong Wang, Sanqiang Zhao, Adam Trischler, and Daqing He. An empirical study on neural keyphrase generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4985–5007, 2021.

[135] Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. Deep keyphrase generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 582–592. Association for Computational Linguistics, 2017.

[136] Zakariae Alami Merrouni, Bouchra Frikh, and Brahim Ouhbi. Automatic keyphrase extraction: a survey and trends. *Journal of Intelligent Information Systems*, pages 1–34, 2019.

[137] Yishu Miao and Phil Blunsom. Language as a latent variable: Discrete generative models for sentence compression. *arXiv preprint arXiv:1609.07317*, 2016.

[138] Rada Mihalcea and Paul Tarau. Textrank: Bringing order into text. In *EMNLP*, pages 404–411. ACL, 2004.

[139] Rada Mihalcea and Paul Tarau. Textrank: Bringing order into texts. Association for Computational Linguistics, 2004.

[140] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[141] Shotaro Misawa, Yasuhide Miura, Motoki Taniguchi, and Tomoko Ohkuma. Multiple keyphrase generation model with diversity. In *European Conference on Information Retrieval*, pages 869–876. Springer, 2019.

[142] Funan Mu, Zhenting Yu, LiFeng Wang, Yequan Wang, Qingyu Yin, Yibo Sun, Liqun Liu, Teng Ma, Jing Tang, and Xing Zhou. Keyphrase extraction with span-based feature representations. *arXiv preprint arXiv:2002.05407*, 2020.

[143] Subhabrata Mukherjee and Ahmed Awadallah. Uncertainty-aware self-training for few-shot text classification. *Advances in Neural Information Processing Systems*, 33:21199–21212, 2020.

[144] Zara Nasar, Syed Waqar Jaffry, and Muhammad Kamran Malik. Textual keyword extraction and summarization: State-of-the-art. *Information Processing & Management*, 56(6):102088, 2019.

[145] Ani Nenkova and Rebecca J Passonneau. Evaluating content selection in summarization: The pyramid method. In *Proceedings of the human language technology conference of the north american chapter of the association for computational linguistics: Hlt-naacl 2004*, pages 145–152, 2004.

[146] Thuy Dung Nguyen and Min-Yen Kan. Keyphrase extraction in scientific publications. In *International Conference on Asian Digital Libraries*, pages 317–326. Springer, 2007.

[147] National Library of Medicine. What are structured abstracts? `https://www.nlm.nih.gov/bsd/policy/structured_abstracts.html`.

[148] Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. Scaling neural machine translation. *arXiv preprint arXiv:1806.00187*, 2018.

[149] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

[150] Eirini Papagiannopoulou and Grigorios Tsoumakas. Local word vectors guide keyphrase extraction. *arXiv preprint arXiv:1710.07503*, 2017.

[151] Eirini Papagiannopoulou and Grigorios Tsoumakas. A review of keyphrase extraction. *WIREs Data Mining and Knowledge Discovery*, 10(2):e1339, 2020.

[152] Krutarth Patel and Cornelia Caragea. Exploring word embeddings in crf-based keyphrase extraction from research papers. In *Proceedings of the 10th International Conference on Knowledge Capture*, pages 37–44, 2019.

[153] Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. Ppdb 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 425–430, 2015.

[154] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[155] Maxime Peyrard. A simple theoretical model of importance for summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1059–1073, 2019.

[156] Thang Pham, Seunghyun Yoon, Trung Bui, and Anh Nguyen. Pic: A phrase-in-context dataset for phrase understanding and semantic search. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1–26, 2023.

[157] Jordi Pons, Oriol Nieto, Matthew Prockup, Erik Schmidt, Andreas Ehmann, and Xavier Serra. End-to-end learning for music audio tagging at scale. *arXiv preprint arXiv:1711.02520*, 2017.

[158] Emerald Publishing. How to write a structured abstract? `https://www.emeraldgrouppublishing.com/how-to/authoring-editing-reviewing/write-article-abstract#structured-abstract`.

[159] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf*, 2018.

[160] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *URL https://openai. com/blog/better-language-models*, 2019.

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

[162] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67, 2020.

[163] Jishnu Ray Chowdhury, Cornelia Caragea, and Doina Caragea. Keyphrase extraction from disaster-related tweets. In *The world wide web conference*, pages 1555–1566, 2019.

[164] Nils Reimers and Iryna Gurevych. 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)*, pages 3982–3992, 2019.

[165] Alexander M. Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 379–389, 2015.

[166] Dhruva Sahrawat, Debanjan Mahata, Mayank Kulkarni, Haimin Zhang, Rakesh Gosangi, Amanda Stent, Agniv Sharma, Yaman Kumar, Rajiv Ratn Shah, and Roger Zimmermann. Keyphrase extraction from scholarly articles as sequence labeling using contextualized embeddings. *arXiv preprint arXiv:1910.08840*, 2019.

[167] Dhruva Sahrawat, Debanjan Mahata, Haimin Zhang, Mayank Kulkarni, Agniv Sharma, Rakesh Gosangi, Amanda Stent, Yaman Kumar, Rajiv Ratn Shah, and Roger Zimmermann. Keyphrase extraction as sequence labeling using contextualized embeddings. *Advances in Information Retrieval*, 12036:328, 2020.

[168] Olivier Salaün, Frédéric Piedboeuf, Guillaume Le Berre, David Alfonso Hermelo, and Philippe Langlais. Europa: A legal multilingual keyphrase generation dataset. *arXiv preprint arXiv:2403.00252*, 2024.

[169] Arnav Saxena, Mudit Mangal, and Goonjan Jain. Keygames: A game theoretic approach to automatic keyphrase extraction. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2037–2048, 2020.

[170] Abigail See, Peter J Liu, and Christopher D Manning. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*, 2017.

[171] Abigail See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[172] Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16)*, 2016.

[173] Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. Minimum risk training for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1683–1692, Berlin, Germany, August 2016. Association for Computational Linguistics.

[174] Xianjie Shen, Yinghan Wang, Rui Meng, and Jingbo Shang. Unsupervised deep keyphrase generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 11303–11311, 2022.

[175] Börkur Sigurbjörnsson and Roelof Van Zwol. Flickr tag recommendation based on collective knowledge. In *Proceedings of the 17th international conference on World Wide Web*, pages 327–336, 2008.

[176] Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June Hsu, and Kuansan Wang. An overview of microsoft academic service (mas) and applications. In *Proceedings of the 24th international conference on world wide web*, pages 243–246, 2015.

[177] Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June Hsu, and Kuansan Wang. An overview of microsoft academic service (mas) and applications. In *Proceedings of the 24th international conference on world wide web*, pages 243–246, 2015.

[178] Min Song, Il-Yeol Song, and Xiaohua Hu. Kpspotter: a flexible information gain-based keyphrase extraction system. In *Proceedings of the 5th ACM international workshop on Web information and data management*, pages 50–53. ACM, 2003.

[179] Mingyang Song, Xuelian Geng, Songfang Yao, Shilong Lu, Yi Feng, and Liping Jing. Large language models as zero-shot keyphrase extractor: A preliminary empirical study. *arXiv preprint arXiv:2312.15156*, 2023.

[180] Mingyang Song, Haiyun Jiang, Shuming Shi, Songfang Yao, Shilong Lu, Yi Feng, Huafeng Liu, and Liping Jing. Is chatgpt a good keyphrase generator? a preliminary study. *arXiv preprint arXiv:2303.13001*, 2023.

[181] Yang Song, Lu Zhang, and C Lee Giles. Automatic tag recommendation algorithms for social recommender systems. *ACM Transactions on the Web (TWEB)*, 5(1):1–31, 2011.

[182] Yang Song, Ziming Zhuang, Huajing Li, Qiankun Zhao, Jia Li, Wang-Chien Lee, and C Lee Giles. Real-time automatic tag recommendation. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 515–522, 2008.

[183] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 2014.

[184] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021, 2020.

[185] Sandeep Subramanian, Tong Wang, Xingdi Yuan, and Adam Trischler. Neural models for key phrase detection and question generation. *CoRR*, abs/1706.04560, 2017.

[186] Si Sun, Zhenghao Liu, Chenyan Xiong, Zhiyuan Liu, and Jie Bao. Capturing global informativeness in open domain keyphrase extraction. In *Natural Language Processing and Chinese Computing: 10th CCF International Conference, NLPCC 2021, Qingdao, China, October 13–17, 2021, Proceedings, Part II 10*, pages 275–287. Springer, 2021.

[187] Si Sun, Zhenghao Liu, Chenyan Xiong, Zhiyuan Liu, and Jie Bao. Capturing global informativeness in open domain keyphrase extraction. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 275–287. Springer, 2021.

[188] Si Sun, Chenyan Xiong, Zhenghao Liu, Zhiyuan Liu, and Jie Bao. Joint keyphrase chunking and salience ranking with bert. *arXiv preprint arXiv:2004.13639*, 2020.

[189] Yi Sun, Hangping Qiu, Yu Zheng, Zhongwei Wang, and Chaoran Zhang. Sifrank: A new baseline for unsupervised keyphrase extraction based on pre-trained language model. *IEEE Access*, 8:10896–10906, 2020.

[190] Zhiqing Sun, Jian Tang, Pan Du, Zhi-Hong Deng, and Jian-Yun Nie. Divgraphpointer: A graph pointer network for extracting diverse keyphrases. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, page 755–764, New York, NY, USA, 2019. Association for Computing Machinery.

[191] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[192] Brian Tomasik, Phyo Thiha, and Douglas Turnbull. Tagging products using image classification. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 792–793, 2009.

[193] Takashi Tomokiyo and Matthew Hurst. A language model approach to keyphrase extraction. In *Proceedings of the ACL 2003 Workshop on Multiword Expressions: Analysis, Acquisition and Treatment - Volume 18*, MWE '03, pages 33–40, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[194] Takashi Tomokiyo and Matthew Hurst. A language model approach to keyphrase extraction. In *Proceedings of the ACL 2003 workshop on Multiword expressions: analysis, acquisition and treatment*, pages 33–40, 2003.

[195] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[196] Isaac Triguero, Salvador García, and Francisco Herrera. Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study. *Knowledge and Information systems*, 42(2):245–284, 2015.

[197] Peter D Turney. Domain and function: A dual-space model of semantic relations and compositions. *Journal of artificial intelligence research*, 44:533–585, 2012.

[198] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748, 2018.

[199] L.J.P. van der Maaten and G.E. Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[200] Hans Van Halteren and Simone Teufel. Examining the consensus between human summaries: initial experiments with factoid analysis. In *Proceedings of the HLT-NAACL 03 Text Summarization Workshop*, pages 57–64, 2003.

[201] Trisha Van Zandt and James T Townsend. Self-terminating versus exhaustive processes in rapid visual and memory search: An evaluative review. *Perception & Psychophysics*, 53(5):563–580, 1993.

[202] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[203] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets. In *International Conference on Learning Representations (ICLR)*, 2016.

[204] Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, pages 2773–2781, 2015.

[205] Xiaojun Wan and Jianguo Xiao. Single document keyphrase extraction using neighborhood knowledge. In *AAAI*, volume 8, pages 855–860, 2008.

[206] Xiaojun Wan and Jianguo Xiao. Single document keyphrase extraction using neighborhood knowledge. In Dieter Fox and Carla P. Gomes, editors, *AAAI*, pages 855–860. AAAI Press, 2008.

[207] Xiaojun Wan, Jianwu Yang, and Jianguo Xiao. Towards an iterative reinforcement approach for simultaneous document summarization and keyword extraction. In *Proceedings of the 45th annual meeting of the association of computational linguistics*, pages 552–559, 2007.

[208] Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Improving text embeddings with large language models. *arXiv preprint arXiv:2401.00368*, 2023.

[209] Minmei Wang, Bo Zhao, and Yihua Huang. *PTR: Phrase-Based Topical Ranking for Automatic Keyphrase Extraction in Scientific Publications*, pages 120–128. Springer International Publishing, Cham, 2016.

[210] Shufan Wang, Laure Thompson, and Mohit Iyyer. Phrase-bert: Improved phrase embeddings from bert with an application to corpus exploration. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10837–10851, 2021.

[211] Xuerui Wang, Andrew McCallum, and Xing Wei. Topical n-grams: Phrase and topic discovery, with an application to information retrieval. In *Seventh IEEE international conference on data mining (ICDM 2007)*, pages 697–702. IEEE, 2007.

[212] Yanan Wang, Qi Liu, Chuan Qin, Tong Xu, Yijun Wang, Enhong Chen, and Hui Xiong. Exploiting topic-based adversarial neural network for cross-domain keyphrase extraction. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 597–606. IEEE, 2018.

[213] Cong Wei, Yang Chen, Haonan Chen, Hexiang Hu, Ge Zhang, Jie Fu, Alan Ritter, and Wenhu Chen. Uniir: Training and benchmarking universal multimodal information retrievers. *arXiv preprint arXiv:2311.17136*, 2023.

[214] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

[215] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

[216] Ian H. Witten, Gordon W. Paynter, Eibe Frank, Carl Gutwin, and Craig G. Nevill-Manning. Kea: Practical automatic keyphrase extraction. In *Proceedings of the Fourth ACM Conference on Digital Libraries*, DL '99, pages 254–255, New York, NY, USA, 1999. ACM.

[217] Di Wu, Wasi Ahmad, and Kai-Wei Chang. Rethinking model selection and decoding for keyphrase generation with pre-trained sequence-to-sequence models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6642–6658, 2023.

[218] Di Wu, Wasi Ahmad, Sunipa Dev, and Kai-Wei Chang. Representation learning for resource-constrained keyphrase generation. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 700–716, 2022.

[219] Huanqin Wu, Wei Liu, Lei Li, Dan Nie, Tao Chen, Feng Zhang, and Di Wang. Unikeyphrase: A unified extraction and generation framework for keyphrase prediction. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 825–835, 2021.

[220] Binbin Xie, Xiangpeng Wei, Baosong Yang, Huan Lin, Jun Xie, Xiaoli Wang, Min Zhang, and Jinsong Su. Wr-one2set: Towards well-calibrated keyphrase generation. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 7283–7293. Association for Computational Linguistics, 2022.

[221] Lee Xiong, Chuan Hu, Chenyan Xiong, Daniel Campos, and Arnold Overwijk. Open domain web keyphrase extraction beyond language modeling. *arXiv preprint arXiv:1911.02671*, 2019.

[222] Wenhan Xiong, Jingfei Du, William Yang Wang, and Veselin Stoyanov. Pretrained encyclopedia: Weakly supervised knowledge-pretrained language model. In *International Conference on Learning Representations*, 2019.

[223] xxxx. Keyphrase generation for scientific document retrieval. In *Proceedings of the 58th annual meeting of the association of computational linguistics*, pages xxx–xxx, 2020.

[224] Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. Luke: Deep contextualized entity representations with entity-aware self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6442–6454, 2020.

[225] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd annual meeting of the association for computational linguistics*, pages 189–196, 1995.

[226] Hai Ye and Lu Wang. Semi-supervised learning for neural keyphrase generation. *CoRR*, abs/1808.06773, 2018.

[227] Jiacheng Ye, Tao Gui, Yichao Luo, Yige Xu, and Qi Zhang. One2set: Generating diverse keyphrases as a set. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4598–4608. Association for Computational Linguistics, 2021.

[228] Wen-tau Yih, Joshua Goodman, and Vitor R Carvalho. Finding advertising keywords on web pages. In *Proceedings of the 15th international conference on World Wide Web*, pages 213–222, 2006.

[229] Mo Yu and Mark Dredze. Learning composition models for phrase embeddings. *Transactions of the Association for Computational Linguistics*, 3:227–242, 2015.

[230] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *arXiv preprint arXiv:2001.06782*, 2020.

[231] Xingdi Yuan, Tong Wang, Rui Meng, Khushboo Thaker, Peter Brusilovsky, Daqing He, and Adam Trischler. One size does not fit all: Generating and evaluating variable number of keyphrases. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7961–7975, 2020.

[232] Wenyuan Zeng, Wenjie Luo, Sanja Fidler, and Raquel Urtasun. Efficient summarization with read-again and copy mechanism. *arXiv preprint arXiv:1611.03382*, 2016.

[233] Qi Zhang, Yang Wang, Yeyun Gong, and Xuan-Jing Huang. Keyphrase extraction using deep recurrent neural networks on twitter. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 836–845, 2016.

[234] Yingyi Zhang, Jing Li, Yan Song, and Chengzhi Zhang. Encoding conversation context for neural keyphrase extraction from microblog posts. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1676–1686, 2018.

[235] Yingyi Zhang and Chengzhi Zhang. Using human attention to extract keyphrase from microblog post. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5867–5872, 2019.

[236] Yingyi Zhang, Chengzhi Zhang, and Jing Li. Joint modeling of characters, words, and conversation contexts for microblog keyphrase extraction. *Journal of the Association for Information Science and Technology*, 2019.

[237] Yong Zhang, Yang Fang, and Xiao Weidong. Deep keyphrase generation with a convolutional sequence to sequence model. In *2017 4th International Conference on Systems and Informatics (ICSAI)*, pages 1477–1485. IEEE, 2017.

[238] Yongzheng Zhang, Nur Zincir-Heywood, and Evangelos Milios. World wide web site summarization. *Web Intelligence and Agent Systems: An International Journal*, 2(1):39–53, 2004.

[239] Yu Zhang, Mingxiang Tuo, Qingyu Yin, Le Qi, Xuxiang Wang, and Ting Liu. Keywords extraction with deep neural network model. *Neurocomputing*, 383:113–121, 2020.

[240] Jing Zhao and Yuxiang Zhang. Incorporating linguistic constraints into keyphrase generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5224–5233, 2019.

[241] Sanqiang Zhao, Rui Meng, Daqing He, Saptono Andi, and Parmanto Bambang. Integrating transformer and paraphrase rules for sentence simplification. *arXiv preprint arXiv:1810.11193*, 2018.