TÉCNICO LISBOA

Adapting Multilingual Sentence Transformers for Unsupervised Key-Phrase Extraction from Long Documents

Artur Rodrigues Abrantes Arede Guimarães

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Bruno Emanuel da Graça Martins
Prof. Mathieu Roche

Examination Committee

Chairperson: Prof. Pedro Miguel dos Santos Alves Madeira Adão
Supervisor: Prof. Bruno Emanuel da Graça Martins
Member of the Committee: Prof. João Miguel da Costa Magalhães

June 2022

Acknowledgments

First and foremost I would like to thank my supervisors, Professor Bruno Martins and Professor Mathieu Roche, for their constant guidance which built the foundation of this work, as without them this would have never come to fruition.

I am very grateful for the support that was given to this research, namely by the European Union's H2020 re-search and innovation program, under grant agreement No.874850 (MOOD), as well as by Fundação para a Ciência e Tecnologia (FCT), specifically through the INESC-ID multi-annual funding from the PIDDAC program with reference UIDB/50021/2020, and also through the project grants with references PTDC/CCI-CIF/32607/2017 (MIMU), DSAIPA/DS/0102/2019 (DEBAQI), and POCI/01/0145/FEDER/031460 (DARGMINTS).

Gostava de agradecer à minha família, em especial ao meus pais e à minha irmã, pelo vosso amor incondicional. Por me educarem livre de fazer as minhas escolhas e de seguir os meus interesses, procurando por a minha felicidade sempre à frente da vossa. Por me aturarem dia após dia numa fase em que muito do meu tempo foi alocado ao meu trabalho, constantemente pondo-vos em terceiro ou quarto plano, sem nunca me darem menos do que suporte absoluto e transversal em tudo o que fazia. Amo-vos, obrigado por tudo.

A toda a Comissão de Praxe da Licenciatura em Engenharia Informática e de Computadores, obrigado por serem minha casa. Por terem conseguido ver valor num miúdo convencido que pouco o tinha, que encontrou algo na sua vida que adorava com todos os pedaços do seu ser. Aos que vieram antes de mim, Xu, Schima, Cruzinha, Antunes, Jesus, Louro, Lord, Loureiro, Alcino, Reynolds, Dinamos, Estalas, TH, Alpha, Palhas, Bia, Manso, Nádia, Cici, Daniel, Bartissol, por tudo o que me ensinaram, por todas as dores de cabeça que vos dei e acima de tudo por não terem desistido de mim, obrigado.

Aos que vieram comigo, Leonardo, Evans, Guilherme, Alexandre, Xavi, Cat, Serenatas, Barradas, Teo, pelo vosso companheirismo que me motivou a nunca desistir, pelos sonhos que concretizamos e os que deixamos para os próximos, pelas memórias felizes que levo de vós. À melhor geração, a do barril, obrigado.

Aos que vieram depois de mim, os Arturias, Duarte, Tiago, Vasco, JM e Lamego, pelo prazer que foi ver-vos crescer, por ter de incansavelmente aturar as vossas invenções que sempre me fizeram rir, e por quão orgulhoso fico de todos vós, obrigado. Aos CaPdBG, Gündersen, Afonso, Sancha, Mia, Chinopa, Nascimento, obrigado pelo vosso espírito que já sustenta esta casa. Desejo-vos a maior das sorte para o vosso ano. Aos mesmo novinhos, Bernardo, JB, André, Maria, Micael, Rento, bem-vindos à casa. Obrigado por perpetuarem algo que me é tão querido, e espero que vos traga tanta ou mais felicidade do que me trouxe a mim.

A todo o Magno Conselho da Praxe do Instituto Superior Técnico, obrigado por me terem recebido de braços abertos no Conselho, por todo o respeito que me deram, e acima de tudo pela total confiança que depositaram em mim, acreditando nas minhas ideias e dando-me oportunidade de as concretizar. Foi a maior das honras poder ter dado o meu contributo para este projeto, e espero ter estado à altura do que esperavam de mim.

Aos mais velhos que tive oportunidade de contactar com, Xu, Francisco, Mariana, Ramos, Antunes, Reynolds, Diana, Santiago, obrigado pelos vossos exemplos, por me fazerem desde cedo perceber o valor que a Praxe do Técnico tinha e o esforço que era necessário para a defender.

Aos meus contemporâneos, Gabriel, Rodrigo, Carolina, Mariana, Mariana, Francisco, Gonçalo, Luis, Daniel, João, Henrique, Gaspar, Diogo, Catarina, Inês, Beatriz, obrigado pelos vossos conselhos, por tudo o que me ensinaram e pelo que conseguimos juntos realizar.

Aos que vi entrar, Alícia, Humberto, Clara, Bernardo, Bárbara, Sancha, Diogo, André, Albino, Leonor, Ryan, vocês são o futuro desta casa. Obrigado por me fazerem sair sabendo que isto está em boas mãos, orgulhoso de cada um de vós e ansioso para descobrir o quanto farão o Conselho evoluir.

Ao Henrique, estiveste sempre lá quando necessário dar-me na cabeça ou para ouvir as minhas intermináveis dúvidas, fazendo o esforço de educar alguém que encontrou tarde o seu caminho. Ao mesmo tempo soubeste partilhar comigo o espírito da casa, ajudando a que o conseguisse passar para as gerações seguintes. Pela tua ajuda e pela tua amizade, obrigado.

Um agradecimento a ti Cristiana, por ter a oportunidade de trabalhar e aprender contigo desdo primeira dia, sabendo que tinha a tua confiança sempre em mim. Obrigado por me teres passado os teus sonhos e as tuas vontades, por ter completado este mandato que sempre será também teu. Foi um prazer ser teu secretário, e é com muito orgulho que vejo o nosso trabalho dar frutos.

À malta do Lab 16, Bisky, Dani, Leo, Custódio e Aylton, obrigado pela companhia diária numa altura difícil. O tempo que passamos a conviver foi consistentemente a highlight dos meus dias, e ajudaram-me imenso a fazer-me voltar a sentir-me eu mesmo.

Obrigado a ti Nádia, por tudo o que me ensinaste e tudo o que sacrificaste para me ajudar. És dos meus maiores exemplos de vida, e tenho a certeza que sem ti nunca teria conseguido acabar esta Tese.

Sancha e Mia, minha dupla da Mónica, obrigado por serem quem são, por me alegrarem sempre que estamos juntos e por todo o vosso afeto. Adorei fazer parte do vosso percurso, e não podia estar mais feliz com o que vocês cresceram ao longo deste anos. Até ao nosso próximo café.

Evans e Teo, minhas presis, agrupo-vos pois não conseguia de outra forma. Obrigado por tudo o que fizeram por mim, por o quanto me ajudaram a ser quem sou. Quando dizem que o Técnico não se faz sozinho são sempre vocês em quem penso, eram sempre vocês que estavam lá para mim. Adoro-vos incondicionalmente.

Sofia, por todos os sacrifícios a que foste sujeita e nada te trouxeram, por todos os sacríficios que fizeste que tanto me deram e me susteram, pelo carinho e amor que me mostras dia após dia, por quem és e pela grande sorte que tenho em te ter na minha vida, obrigado do fundo do meu coração. Amo-te muito.

Sempre acreditei que nada mais somos que pedaços daqueles de que gostamos, dos que prezamos e que peça a peça fazem a nossa pessoa. Obrigado a todos vós por fazerem a minha, levo-vos comigo no coração para a vida.

Abstract

Key-phrase extraction is commonly framed as the task of retrieving a small set of phrases that encapsulate the core concepts of an input text, usually a single document. State-of-the-art supervised systems for key-phrase extraction require large amounts of labelled data and generalize poorly outside the training domain, while unsupervised approaches generally present a lower accuracy. This work starts by presenting a new multilingual unsupervised approach to single document key-phrase extraction, improving upon previous methods in several ways (e.g., using representations from pre-trained Transformer models, while supporting the processing of long documents). Nothing that few studies have addressed multi-document key-phrase extraction, despite its utility for describing and summarizing sets of documents relating to a given topic, this work also advances a multi-document approach. In this multi-document context, key-phrase re-ranking approaches were adapted from the developed single document ones, expanding them to the multi-document task. Experimental results, on both new and pre-existing datasets, covering multiple languages and domains attest to the quality of the extracted key-phrases.

Keywords

Key-phrase extraction, Multi-document key-phrase extraction, Multilingual text processing, Transformers

Resumo

A extração de palavras chave é uma tarefa que consiste em recolher um pequeno conjunto de frases que contenha os conceitos chaves de um determinado texto, normalmente um único documento. Os sistemas supervisionados estado da arte para este problema necessitam de grandes quantidades de data anotada e têm fraca generalização para fora do domínio onde foram treinados, enquanto os sistemas não supervisionados têm na sua grande maioria resultados piores. Este trabalho começa por apresentar novas abordagens multilingues para a extração de palavras chave num único documento, melhorando resultados anteriores utilizando representações obtidas por Transformers pré treinados e suportando o processamento de documento de texto longos. Atentando ao facto de poucos estudos existirem referentes à extração de palavras chave num cenário multi-documento, embora seja uma tarefa muito valiosa para trabalhos de sumarização de tópicos, este trabalho também contém abordagens para o cenário multi-documento. Neste contexto, as abordagens para um documento único foram adaptadas utilizando operações de reclassificação, expandindo-as para o cenário multi-documento. Os resultados experimentais obtidos, em datasets novos e antigos, em várias línguas e em diferentes domínios confirmam a qualidade das palavras chave extraídas por estas novas abordagens.

Palavras Chave

Extração de palavras chave, Extração de palavras chave multi-documento, Processamento de texto multilíngue, Transformers

Contents

List of Figures

List of Tables

Acronyms

BERT	Bidirectional Encoder Representations from Transformers
CNN	Convolutional Neural Network
DeBERTa	Decoding-enhanced Bidirectional Encoder Representations from Transformers with disentangled attention
Electra	Efficiently Learning and Encoder that Classifies Token Replacements Accurately
IR	Information Retrieval
KPE	Key-Phrase Extraction
SDKPE	Single-Document Key-Phrase Extraction
MDKPE	Multi-Document Key-Phrase Extraction
MLM	Masked Language Modeling
MSL	Multilingual Sentence-Longformer
MMR	Maximal Marginal Relevance
MLP	Multi-Layer Perceptron
NLP	Natural Language Processing
NSP	Next Sentence Prediction
POS	Part-of-Speech
RoBERTa	Robustly Optimized Bidirectional Encoder Representations from Transformers
LMEmbedRank	Longformer Multilingual EmbedRank
LMMaskRank	Longformer Multilingual MaskRank
LMRank	Longformer Multilingual Rank
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
MAP	Mean Average Precision

nDCG Normalized Discounted Cumulative Gain

1

Introduction

Contents

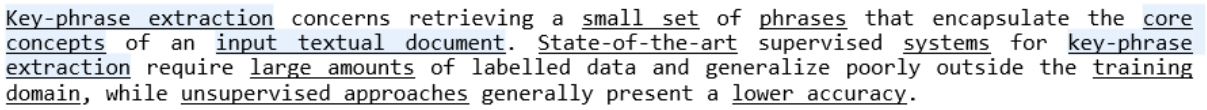

Figure 1.1: Key-phrase extraction example.

Key-Phrase Extraction (KPE) is the task of selecting important and topical phrases from within a body of text, typically a single input document. In Figure 1.1 we can see a small example of the task at hand, where the underlined words represent candidates phrases (i.e., sequences of words that might be key-phrases), and the highlighted ones represent the top ranked candidates (i.e., phrases that are retrieved as the key-phrases for the input text). Similarly to other text analysis and Information Retrieval (IR) tasks, recent methods involve the use of text representations produced through neural models.

Multi-Document Key-Phrase Extraction (MDKPE) can be seen as an extension of the simpler KPE task. The problem is framed as retrieving a small set of phrases that encapsulate the core concepts within a collection of documents discussing a common topic, often used to describe or summarize that collection. Despite the value of this task for gleaning high-level depictions on sets of related documents, its has only sporadically been researched (Shapira et al., 2021), with previous studies having, for instance, proposed to merge and re-rank the results from Single-Document Key-Phrase Extraction (SDKPE), e.g. preferring key-phrase candidates that appear on multiple documents associated to the topic.

1.1 Motivation

Supervised approaches for KPE require quantity and quality of in-domain annotated data, motivating work on transfer learning (Xiong et al., 2019), weakly-supervised (Wang et al., 2020), or unsupervised (Bennani-Smires et al., 2018a; Ding and Luo, 2021; Sun et al., 2020; Meng et al., 2017; Zhang et al., 2021) methods, that do not require the usage of expensive annotations nor extensive training procedures to obtain strong results. Although recent approaches have shown promising results, they focused mostly on the English language and on methods suited to handle short documents. As experiences conducted on multilingual domains have been limited, there remain questions about how well these approaches can be generalized to different languages or application domains, without factoring into account domain specific characteristics.

As for MDKPE, the lack of research concerning this task leaves a great margin for improvement, specifically taking advantage of Transformer language models. Re-ranking operations have shown good results in this task (Shapira et al., 2021), allowing KPE approaches to be adapted to it.

1.2 Thesis Proposal

This thesis explores KPE in an unsupervised multilingual scenario, specifically by adapting and reconfiguring pre-existing methods (i.e., EmbedRank (Bennani-Smires et al., 2018a) and MDERank (Zhang et al., 2021)) to work with representations produced with a pre-trained model from the Sentence-Transformers (Reimers and Gurevych, 2019) library, whilst supporting the processing of long text documents by converting the Sentence Transformers model into a Long Document Transformer (Longformer) (Beltagy et al., 2020). The proposed KPE methods were evaluated on different domains (i.e., involving texts of different sizes, types, and languages), and results show they offer good generalization and improvements over previous approaches.

As for the MDKPE scenario, the proposed KPE methods were adapted to a multi-document scenario, in order to test their effectiveness and prove if a simple re-ranking approach is good enough to obtain competitive results.

1.3 Contributions

The main scientific contributions this thesis offers are:

- The development of unsupervised approaches to KPE that support the processing of long documents, whilst remaining multilingual, with their corresponding source code publicly available online [1]. Extensive evaluation was performed in several datasets which encompass different domain characteristics, languages and varying document lengths. Competitive results with previous unsupervised methods were achieved, even surpassing past results on some datasets and metrics. The work on the KPE task was compiled into a paper that was submitted to a conference on computational linguistics, currently pending revision.

- The creation of a dataset comprised of scientific articles from ResistanceBank.org [2], an open access repository for surveys and maps of antimicrobial resistance in animals, suited as a resource for evaluating KPE from scientific publications in the specific domain of epidemiology, also publicly available online [3]. Baseline results for the usage of this dataset were obtained, with candidate extraction rates very close to finding all explicit candidate mentions.

- The development of multilingual unsupervised approaches for MDKPE that adapt the KPE approaches, testing how a simple re-ranking operation can work in a multi-document scenario. Results surpassed state-of-the-art methods in some metrics, and remained similar in the rest.

[1] https://github.com/araag2/KP_Extraction
[2] https://resistancebank.org/
[3] https://github.com/araag2/ResistanceBank-Dataset

1.4 Organization of the Dissertation

The rest of this document is structured as follows: Chapter 2 discusses fundamental topics related to machine learning with neural networks, together with related work on the topic of KPE, divided between supervised and unsupervised methods, and briefly covering the state-of-the-art in the MDKPE task. Chapter 3 explains the multilingual unsupervised approaches for KPE that were developed, as well as their different configurations, weighting schemes and extraction methodologies, alongside a thorough description of the experimental evaluations conducted. Chapter 4 concerns the multilingual unsupervised approaches for MDKPE, namely the way it was built upon an adaptation of the proposed KPE approaches, afterwards reporting experimental evaluation against the current baselines. At last, Chapter 5 delineates a discussion about this dissertation, the main conclusions that were obtained, and what future work could be done.

2

Concepts and Related Work

Contents

2.1 Fundamental Concepts

This section discusses the fundamental concepts required to understand the remaining techniques presented in this Thesis. Specifically, topic such as the basis of what neural networks are, how they can be used in machine learning, and which techniques can train and optimize them, will be covered. Lastly a full description of the Transformer architecture and the Bidirectional Encoder Representations from Transformers (BERT) model will be done, which are two neural network models commonly used in text processing tasks.

2.1.1 Machine Learning with Neural Networks

In very brief terms, neural networks can be broadly defined as a class of supervised machine learning algorithms (Goldberg, 2017), having their name attributed due to the resemblance to biological brains. In these networks, neurons are computational units that have scalar inputs and outputs, having weighted connections between them to pass information, akin to how biological synapses work. When several input connections are present, a neuron sums all scalar values, applies an internal (activation) function to the result, and passes that value to all its output connections.

The groundwork that brought this model to fruition was first presented in 1943 (McCulloch and Pitts, 1943) when a computational model for neural networks was first created. Still, only in 1957, with the introduction of the Perceptron (Rosenblatt, 1957), did the model come close to practical applications. The perceptron can be seen as the simplest form of a neural network (Goldberg, 2017), with just one node. It corresponds to a linear model that, for a given input $\boldsymbol{x}$, with a weight vector $\boldsymbol{w}$ and a bias term b_0, can be described as:

$$\mathrm{f}(\boldsymbol{x}) = \boldsymbol{x} \cdot \boldsymbol{w} + b_0. \tag{2.1}$$

In order to approximate non-linear decision functions, we need to add hidden layers to this simple model, combining multiple perceptrons. When doing so, we create a Multi-Layer Perceptron (MLP). As an example, let us consider 3 hidden layers in our model. For a given input $\boldsymbol{x}$, with $\boldsymbol{W_n}$ denoting the weight matrix and $\boldsymbol{b_n}$ the bias term of the linear transformation $\boldsymbol{n}$, we obtain:

$$\begin{aligned} \mathrm{h}_1(\boldsymbol{x}) &= \boldsymbol{x} \cdot \boldsymbol{W_1} + \boldsymbol{b_1}, \\ \mathrm{h}_2(\boldsymbol{x}) &= \mathrm{h}_1(\boldsymbol{x}) \cdot \boldsymbol{W_2} + \boldsymbol{b_2}, \\ \mathrm{h}_3(\boldsymbol{x}) &= \mathrm{h}_2(\boldsymbol{x}) \cdot \boldsymbol{W_3} + \boldsymbol{b_3}, \\ \mathrm{f}(\boldsymbol{x}) &= \mathrm{h}_3(\boldsymbol{x}) \cdot \boldsymbol{W_4} + \boldsymbol{b_4}. \end{aligned} \tag{2.2}$$

Since we want to use the perceptron or MLP models for supervised learning, we take as input a

training set of n examples $\mathfrak{x} = (\boldsymbol{x_1}, \boldsymbol{x_2}..., \boldsymbol{x_n})$ coupled with $\mathfrak{y} = (\boldsymbol{y_1}, \boldsymbol{y_2}, ..., \boldsymbol{y_n})$ corresponding labels. The goal of a learning algorithm is to return a function $\mathrm{f}(\boldsymbol{x})$, such as those shown in Equations 1 or 2 (i.e., discover the parameters of those functions) having as an objective that the predictions $\mathrm{f}(\boldsymbol{x}) = \widehat{\boldsymbol{y}}$ over our training set are as accurate as possible.

To precisely quantify the disparity between the predicted labels and the true labels, we introduce the notion of a loss function. In this context, we can see the loss as an arbitrary function that maps two vectors to a scalar value in the form $\mathrm{L}(\widehat{\boldsymbol{y}}, \boldsymbol{y})$, measuring the cost of predicting $\widehat{\boldsymbol{y}}$ when the true output is $\boldsymbol{y}$ and formally assigning the output to a numerical score. Considering the previous MLP example, we would set values for the matrices $\boldsymbol{W_n}$ and the $\boldsymbol{b_n}$ bias terms in order to minimize the loss value.

To define the loss in a formal manner, with Θ as our set of parameters, we can write the corpus-wide average loss $\mathcal{L}(\Theta)$ over our training examples as:

$$\mathcal{L}(\Theta) = \frac{1}{n} \sum_{i=1}^{n} \mathrm{L}(\mathrm{f}(\boldsymbol{x_i}; \Theta), \boldsymbol{y_i}). \tag{2.3}$$

In the same vein, with $\widehat{\Theta}$ as the parameter values that minimize the loss, we can denote the MLP training objective as:

$$\widehat{\Theta} = \arg\min_{\Theta} \mathcal{L}(\Theta). \tag{2.4}$$

For many types of models, this method of blindly minimizing the loss comes with a severe risk of overfitting attached, which would in turn gravely hamper the model. As a way to counteract this effect, we often pose soft restrictions on the form of the solution we seek. These restrictions are information added to our function that serves as regularization to our model, with the intent to help generalize it even at a cost of training accuracy. We can define a regularization term as $\mathrm{R}(\Theta)$, and with λ as a fixed scalar we can rewrite $\widehat{\Theta}$ as:

$$\widehat{\Theta} = \arg\min_{\Theta} \Big(\mathcal{L}(\Theta) + \lambda \mathrm{R}(\Theta) \Big). \tag{2.5}$$

Having all these concepts carefully defined, we can now firmly state that we are in the presence of an optimization problem, which entails that learning algorithms will try to minimize this function by systematically using values from our input set and updating the parameters according to their results.

A very popular approach to solving optimization problems is based on gradient calculus, which in the case of neural networks is a direct application of the chain-rule of differentiation to solve consecutive transformations. To this effect, the backpropagation algorithm is used to efficiently calculate gradients with respect to the weights of the network at hand, propagating the gradient from the output layer to the hidden layers, applying the chain-rule back to front. The process of systematically updating the network weights by moving from the end to the beginning of a network is usually referred to as the backward

pass, as opposed to the forward pass which is used to calculate the network output based on our input data, and computing the loss associated with the output.

Applying the backpropagation principle, the gradient descent algorithm is one of the most common ways to optimize neural networks according to a given loss function. Gradient descent can be defined as a way to minimize an objective function $\mathcal{L}(\Theta)$ by updating the parameters in the opposite direction of the gradient of the objective function $\Delta_\Theta \mathcal{L}(\Theta)$ relative to the parameters (Ruder, 2016). In this context, we can also define η as a learning rate, which determines the size of each update step to the parameters.

We can distinguish three variants of gradient descent that differ in how much data is used to compute the objective function, striving to reach the best trade-off between accuracy and efficiency, in the form of how long each update takes. The simplest form of gradient descent is usually referred to as batch gradient descent, which computes the gradient of the cost function relative to the parameters Θ for the entire training dataset simultaneously in a single update.

$$\Theta_{t+1} = \Theta_t - \eta \cdot \Delta_\Theta \mathcal{L}(\Theta_t). \tag{2.6}$$

In stark contrast to this batch approach, we can also compute the gradient for each training example individually, a process which we denominate as Stochastic Gradient Descent (SGD). With training examples $\boldsymbol{x_i}$ and labels $\boldsymbol{y_i}$, this can be expressed in the form:

$$\Theta_{t+1} = \Theta_t - \eta \cdot \Delta_\Theta \mathcal{L}(\Theta_t, \boldsymbol{x_i}, \boldsymbol{y_i}). \tag{2.7}$$

While in batch gradient descent we have guarantees of global minimum convergence in what regards to the parameters, in SGD the individual updates cause result fluctuation. This allows the values to move towards potentially better local minima, but complicates convergence to the minimum. In terms of efficiency, SGD is usually much faster as its update policy eliminates any redundancy of similar updates being computed multiple times. As a way to fuse these approaches, mini-batch gradient descent unites their advantages and performs single updates for each mini-batch of n training examples. With $\boldsymbol{x_{i:i+n}}$ as mini-batches and $\boldsymbol{y_{i:i+n}}$ as labels for each training example within a mini-batch n, we have:

$$\Theta_{t+1} = \Theta_t - \eta \cdot \Delta_\Theta \mathcal{L}(\Theta_t, \boldsymbol{x_{i:i+n}}, \boldsymbol{y_{i:i+n}}). \tag{2.8}$$

The mini-batch method still presents the SGD advantage of optimizing the computation of repeated training examples, whilst reducing variance in each update. Due to its inherent advantages over the other two, mini-batch gradient descent is a common variant of choice for training neural network models.

Albeit very advantageous, gradient descent still presents challenges that need to be addressed, namely the difficulty of choosing a proper learning rate, the lack of schedules for updating the learning rate, and the way it ubiquitously applies to all parameters, to name a few. In order to mitigate and, to

some degree, solve these challenges, several optimization algorithms and methods were developed.

Momentum (Qian, 1999) is a method that helps SGD accelerate towards relevant directions, allowing it to more easily escape local optima. It also diminishes the amount of oscillation in consecutive updates, decreasing the total number of updates needed until convergence. The way it does so is by changing the update rule and adding a fraction of each last update to the current one. With a scalar γ as the momentum term and v_t as update values in each time step, momentum can be defined as:

$$\begin{aligned} v_t &= \gamma v_{t-1} + \eta \cdot \Delta_\Theta \mathcal{L}(\Theta_{t-1}), \\ \Theta_t &= \Theta_{t-1} - v_t. \end{aligned} \tag{2.9}$$

With the notion of momentum explained, we can now discuss Adaptive moment estimation (Adam), as proposed by Kingman and Ba (2015). This is an algorithm that computes adaptive learning rates for each parameter, being very appropriate for problems with sparse gradients or large amounts of terms/parameters. With v_t as a decaying average of past squared gradients, m_t as an average of squared gradients, and with β_1 and β_2 as momentum terms (both close to 1), we can write estimates of the first moment (i.e., mean) and the second moment (i.e., the uncentered variance) as:

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1)\Delta_\Theta \mathcal{L}(\Theta_{t-1}), \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2)(\Delta_\Theta \mathcal{L}(\Theta_{t-1}))^2. \end{aligned} \tag{2.10}$$

As m_t and v_t are initialized as vectors of zeros, the vectors are biased towards zero. To counteract those biases, Adam computes corrections as:

$$\begin{aligned} \widehat{m}_t &= \frac{m_t}{1-{\beta_1}^t}, \\ \widehat{v}_t &= \frac{v_t}{1-{\beta_2}^t}. \end{aligned} \tag{2.11}$$

With the corrections defined and with a small scalar ϵ to prevent division by zero, the final Adam update rule is as follows:

$$\Theta_{t+1} = \Theta_t - \frac{\eta}{\sqrt{\widehat{v}_t} + \epsilon}\widehat{m}_t. \tag{2.12}$$

2.1.2 The Transformer Architecture

Various neural network architectures have been designed to handle specific tasks, going beyond MLPs in the sense of considering a particular structure in the inputs. For instance, Convolutional Neural Networks (CNNs) were designed to handle matrices or tensors (e.g., images) as input data, and Recurrent Neural Networks (RNNs) operate explicitly with sequential data, where temporal relations are in effect. Within the field of Natural Language Processing (NLP), the input data can be modelled as sequences of word representations, and hence RNNs are commonly used. That being said, in the context of this the-

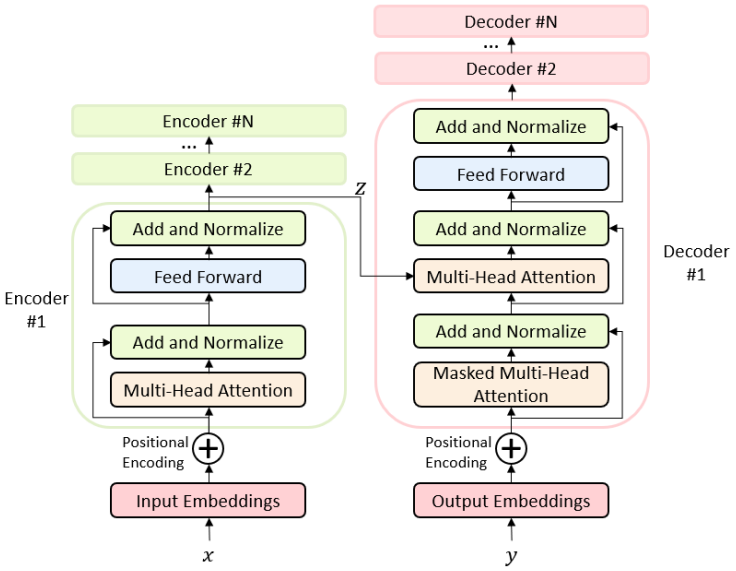

Figure 2.1: The Transformer neural network architecture.

sis a more recent architecture that produces state-of-the-art results in NLP problems will be analyzed, called the Transformer (Vaswani et al., 2017). Several models discussed over the course of this work will have the Transformer architecture as their basis, making it necessary to thoroughly explain it.

The main difference that separates the Transformer architecture from previous work (e.g., approaches based on RNNs) is its complete reliance on a neural attention mechanism, which we will cover in detail. The Transformer corresponds to an encoder-decoder structure, designed to encode a representation from a given input (e.g., a small piece of text) and decode an output sequence (e.g., another sentence) conditioned on the input. As a very general overview on how this encoder-decoder structure works, for a given set of input symbol representations $\mathfrak{x} = (\boldsymbol{x_1}, \boldsymbol{x_2}, ..., \boldsymbol{x_n})$ the encoder generates a sequence of continuous representations $\mathfrak{Z} = (\boldsymbol{z_1}, \boldsymbol{z_2}, ..., \boldsymbol{z_n})$, that the decoder uses to generate an output sequence $\mathfrak{y} = (\boldsymbol{y_1}, \boldsymbol{y_2}, ..., \boldsymbol{y_m})$, one element at a time. The model follows the aforementioned structure as show in Figure 1, with arbitrarily sized encoder and decoder stacks, relying on modules that employ a multi-head self-attention mechanism.

Before the input is processed by the encoder module, we change its representation using embeddings. Each symbol (e.g., a textual token) of our input will be transformed into a vector of size $d_{model} = 512$ in the original proposal, and this size remains fixed for all embeddings and encoders. This embedding process strives to build a vector representation of each token in order to preserve its contextual similarity while abstracting its existence into a low dimensional continuous space, allowing the model to standardize the way it processes sentences.

In order to make use of the sequential order of our input, since the model contains no recurrence nor convolution operations, it is necessary to add additional information to our embeddings about the relative

and absolute position of each token. To this effect, we add positional encodings to the input/output embeddings. These encodings have the same dimensionality d_{model} as the embedding vectors, and therefore in order to add them the vectors are simply summed. Several choices of positional encodings exist but, to easily learn to attend by relative positions, sine and cosine functions of different frequencies were chosen. With pos as the position and i as dimension, these functions are expressed as:

$$\begin{aligned} \mathrm{PE}\left(pos, 2i\right) &= \sin\Big(\frac{pos}{10000^{2i/d_{model}}}\Big), \\ \mathrm{PE}\left(pos, 2i+1\right) &= \cos\Big(\frac{pos}{10000^{2i/d_{model}}}\Big). \end{aligned} \tag{2.13}$$

A Transformer encoder module is composed by two sub-layers, each focused on a specific aspect of our token embeddings. The first one is a multi-head self-attention mechanism that focuses on how surrounding positions affect the current one, whilst the second one is a simple position-wise fully connected feed-forward network. As show in Figure 1, a residual connection is employed around each sub-layer and we proceed to normalize each layer's result, in the form $\mathrm{LayerNorm}\left(\boldsymbol{x} + \mathrm{SubLayer}\left(\boldsymbol{x}\right)\right)$, where $\mathrm{SubLayer}\left(\boldsymbol{x}\right)$ is the implemented sub-layer function.

A Transformer decoder module is very similar to an encoder one, with the addition of a third sub-layer, which performs masked multi-head self-attention. The term masked refers to how the predictions for position i can only depend on the outputs at positions before i, therefore imposing a sequential (i.e., auto-regressive) relationship on the decoding process.

After seeing the core architecture, we can focus on the main innovation that makes the Transformer truly remarkable, namely the attention mechanism. In a general sense, attention within the Transformer can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values and output are all vectors. Self-attention is used in this model as a way to provide an understanding of context, producing a method for other relevant positions (i.e., other tokens in the input) to influence the one we are currently processing.

The particular attention formulation that is used in the Transformer architecture is called scaled dot-product attention. In order to calculate it, for each word embedding we will create a query vector, a key vector, and a value vector, which represent useful abstractions for our problem. We obtain each of these vectors by multiplying a given token embedding by specific weight matrices the model learns during the training process. The size of both the query and the key vectors is d_k = 64, while the size of the value vectors is d_v = 64. To express the transformation in matrix form, we must pack the individual embeddings for each word into a single matrix $\boldsymbol{X} \in \mathbb{R}^{w \times d_{model}}$, where w is the number of individual embeddings. We define the learned weight matrices as $\boldsymbol{W^Q} \in \mathbb{R}^{d_{model} \times d_k}$, $\boldsymbol{W^K} \in \mathbb{R}^{d_{model} \times d_k}$, and $\boldsymbol{W^V} \in \mathbb{R}^{d_{model} \times d_v}$ for query, key and value weights, respectively. With these matrices defined, we obtain $\boldsymbol{Q} \in \mathbb{R}^{w \times d_k}$, $\boldsymbol{K} \in \mathbb{R}^{w \times d_k}$, and $\boldsymbol{V} \in \mathbb{R}^{w \times d_v}$ by calculating the dot product of $\boldsymbol{X}$ with each learned weight matrix for query, key and value vectors, respectively.

With these matrices defined, we can compute the scaled dot-product attention by multiplying $\boldsymbol{Q}$ by $\boldsymbol{K^T}$, scaling the result with the term $\sqrt{d_k}$, and applying a row-wise softmax activation function to the result, transforming the values into probabilities (commonly referred to as the attention weights). Finally, the softmax results are multiplied by $\boldsymbol{V}$. In equation form, this can be expressed as:

$$\text{Attention}\left(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}\right) = \text{softmax}\left(\frac{\boldsymbol{QK^T}}{\sqrt{d_k}}\right)\boldsymbol{V}. \tag{2.14}$$

The multi-head attention mechanism extends this concept to allow the model to jointly attend to information from different representation subspaces, employing h parallel attention layers, also known as heads. In order to do so, we calculate attention with different projected $\boldsymbol{Q}$, $\boldsymbol{K}$ and $\boldsymbol{V}$ values in each head. Afterwards we concatenate the results and project the concatenation onto the original dimensions. In equation form, it can be expressed as follows:

$$\begin{aligned} \text{MultiHead}\left(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}\right) &= \text{Concat}\left(\boldsymbol{head_1}, ..., \boldsymbol{head_h}\right)\boldsymbol{W^O}, \text{with} \\ \boldsymbol{head_i} &= \text{Attention}\left(\boldsymbol{QW_i^Q}, \boldsymbol{KW_i^K}, \boldsymbol{VW_i^V}\right). \end{aligned} \tag{2.15}$$

The aforementioned projections are expressed as the parameter matrices $\boldsymbol{W_i^Q} \in \mathbb{R}^{d_{model} \times d_k}$, $\boldsymbol{W_i^K} \in \mathbb{R}^{d_{model} \times d_k}$, $\boldsymbol{W_i^V} \in \mathbb{R}^{d_{model} \times d_v}$, and $\boldsymbol{W^O} \in \mathbb{R}^{hd_v \times d_{model}}$.

As a final aspect worth mentioning, recall that the lack of recurrence or convolution operations in the model leads to the need for a mechanism that informs about the relative or absolute position of each token in a sequence. With that aspect in mind, positional encodings are also added at the bottom of the decoder stack.

2.1.3 Bidirectional Encoder Representation from Transformers

The BERT model (Devlin et al., 2018) was designed to produce deep bidirectional representations from text, that can be explored for target tasks. BERT is a conceptually simple but empirically powerful model, based on the encoder stack from the transformer architecture, explained in the previous subsection. BERT has two main steps in its framework, namely pre-training and fine-tuning.

Firstly, it is important to mention how BERT handles input/output representations, as that is the basis of all further procedures. In this context, we will define sentence as an arbitrary span of contiguous text, and a sequence as an input token sequence. The input representation is composed by three different embeddings for each token: a token embedding, that is based on WordPiece tokenization and embeddings (Wu et al., 2016) with a vocabulary of 30,000 entries to distinguish between tokens; a segment embedding, that distinguishes whether a token belongs to sentence A or B within a given sequence; and finally a position embedding, which determines the global position of any given token within a sequence. To be able to unambiguously represent both a single sentence and a pair of sentences as a single se-

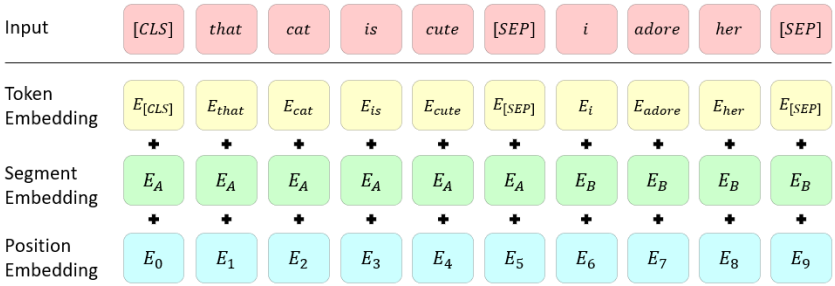

Figure 2.2: BERT Input Representation.

quence, BERT uses two special classification tokens: the [CLS] token, which appears exclusively at the beginning of every sequence, and the [SEP] token, which symbolizes the ending of a sentence and simultaneously separates sentences within a sequence, whilst also being the last token of each sequence. For any given token, its input representation is given by the sum of the token, segment (i.e., an indicator for each sentence within a sequence), and position embeddings, as shown in Figure 2.

For the pre-training part, two unsupervised language modelling tasks are used to replace traditional left-to-right/right-to-left language modeling tasks. The first of these tasks is Masked Language Modeling (MLM). It consists in masking some percentage of the input tokens at random and then trying to predict those masked tokens. This allows for bidirectional training of the model, as no word will indirectly see itself. To mitigate a possible downside of creating a mismatch between pre-training and fine-tuning, further randomness is introduced in the masking process. If the i-th token is chosen, 80% of the time it will be replaced with the [MASK] token, 10% of the time with a random token, and the final 10% of the time the i-th token will remain unchanged.

As for the second task, Next Sentence Prediction (NSP) is used to train the model to understand sentence relationships, using a binarized NSP problem that can be generated from any monolingual corpus. When given two sentences A and B for each example, 50% of time B is an actual sentence that follows A in the corpus, and the other 50% it is a random sentence from the corpus. Despite its simplicity, this NSP task is very important to many downstream tasks.

When approaching the fine-tuning aspect of this framework, we realize one of BERTs biggest advantages: the same model can apply to multiple downstream tasks involving single texts or text pairs, just by swapping out the appropriate input and output (i.e., we merely plug in the task specific inputs/outputs, and fine tune all parameters end-to-end). As for the output level, the individual token representations can be fed into an output layer for token-level tasks, while the [CLS] representation can be fed into another output layer for classification.

After the BERT model was created, several authors proposed extensions and reconfigurations to it, that sought to improve its performance. As an example of one of these reconfigurations, Liu et al. (2019) designed the Robustly Optimized Bidirectional Encoder Representations from Transformers (RoBERTa)

approach. This approach was based on experimental findings suggesting that the original BERT model was severely undertrained, and that with modifications to the training procedure state-of-the-art results could be matched or achieved in several datasets.

The training procedure for RoBERTa differs from the original one used in BERT in 5 main aspects: (1) the authors greatly increased the training data; (2) performed dynamic masking, where each training token is masked in 10 different ways instead of the original static masking; (3) removed the NSP task, as the authors found this task to worsen the performance of the model and opted to use full-sentence prediction without NSP; (4) changed the large mini-batch sequences of tokens, going from the original 1M steps for 256 sequences, to 125k steps for 2k sequences; (5) changed the tokenization process, opting to use byte-pair encoding instead of word pieces.

Another recent approach that was designed as an extension to BERT is named as Efficiently Learning and Encoder that Classifies Token Replacements Accurately (Electra), proposed by Clark et al. (2020). This approach differs from the original BERT model by acknowledging that the MLM task requires large amounts of data to be effective, and as such it could be improved upon. The proposed replacement task is called replaced token detection, where instead of masking parts of the input they are substituted by plausible sample tokens created by a small generator network. Afterwards, instead of training the model for trying to decode the original identity of the masked tokens, the approach trains a discriminative model that, for each token, individually predicts whether it is the original token or a generated one. An advantage of this approach relates to the fact that it is being trained over all input tokens, providing a much larger training set for the same amount of input data.

A distinct novel approach, designated Decoding-enhanced Bidirectional Encoder Representations from Transformers with disentangled attention (DeBERTa), was proposed by He et al. (2020) as a model architecture improving upon BERT and RoBERTa. DeBERTa introduced two main innovations. The first one is a disentangled attention mechanism that instead of using single vectors for each word to calculate the attention weights, splits the vector into content and position vectors, afterwards computing the attention weights using disentangled matrices over each one separately. Secondly, they incorporated absolute positions in the MLM task by applying word position embeddings before passing the vectors through the softmax layer, allowing the model to account for this additional syntactic element. As a final noteworthy mention, the authors also presented a new virtual adversarial training method for fine tuning the model to downstream NLP tasks, which they empirically showed to be quite effective at improving model generalization.

2.2 Related Work

This section contains in-depth discussions about previous work on key phrase extraction, detailing studies that are considered relevant in the context of this dissertation. Extending on surveys such as that from Hasan and Ng (2014), this section focuses on more recent and novel approaches, which will be combined and extended in order to build the proposed approaches with the best possible results. The section is composed by three subsections: one discussing work based on supervised methods, another focusing on unsupervised approaches, and a third one exposing the current state of the MDKPE task.

2.2.1 Supervised Methods for Key-Phrase Extraction

In the context of machine learning, the essence of supervised learning is the creation of mechanisms that can look at labeled examples and produce generalizations (Goldberg, 2017) in the form of models that can be used to process other instances. The labeled examples are often very expensive to produce, since good annotations for most existing tasks is quite hard to do.

With a supervised approach in mind, Sun et al. (2020) proposed the BERT-JointKPE model for KPE, i.e. a multi-task model built upon BERT that jointly learns the chunking of self-contained phrases and an estimation of their salience in an entire document. The model training process minimizes a loss function that merges n-gram chunking and pairwise salience rankings, both using ground truth key-phrase labels.

The proposed model has two main components, namely a chunking network and a ranking network. Both use n-gram representations with the contextualized token embeddings from BERT. With document D as input, divided in sequences $\mathfrak{D} = (\boldsymbol{s_1}, \boldsymbol{s_2}, ..., \boldsymbol{s_n})$ that generate embeddings $\boldsymbol{w_i}$, JointKPE encodes the document to a sequence of vectors $\mathfrak{H} = (\boldsymbol{h_1}, \boldsymbol{h_2}, ..., \boldsymbol{h_n})$. We can state that $\mathfrak{H} = \mathrm{BERT}\,(D)$, with each vector $\boldsymbol{h_i}$ corresponding to the contextualization of each $\boldsymbol{w_i}$ embedding.

Using the contextualized representations, JointKPE composes them to n-gram representations using CNNs. The representation of the i-th k-gram ${c_i}^k$, corresponding to embeddings $\boldsymbol{w_{i:i+k-1}}$, is calculated in set form $\mathfrak{G}_i^k = ({c_1}^k, {c_2}^k, ..., {c_n}^k)$ using the following equation:

$$\mathfrak{G}_i^k = \mathrm{CNN}^k(\mathfrak{H}). \tag{2.16}$$

In the previous equation, CNN^k corresponds to a set of CNNs with window size k $(1 \leq k \leq 5)$, which are used to compose the transition from token embeddings to n-gram representations.

The input representation is complete after the aforementioned two steps. The chunking network is now used with the set of n-gram input representations to predict whether each specific n-gram corresponds to a key-phrase or not. The probability of each ${c_i}^k$ is calculated by applying a softmax function on the output of a linear layer. In equation form this corresponds to:

$$P(\mathfrak{G}_i^k) = \text{softmax}\left(\text{Linear}\left(\mathfrak{G}_i^k\right)\right), \tag{2.17}$$

yielding a binary classification output ${y_i}^k$ on each n-gram ${c_i}^k$, with ${y_i}^k$ = 1 stating that the n-gram is a key-phrase, and ${y_i}^k$ = 0 otherwise.

Finally, another linear layer is used on the n-gram representations $\mathfrak{G}_i^k$ to generate salience scores for each n-gram ${c_i}^k$, according to:

$$\text{f}\left({c_i}^k, D\right) = \text{Linear}\left(\mathfrak{G}_i^k\right). \tag{2.18}$$

The function $\text{f}\left({c_i}^k, D\right)$ represents the score for a single n-gram in a document. As the same phrase may appear several times in a document, JointKPE gathers all the n-gram scores for the same phrase $\text{f}\left({c_i}^k, D\right)$ and maps them using a new functionf$^*\left(p^k, D\right)$ to score each phrase uniquely. For the phrase p^k with length k, JointKPE first gathers all k-grams in the document that are contained within the phrase, and then merges their scores using a max-pooling operation to obtain the final salience score for each phrase p^k. In equation form:

$$\text{f}^*\left(p^k, D\right) = \max\left(\text{f}\left({c_j}^k, D\right), ..., \text{f}\left({c_l}^k, D\right), ..., \text{f}\left({c_m}^k.D\right)\right). \tag{2.19}$$

BERT-JointKPE is trained with a multi-task loss that is composed by both chunking and ranking loss functions, in order to better achieve a balance between the quality and salience of the extracted phrases. This multi-task loss is a linear combination given by:

$$\mathcal{L} = \mathcal{L}_{Chunk} + \mathcal{L}_{Rank}, \tag{2.20}$$

where both the $\mathcal{L}_{Chunk}$ and $\mathcal{L}_{Rank}$ components are formulated using the ground truth key-phrase labels. For a given document D, $\widehat{\mathfrak{P}}$ = $(\widehat{p_1}, \widehat{p_2}, ..., \widehat{p_n})$ represents the set of ground truth key-phrase labels associated with it. To obtain the associated chunking labels, any n-gram that can match any key-phrase in $\widehat{\mathfrak{P}}$ is added to the positive set $\mathfrak{C}_+$, calculating the label associated with n-gram ${c_i}^k$ with the formula:

$${y_i}^{k*} = \begin{cases} 1 & {c_i}^k \in \mathfrak{C}_+ \\ 0 & {c_i}^k \notin \mathfrak{C}_+. \end{cases} \tag{2.21}$$

The chunking loss is calculated using the cross-entropy loss for matching with the chunking labels, according to the following equation:

$$\mathcal{L}_{Chunk} = \text{CrossEntropy}\left(\text{P}({c_i}^k = {y_i}^{k*})\right) = -\sum_i {c_i}^k \log {y_i}^{k*}. \tag{2.22}$$

As for the ranking labels, they are obtained by adding all the phrases that are labeled as key-phrases

to a positive set $\mathfrak{P}_+$, and the rest to a negative set $\mathfrak{P}_-$. The standard hinge loss for pairwise ranking is then used to calculate the ranking loss, in the form:

$$\mathcal{L}_{Rank} = \sum_{\mathfrak{P}_+, \mathfrak{P}_- \in D} \max\left(0, 1 - \mathrm{f}^*\left(\mathfrak{P}_+, D\right) + \mathrm{f}^*\left(\mathfrak{P}_-, D\right)\right). \tag{2.23}$$

The experimental procedure followed to test the BERT-JointKPE model used two datasets, namely OpenKP (Xiong et al., 2019) as a web KPE benchmark, and KP20K (Meng et al., 2017) as a scientific paper KPE benchmark, associated with the MS-MARCO leaderboard[1]. For evaluation metrics, Precision (P), Recall (R) and F_1-score (F_1) over the top N key-phrase predictions were used, with k = (1,3,5) for OpenKP and k = (5,10) for KP20K.

Eight different models were used as baselines for result comparison: the top two models at that time over the OpenKP leaderboard[2], namely BLING-KPE (Xiong et al., 2019) and LLbeBack; two state-of-the-art models on the KP20K dataset, namely CopyRNN (Meng et al., 2017) and DivGraphPointer (Sun et al., 2019); two implementations made by the authors, namely BERT-SpanKPE[3], which operates by first calculating the probability of a token being the start word for a key-phrase and then applying a self-attention layer to each subsequent word to calculate the probability of it being the last one, and BERT-TagKPE[4], which performs a soft-select method to decode phrases (e.g., given all 3-grams, only keeping the beginning of key-phrase score for the first word, middle of the key-phrase score for the middle one, and ending of the key-phrase score for the last word instead of keeping all scores for all words) and extracts multiple n-grams from the word-level representations, selecting them by applying different tag patterns (i.e., patterns based on the tag probabilities of each word given five different scores, O : non key-phrase; B : begin word of key-phrase; I : middle word of key-phrase; E : end word of key-phrase; U : uni-word key-phrase); finally two previously mentioned sub-networks of JointKPE, namely ChunkKPE and RankKPE, respectively for the chunking and ranking networks.

BERT-JointKPE improved all metrics by approximately 6% when compared to the best baselines, namely BLING-KPE and DivGraphPointer. Remarkably, the $F_{1,3}$-score on the OpenKP dataset was 28% higher than the previous state-of-the-art model. BERT-JointKPE also outperformed other methods using pretrained Transformer models. When compared to its individual chunking and ranking networks, the model performed more stably in all the metrics, demonstrating the positive effect of its multi-task learning. Finally, its effectiveness was further improved when using BERT variants such as SpanBERT (Joshi et al., 2020) or RoBERTa (Liu et al., 2019).

[1] https://github.com/spacemanidol/MSMARCO/tree/master
[2] https://github.com/microsoft/OpenKP/tree/master/LeaderboardResults
[3] https://github.com/thunlp/BERT-KPE#-bert-spankpe-see-code
[4] https://github.com/thunlp/BERT-KPE#-bert-tagkpe-see-code

2.2.2 Unsupervised Methods for Key-Phrase Extraction

In direct contrast with supervised learning, unsupervised learning operates by inferring patterns from unlabeled data. Within the realm of key phrase extraction, several types of approaches were devised to create unsupervised methods. We will cover the following: a graph-based approach, which builds a graph from the input model and uses a graph-based ranking method to calculate and rank the importance of each node; two embedding-based approaches, that use embeddings to map the words from the input document into a dense representation, later measuring the similarity between these representations against a representation from the document; a hybrid approach, which is able to produce both present and absent key-phrases for a document based on the corpus it is contained in.

2.2.2.A The Multipartite Graph Model

A recent example of a graph-based approach was created by Boudin (2018), using multipartite graphs to extract, score, and rank key-phrases in documents. The proposed model operates by first building a graph representation of the source document, with key-phrase candidates as nodes, and then applying a ranking algorithm that assigns a relevance score to each key-phrase. Following the setup used by Bougouin et al. (2013a), the author used the pattern `(/Adj*Noun+/)`, which represents a sequence of one or more consecutive nouns preceded by a sequence of zero or more adjectives, to identify key-phrase candidates, after associating part-of-speech tags to each word. All the key-phrase candidates go through a stemming process to ensure that similar words are condensed into the same representations. Afterwards, the candidates are grouped into topics by using hierarchical agglomerative clustering, i.e. a bottom-up clustering method where each candidate starts as their own topic, considering only the exact words that are shared by the candidates (i.e., the similarity is given by exact word matches), with the average linkage criteria as a clustering metric.

To encapsulate the document structure, a complete directed multipartite graph is built, using the candidate key-phrases as nodes. The nodes are only connected if they belong to different topics, as assigned by the clustering algorithm, and each edge is weighted according to the distance between two given candidates. Given two candidate key-phrases from different topics, c_i and c_j, and having their set of word offset positions as $\mathfrak{P}(c_i)$ and $\mathfrak{P}(c_j)$, respectively, the weight corresponding to the edge from node i to node j is defined as:

$$w_{ij} = \sum_{p_i \in \mathfrak{P}(c_i)} \sum_{p_j \in \mathfrak{P}(c_j)} \frac{1}{|p_i - p_j|}. \tag{2.24}$$

After this process, a complete k-partite graph is created, partitioned into k different independent sets, as shown in the example from Figure 3. As relying on the importance of a given candidate for selection of key-phrases is not sufficient (Hasan and Ng, 2014), the author decided to incorporate the position of

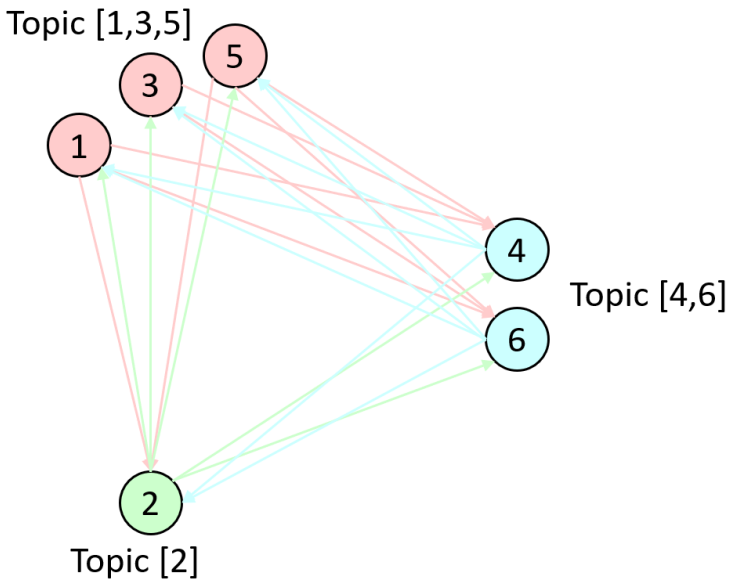

Figure 2.3: Multipartite graph representation.

the candidate within the document as an additional measure to address this issue. In order to do so, the edge weights are adjusted so that candidates occurring at the beginning of the document are promoted in relation to other candidates belonging to the same topic. With $\mathfrak{T}\, c_j$ as the set of nodes belonging to the same topic as c_j, with p_i as the offset position of the first occurrence of candidate c_i, and with α as an adjustable hyperparamenter to control the strength of the adjustment, the weight update rule is given by the following equation:

$$w_{ij,t+1} = w_{ij,t} + \alpha e^{\frac{1}{p_i}} \sum_{c_k \in \mathfrak{T}\,(c_j) \setminus \{c_j\}} w_{ki,t}. \tag{2.25}$$

For ranking the key-phrase candidates, the author chose to use an approach similar to that of the TextRank algorithm (Mihalcea and Tarau, 2004), taking into account edge weights. With $\mathfrak{I}\,(c_i)$ as the set of predecessors of c_i, with $\mathfrak{O}\,(c_j)$ as the set of successors of c_j, with n as the total number of nodes, and with λ as a damping factor, the algorithm can be expressed as follows:

$$S\,(c_i) = \frac{(1-\lambda)}{n} + \lambda \sum_{c_j \in \mathfrak{I}\,(c_i)} \frac{w_{ij}\, S\, c_j}{\sum_{c_k \in \mathfrak{O}\,(c_j)} w_{jk}}. \tag{2.26}$$

To evaluate the results, three datasets were chosen to run experiments on: the Inspec dataset (Hulth, 2003a), containing short documents from scientific journal abstracts; the Marujo-2013 Portuguese dataset (Marujo et al., 2013), consisting of annotated news articles distributed over 10 categories; and the SemEval-2010 dataset, which is composed by several scientific articles collected from the ACM digital library. For evaluation metrics, Mean Average Precision (MAP) and F_1-score at the top

$k = 5$ and $k = 10$ key-phrases were reported, after applying stemming to all key-phrases.

In order to establish a direct comparison with previous work, three baseline models were used: TopicRank (Bougouin et al., 2013a), which is a model very similar to the one based on multipartite graphs, and which served as a strong influence to its creation; Single Topical PageRank (Sterckx et al., 2015), which is an improved version of the work by Liu et al. (2010); and PositionRank (Florescu and Caragea, 2017), i.e. a model that leverages word position and frequency to improve ranking accuracy. In order to reduce over-generation errors (i.e., cases in which the model correctly assesses key-phrases based on an important word but simultaneously wrongfully predicts other key-phrases based on that same important word), the author normalized the candidate scores by their length, both for the Single Topical PageRank and PositionRank baselines (Boudin, 2015). All models use the same candidate selection heuristic, in order to achieve a fair comparison.

When compared to the aforementioned baselines, the proposed model achieves the best result in $F_{1,5}$-score, $F_{1,10}$-score, and MAP across all datasets, albeit having smaller improvements in the shorter datasets of Inspec and Marujo-2013 when compared to the larger SemEval-2010 dataset. The author also reported results without the weight adjustment operation for the proposed model, being mostly on-par with the baselines except for the $F_{1,5}$ scores, going from best to worst performer in two of the datasets. A working hypothesis for this is that the model struggles to select the most representative candidates from each topic using TextRank as an unique feature. Across the board the model extracts over 92% of the top-10 key-phrases for each topic, which reveals that the model has a wide coverage and promotes diversity without any hard constraints.

2.2.2.B EmbedRank

EmbedRank (Bennani-Smires et al., 2018a) was created as an unsupervised method to extract key-phrases from a document, leveraging the notion of phrase embeddings to achieve this purpose. This method computes semantic relatedness among text fragments by measuring similarity in the feature space provided by phrase embeddings. Breaking the EmbedRank method into steps, we can distinguish the following: (1) candidate phrase extraction based on parts-of-speech sequences, where only phrases with zero or more adjectives followed by one or more nouns are kept; (2) using sentence embeddings to represent both the candidate phrases and the whole document in the same vector space; (3) ranking the candidate phrases to select those that correspond to the output key-phrases.

Embeddings are used to capture semantic relatedness, by measuring distances between each vector representation within a shared vector space. This mechanism is used to rank the candidate phrases by comparing their embedding vectors to the embedding of the original document. The method starts by creating the document embedding, including a noise reduction procedure that only keeps relevant information, in this case adjectives and nouns. Then, embeddings for each candidate phrase are com-

puted separately. In order to create the embeddings and evaluate their impact on the final results, the authors made tests with pre-trained models following Doc2Vec (Lau and Baldwin, 2016), with $Z = Z_d =$ 300 dimensions, and Sent2Vec (Pagliardini et al., 2017), with $Z = Z_s =$ 700 dimensions. These two methods are respectively referred to as EmbedRank d2v and s2v.

The aforementioned embedding methods are useful, as they allow us to represent a sequence of words with arbitrary-length, thus being appropriate to embed both phrases and documents. Although both EmbedRank d2v and s2v generate vector representations in the same semantic space, d2v is much faster. This latter approach performs only a single linear pass through the text and computes the phrase/document embeddings by averaging the pre-computed representations of the components (i.e., words and n-grams), whilst d2v uses an embedding network to infer a vector for the whole document.

After document and candidate phrase embeddings are computed, the cosine similarity between them is used as a measure to select the top candidate phrases. With $\boldsymbol{c_i}$ as the embedding vector for candidate phrase i, and with $\boldsymbol{d}$ as the embedding vector for the document itself, the cosine similarity is defined as:

$$\cos_{\text{sim}}(\boldsymbol{c_i}, \boldsymbol{d}) = \frac{\boldsymbol{c_i} \cdot \boldsymbol{d}}{\|\boldsymbol{c_i}\| \|\boldsymbol{d}\|}. \tag{2.27}$$

After ranking candidates through this similarity measure, the top N candidates are returned, accounting only for how informative they are. A potential problem relates to the fact that this process fails to consider how redundant key-phrases with high ranks can be between themselves. In order to counteract the diversity problem, the authors proposed an extension named EmbedRank++, which leverages ideas from search result diversification. Specifically, the author used the Maximal Marginal Relevance (MMR) procedure (Carbonell and Goldstein, 1998), as this is a simple and effective solution to combine the concepts of relevance and diversity. With λ as the balance factor between relevance and diversity, $\mathfrak{C}$ as the set of candidate key-phrases, $\boldsymbol{c_j}$ as the embedding vector for candidate phrase j, and $\mathfrak{K}$ as the set of extracted key-phrases, MMR is defined as:

$$\text{MMR}(\mathfrak{C}, \boldsymbol{d}) = \underset{\boldsymbol{c_i} \in \mathfrak{C} \setminus \mathfrak{K}}{\arg\max} \left(\lambda \, \overline{\cos_{\text{sim}}}(\boldsymbol{c_i}, \boldsymbol{d}) - (1 - \lambda) \max_{\boldsymbol{c_j} \in \mathfrak{K}} \overline{\cos_{\text{sim}}}(\boldsymbol{c_i}, \boldsymbol{c_j}) \right). \tag{2.28}$$

In the previous equation, $\overline{\cos_{\text{sim}}}$ represents a normalized cosine similarity that ensures that when λ = 0.5 the key-phrase relevance and diversity have equal importance. With σ as the normalization factor, the normalized cosine similarity as is defined as:

$$\begin{aligned} \overline{\cos_{\text{sim}}}(\boldsymbol{c_i}, \boldsymbol{d}) &= 0.5 + \frac{\text{ncos}_{\text{sim}}(\boldsymbol{c_i}, \boldsymbol{doc}) - \text{ncos}_{\text{sim}}(\mathfrak{C}, \boldsymbol{doc})}{\sigma(\text{ncos}_{\text{sim}}(\mathfrak{C}, \boldsymbol{doc}))}, \text{with} \\ \text{ncos}_{\text{sim}}(\boldsymbol{c_i}, \boldsymbol{d}) &= \frac{\cos_{\text{sim}}(\boldsymbol{c_i}, \boldsymbol{doc}) - \min_{\boldsymbol{c_j} \in \mathfrak{C}} \cos_{\text{sim}}(\boldsymbol{c_j}, \boldsymbol{doc})}{\max_{\boldsymbol{c_j} \in \mathfrak{K}}(\cos_{\text{sim}}(\boldsymbol{c_j}, \boldsymbol{doc}))}. \end{aligned} \tag{2.29}$$

In order to test the EmbedRank and EmbedRank++ methods, three common datasets were used:

Inspec (Hulth, 2003a) dataset, containing short documents from scientific journal abstracts; DUC 2001 (Wan and Xiao, 2008), consisting of medium length newspaper articles from TREC-9; and NUS (Nguyen and Kan, 2007), composed of long documents from full scientific conference papers, between 4 and 12 pages. For evaluation metrics, precision, recall and macro F_1-scores were chosen and computed for the top 5, 10 and 15 candidate key-phrases.

As baselines, the authors compared EmbedRank s2v and d2v to five strong models: TextRank (Mihalcea and Tarau, 2004), SingleRank (Wan and Xiao, 2008), WordAttractionRank (Wang et al., 2014), TopicRank (Bougouin et al., 2013a), and the multipartite graph (Boudin, 2018) ranking approach that was presented in the previous subsection. For TextRank and SingleRank, the window sizes (which refer to the maximum distance between words for them to present a co-occurrence relation) were set to 2 and 10, respectively. The same part-of-speech tagged text was used for all methods and all text was stemmed with the Porter stemmer (Porter, 1980).

Noteworthy results were gathered from the experimental evaluation. In two of the three datasets (Inspec and DUC2001) EmbedRank outperformed all other competing methods in all three evaluation metrics. Despite this success, the authors observed that the multipartite graph model outperformed all others in the NUS dataset. They attributed this discrepancy in performance to the ability of the multipartite graph model to incorporate positional information about the candidate key-phrases. When testing EmbedRank++ with λ=0.5 in the MMR equation, the results showed a reduction on key-phrase redundancy, but at the cost of also slightly reducing the $F1$-scores.

2.2.2.C SIFRank

Another state-of-the-art model within the realm of unsupervised key-phrase extraction is the SIFRank approach, which was created by Sun et al. (2020) as an extension over EmbedRank. The authors used their own sentence embedding model, called SIF, in conjunction with the pre-trained language model ELMo (Peters et al., 2018), to compute phrase and document embeddings. They also proposed a new method, called document segmentation, to speed up the process of computing phrase embeddings. This was especially useful when dealing with longer documents.

The process used by SIFRank to extract key-phrases can be divided in 5 main steps: (1) a document is tokenized and each token is annotated using a Part-of-Speech (POS) tag; (2) noun phrases (NPs) are extracted from the tokenized sequence according to a particular pattern on POS tags, corresponding to the initial candidate key-phrases; (3) the initial token sequence is embedded using the ELMo pre-trained language model; (4) the sentence embedding model is used to transform the embeddings obtained in the previous step into NP (according to the extracted NPs) and document embeddings, respectively; (5) the top-n key-phrases are selected by calculating the cosine similarity between the document embedding and the various NP embeddings, choosing the most similar ones. A general scheme for this framework

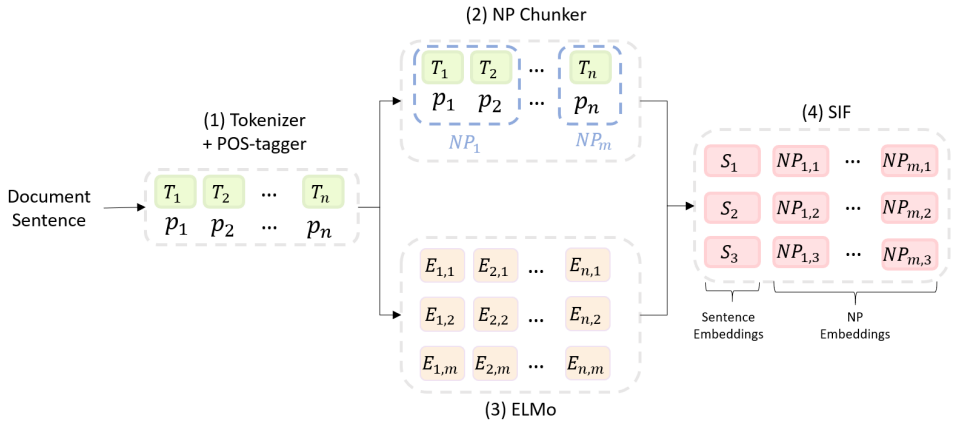

Figure 2.4: Overview on the SIFRank model.

can be seen in Figure 3.

In order to embed NPs and the document itself, the authors chose the sentence embedding model named SIF, due to its ability to accurately reflect the topic of the document and its compatibility with most pre-trained language models. As described in this model, for a given document d, the generation of a sentence s is a dynamic walk process where the k^{th} word w_k is generated at step k. The sentence embedding for a sentence s corresponds to the maximum likelihood estimation of the topic embedding that determines the whole document, and thus calculating the distance between candidate key-phrase embeddings and document embeddings is the same as calculating the similarity between the candidate key-phrase embeddings and the document topic. With c_d as a topic and with f_w as the statistical probability of a word appearing in a large corpus, the generation probability of the sentence s can be expressed through:

$$\Pr\left[s|c_d\right] = \prod_{w \in s} \Pr\left(w|c_d\right) = \prod_{w \in s} \left[\alpha f_w + (1-\alpha)\frac{\exp\left(< v_w, \tilde{c}_d >\right)}{Z_{\tilde{c}_d}}\right], \tag{2.30}$$

where $Z_{\tilde{c}_d} = \sum_{w \in V} \exp\left(< \tilde{c}_d >, v_w\right)$ with V as the set of all corpus words, $\tilde{c}_d = \beta c_0 + (1-\beta)c_d$ and $f_w = value_w / \sum_{j \in s} value_j$, with α and β as parameterizable hyper-parameters.

A sentence vector, corresponding to the maximum likelihood estimation of a topic, can then be defined as follows, with a as a hyper-parameter in the suitable range $a \in [10^{-3}, 10^{-4}]$:

$$v_s = \frac{1}{|s|}\sum_{w \in s} \text{Weight}\left(w\right) v_w = \frac{1}{|s|}\sum_{w \in s} \frac{a}{a + f_w} v_w. \tag{2.31}$$

The SIF embedding model is then combined with the auto-regressive pre-trained language model ELMo, in order to compute phrase and document embeddings. The word embeddings generated by ELMo have 3 layers, which within the context of this work are referenced to as L0, L1 and L2. The representation at each layer has a fixed size of 1024 dimensions. The first layer, L0, is a CNN layer that

generates static embeddings for each token, serving as the character encoding layer. As for the other two layers, i.e. L1 and L2, they produce contextual word embeddings using a biLSTM, as proposed in the original paper (Peters et al., 2018). These layers have designated strengths, as L1 is described as being better to capture grammatical information, and L2 as better to capture context-dependent semantic information. After inferring the performance of each layer through an experimental procedure, the authors chose to use the static layer L0 for documents longer than 128 words, while using the contextual embedding layer L1 on documents shorter than 128 words. The rationale is that as semantic word features could be better extracted from shorter text.

For a given candidate key-phrase NP, the SIFRank score is defined as its similarity or correlation score with a given document. With a document embedding v_d of document d, and the embedding of a candidate key-phrase v_{NP}, SIFRank is defined as:

$$\text{SIFRank}\left(v_{NP_i}, v_d\right) = \text{Sim}\left(v_{NP_i}, v_d\right), \tag{2.32}$$

with $\text{Sim}\left(v_{NP_i}, v_d\right)$ corresponding to the standard cosine similarity function, defined in Equation 2.27.

In order to adapt the SIFRank model to different domains, the authors also created a weight function for each word that is used when calculating the sentence embeddings. This function takes into account the current task domain corpus, but simultaneously uses a common corpus (e.g., Wikipedia) as a baseline. With a balance factor $\lambda \in [0, 1]$, the weight function $\text{Weight}(w)$ is given by:

$$\begin{aligned} \text{Weight}\left(w\right) &= \lambda\, \text{Weight}_{\,com}(w) + (1-\lambda)\, \text{Weight}_{\,dom}(w) \\ &= \lambda \frac{a}{a + f_w} + (1-\lambda)\frac{a^{'}}{a^{'} + f_w^{'}}. \end{aligned} \tag{2.33}$$

With the sentence embeddings fully detailed, a new notation has to be defined for each word embedding, as a different context or position changes the embedding of a word. Therefore, with w_i corresponding to the word at hand, s_j as the sentence that contains the word, and p the position of the word in the sentence, a word embedding is defined as $\text{v}_{\text{w}_\text{i}}\left(s_j, p\right)$.

Since calculating the embeddings for a whole document would take a long time, the authors proposed a document segmentation algorithm that would divide a document in several parts as one united batch. This allows the embedding process to be executed independently and in parallel over each part. A Minimum Sequence Length (MSL) was defined as the smallest size possible for each document segment to have, meaning each instance would have to have a word size equal or longer than the MSL. Albeit having several performance advantages, the segmentation of the document causes the model to lose context of the entire document when creating the embeddings, which can hinder its effectiveness. To counteract this effect, an embedding alignment method was used in conjunction with the segmentation process, in order to preserve contextual information of the entire document. This method uses the notion of an embedding anchor, which corresponds to the average of all contextual embeddings for a word,

running through each sentence and position it occurs in. After calculating all contextual embeddings from each individual document segment, the embedding vector $\bar{v}_{wi}$ replaces each word embedding in order to align all embeddings, and is calculated with the following expression:

$$\overline{\mathrm{v_{w_i}}} = \frac{1}{n} \sum_{s_j, p} \mathrm{v_{w_i}} \left(s_j, p\right). \tag{2.34}$$

Similarly to the previously covered example of EmbedRank++, the authors of SIFRank also devised SIFRank+, which takes into account positional information to change the weight of each word, and is, therefore, especially useful in long documents. The position-biased weight only considers the first occurrence of a word, mainly to avoid double counting the word frequency information already used in SIF. With p_1 as the first occurrence of a key-phrase relative to all other key-phrases NP_i, and with a hyper-parameter μ to optimize the weight, the position-biased weight is expressed as:

$$\mathrm{p}\left(NP_i\right) = \frac{1}{p_1 + \mu}. \tag{2.35}$$

In order to further narrow the gap of the position-bias for candidate key-phrases in close proximity, the softmax function is used to normalize the position-biased weight, in the form:

$$\tilde{\mathrm{p}}\left(NP_i\right) = \mathrm{softmax}\left(\mathrm{p}\left(NP_i\right)\right). \tag{2.36}$$

Finally, for long documents, the authors proposed the ranking method SIFRank+, which uses position-biased weight and is expressed as:

$$\mathrm{SIFRank+}\left(v_{NP_i}, v_d\right) = \tilde{\mathrm{p}}\left(NP_i\right) \mathrm{Sim}\left(v_{NP_i}, v_d\right). \tag{2.37}$$

In order to test the SIFRank and SIFRank+ methods, the authors used three public datasets. Two of them were the previously mentioned Inspec (Hulth, 2003a) and DUC 2001 (Wan and Xiao, 2008) datasets, while the third one was the SemEval2017 (Augenstein et al., 2017) dataset from Task 10 in the 2017 SemEval competition, which contains 493 selected paragraphs from the ScienceDirect journals accompanied with annotations. For evaluation metrics, precision, recall and macro F_1-scores were chosen and computed for the top 5, 10 and 15 candidates.

As baselines, the authors compared their model with 3 different types of methods: statistical models, namely TF-IDF and YAKE (Campos et al., 2018); graph-based models, namely TextRank (Mihalcea and Tarau, 2004), SingleRank (Wan and Xiao, 2008), TopicRank (Bougouin et al., 2013a), PositionRank (Florescu and Caragea, 2017) and the multipartite approach (Boudin, 2018); embedding-based models, namely RVA (Papagiannopoulou and Tsoumakas, 2018) and EmbedRank (Bennani-Smires et al.,

2018a). SIFRank and SIFRank+ both used the original ELMo model pretrained by AllenNLP[5], using the L0 layer when the documents are shorter than 128 words and the L1 layer otherwise. All the models used Standford CoreNLP[6] to perform tokenization and POS tagging, under the same environment.

In the experimental evaluation, SIFRank managed to outperform all other baseline methods in all metrics in two of the three datesets, namely Inspec and SemEval2017, achieving state-of-the-art results. On the long document dataset DUC2001, SIFRank+ took advantage of its position-biased weights to surpass all other methods in all metrics.

Overall, EmbedRank, EmbedRank++, SIFRank and SIFRank+ are simple and scalable methods for KPE from a single document, which are entirely unsupervised and corpus-independent. They all presented results that were superior or competitive to the vast majority of state-of-the-art unsupervised models, and thus these are methods that can serve as a strong foundation for other more advanced approaches based on sentence embeddings.

2.2.2.D MDERank

With the main advantages of EmbedRank (Bennani-Smires et al., 2018a) and SIFRank (Sun et al., 2020) described, a discussion about their shortcomings is crucial to further improve upon their results. These methods treat key-phrase extraction solely as a sequence tagging problem, following the pattern of firstly selecting candidates according to their POS tags and secondly ranking them using a scoring function. Albeit having a high retrieval speed and outstanding performance concerning predecessor methods, this procedure is vulnerable to sequence length mismatches between the scoring comparisons, as input sequences of different lengths are hard to align correctly in a semantic space. The resulting bias favours frequent and lengthy candidates, and these methods also frequently fail to encapsulate the context in which the candidates occur.

Having these aspects in consideration, Zhang et al. (2021) proposed a novel unsupervised method entitled MDERank, which is based on the intuition that the absence of a key-phrase would vary the semantic representation of a document. More precisely, instead of comparing a selected candidate to its source document directly, MDERank masks the candidate using a `[MASK]` token, comparing the similarity of the source document to the masked document. To then rank the candidates, the more similar a candidate is to the source document, the lower its ranking, as described by the intuition behind this method. As for embedding models, several were tried to guarantee the utility of MDERank, but the main one was the aforementioned BERT model.

Due to the implicit encoding of frequency and offset position information, the authors state that MDERank is better adapted to different domains and document lengths. Wanting to further emphasize its versatility, the authors also implemented four different masking techniques, which can be chosen

[5] https://allennlp.org/elmo
[6] https://stanfordnlp.github.io/CoreNLP/10901

according to the characteristics of a specific dataset. With d as the source document, d_m as the masked document, c_x as the occurrence x of a given candidate c, we can describe the various techniques as: (1) Mask all, where all occurrences of a candidate c are masked; (2) Mask first, where only the first occurrence of a candidate c is masked; (3) Mask highest, which only masks the occurrence of a candidate c that has the highest cosine similarity score to document d; and (4) Mask subset, which masks each occurrence of a candidate c that does not overlap with any other longer candidate, in order to allow the method to prefer longer candidates within nested candidates.

With regards to similarity measures, the authors propose using a standard cosine similarity function, as described in Equation 2.27. As alternative, the Euclidean distance was also tested. With n as the document length, the distance between two embeddings $\boldsymbol{e_1}$ and $\boldsymbol{e_2}$ is:

$$\mathrm{D}\left(\boldsymbol{e_1}, \boldsymbol{e_2}\right) = \sqrt{\sum_{i=1}^{n} (\boldsymbol{e_1}^i - \boldsymbol{e_2}^i)^2}. \tag{2.38}$$

As for the experimental evaluation of MDERank, six commonly used datasets were chosen: the previously mentioned DUC2001 (Wan and Xiao, 2008), Inspec (Hulth, 2003a), NUS (Nguyen and Kan, 2007), SemEval-2010 (Kim et al., 2010), and SemEval2017 (Augenstein et al., 2017) datasets, with the addition of the Krapivin (Krapivin et al., 2009) dataset, a large dataset consisting of 2k scientific papers from the computer science domain, published by ACM. The F_1-scores for the top 5, 10 and 15 candidates were chosen as evaluation measures, with stemming applied to reduce the mismatch frequency.

A toolkit for key-phrase extraction, namely PKE[7] was used to provide baseline unsupervised algorithms: TextRank (Mihalcea and Tarau, 2004), SingleRank (Wan and Xiao, 2008), TopicRank (Bougouin et al., 2013a), YAKE (Campos et al., 2018), MultipartiteRank (Boudin, 2018) and EmbedRank (Bennani-Smires et al., 2018a). Externally, SIFRank (Sun et al., 2020) was also used as a baseline, as at the time this was the state-of-the art method in key-phrase extraction. The Standford CoreNLP[8] part-of-speech tagger was used to extract candidates that follow the expression `<NN.*|JJ>*<NN.*>` for noun phrases.

The experimental evaluation unveils competitive results across all datasets with the best baseline available, namely SIFRank, achieving better results in the NUS, Krapivin, SemEval2010 and SemEval2017 datasets. MDERank therefore achieves convincing results on long documents datasets, which corroborates the hypothesis that matching sequence lengths is pivotal to compute textual similarities, a hypothesis which is further analyzed and confirmed in the paper with additional experimentation.

The overarching conclusion of this work is that MDERank improves upon state-of-the-art unsupervised key-phrase extraction methods, especially on long document datasets, whilst offering a more robust solution that counteracts biases on other methods such as EmbedRank and SIFRank. Moreover, with its different masking implementations, MDERank provides a new versatile baseline for key-phrase

[7] https://github.com/boundinfl/pke
[8] https://stanfordnlp.github.io/CoreNLP/

extraction, with its major downside being the additional temporal overhead of computing several different masked document embeddings, which is especially prominent in long documents.

2.2.2.E Unsupervised Deep Key-Phrase Generation

As a broader task in which key-phrase extraction is contained, key-phrase generation aims to produce a list of phrases that can accurately summarize and characterize a document, without necessarily needing to be present in that document. This trade-off between the reliance on within document key-phrase supervision and absent key-phrase generation has long existed, and whilst previously discussed methods were all extractive, in the sense of only focusing on the former, other deep neural network models (Meng et al., 2017; Yuan et al., 2018; Meng et al., 2020; Zhao et al., 2021) are capable of taking advantage of the latter to generate key-phrases. The main problem with most generative methods is the amount of annotated data they require to achieve state-of-the-art performance, as this data requires a large amount of resources to produce.

With the goal of alleviating the annotated data constrain on standard generative methods, but maintaining the extraction within document limitations in mind, Shen et al. (2021) proposed a novel hybrid unsupervised method capable of producing both present and absent key-phrases for a document, denominated AutoKeyGen. The authors observed that document key-phrases might be present in other documents within the same corpus or even within the same document, but existing as separate tokens, leveraging these observations as the foundation to the model they proposed.

The training process of the AutoKeyGen model can be divided in three main steps: (1) extracting candidate key-phrases from all documents and inferring relations between absent key-phrases; (2) ranking all candidate key-phrases based on TF-IDF information and embedding cosine similarity; (3) training a Seq2Seq generative model using the labels obtained from the previous step to generate more candidate phrases for each document.

Concerning the extraction of key-phrases from all documents in a given raw document collection, AutoKeyGen follows the same part-of-speech sequence pattern matching approach that was previously discussed for the EmbedRank model (Bennani-Smires et al., 2018b), creating a phrase bank containing all candidates in stemed form. Afterwards, given an input document, the model identifies a phrase as a candidate for that document only if the document contains every token that comprises that phrase, regardless of where it is in that document, testing every phrase in the phrase bank for every document.

For the ranking portion of this model, both lexical and semantic similarities were considered to be equally important and, to some degree, effective in key-phrase ranking. Using TF-IDF as a strong yet simple baseline (Meng et al., 2017; Campos et al., 2018) to rank the lexical similarity between a document $\boldsymbol{d}$ belonging to corpus $\mathfrak{D}$ and a candidate key-phrase c, with $\mathrm{TF}\left(c, \boldsymbol{d}\right)$ as the term frequency of c in $\boldsymbol{d}$, and with $\mathrm{DF}\left(c, \mathfrak{D}\right)$ as the document frequency of c in $\mathfrak{D}$, the lexical score is defined as:

$$\mathrm{Lex}\left(\boldsymbol{d}, c\right) = \frac{\mathrm{TF}\left(c, \boldsymbol{d}\right)}{|\boldsymbol{d}|} \log\left(\frac{\mathfrak{D}}{\mathrm{DF}\left(c, \mathfrak{D}\right)}\right). \tag{2.39}$$

The semantic similarity portion was measured based on previous work done by (Bennani-Smires et al., 2018b), as mentioned when presenting EmbedRank, using Doc2Vec (Lau and Baldwin, 2016) embeddings pre-trained on the large English Wikipedia corpus to generate embeddings for all the matching candidate phrases and the input document. Having candidates embedded within the same vector space, their semantic score is measured using, once again, the cosine similarity. With $\boldsymbol{e_c}$ corresponding to the candidate key-phrase embedding and $\boldsymbol{e_d}$ to the document embedding, and using the previously defined cosine similarity function, the semantic score is calculated as:

$$\mathrm{Sem}\left(\boldsymbol{d}, c\right) = \cos_{\mathrm{sim}}\left(\boldsymbol{e_d}, \boldsymbol{e_c}\right). \tag{2.40}$$

Having both the lexical and semantic similarity scores defined, the authors fused these two heuristic measures for key-phrase ranking using a geometric mean. The fused ranking score of a key-phrase c in relation to document $\boldsymbol{d}$ is written as:

$$\mathrm{Fused}\left(\boldsymbol{d}, c\right) = \sqrt{\mathrm{Lex}\left(\boldsymbol{d}, c\right) \mathrm{Sem}\left(\boldsymbol{d}, c\right)}. \tag{2.41}$$

For the final step in the model, the authors use the top-5 present and top-5 absent key-phrases to train a Seq2Seq generative model based on RNNs, in order to balance the importance of both types of key-phrases. The model follows a classical encoder-decoder structure, already discussed in Section 2, implementing the encoder section with a BiLSTM (Gers and Schmidhuber, 2001) and the decoder with a LSTM. The encoder maps the sequence of tokens contained within a document $\boldsymbol{d}$ to a sequence of hidden representations, in the form $(\boldsymbol{h_e^1}, \boldsymbol{h_e^2}, ..., \boldsymbol{h_e^{|d|}})$, while a RNN decoder generates a target key-phrase $\boldsymbol{y} = (y^1, y^2, ..., y^{|y|})$ token by token in an auto-regressive manner. With $\boldsymbol{h_e^t}$ and $\boldsymbol{h_d^t}$ as hidden states at time t for the encoder and the decoder, respectively, $\mathrm{f_e}$ and $\mathrm{f_d}$ as auto-regressive functions implemented by a LSTM, o^{t-1} as the predicted output at time $t-1$, and $\boldsymbol{cont}$ as the context vector derived from all the hidden states through a non-linear function $\mathrm{q}\left(\right)$, the update rules at each time-step are defined as:

$$\begin{aligned} \boldsymbol{h_e^t} &= \mathrm{f_e}\left(\boldsymbol{h_e^{t-1}}, |\boldsymbol{d}|\right), \\ \boldsymbol{cont} &= \mathrm{q}\left(\boldsymbol{h_e^1}, \boldsymbol{h_e^2}, ..., \boldsymbol{h_e^{|d|}}\right), \\ \boldsymbol{h_d^t} &= \mathrm{f_d}\left(\boldsymbol{h_d^{t-1}}, o^{t-1}, \boldsymbol{cont}\right). \end{aligned} \tag{2.42}$$

The prediction of y^t at time step t is based on a distribution over a fixed vocabulary, conditioned by the previously generated encoder source representations $\boldsymbol{h_e}$, and by the previously generated tokens $\boldsymbol{h_d}$. With $\mathrm{f}_{out}\left()\right.$ as a non-linear function that outputs the probabilities over all words in a preset vocabulary (i.e., a softmax classifier with an attention mechanism), the generation probability of y^t is defined as:

$$\mathrm{p_g}\left(y^t|y^{1,2,\ldots,t-1}, \boldsymbol{d}\right) = \mathrm{f}_{out}\left(y^{t-1}, \boldsymbol{h_d^t}, \boldsymbol{cont}\right). \tag{2.43}$$

The authors use guided beam search to generate diverse key-phrases for each document. Previous literature (Meng et al., 2017) has shown that similar Seq2Seq models can collapse and fail to generate high quality candidate key-phrases, and thus the authors proposed to encourage the decoder model to generate words that appear in the input document $\boldsymbol{d}$. Specifically, they doubled the generation probability of all words that occur in the input document.

In relation to the experimental evaluation of the AutoKeyGen model, the authors adopted five scientific publication datasets: the previously discussed Inspec (Hulth, 2003a), NUS (Nguyen and Kan, 2007), SemEval-2010 (Kim et al., 2010) and Krapivin (Krapivin et al., 2009) datasets; and lastly the KP20k (Meng et al., 2017) dataset, consisting of 528k scientific papers also from the computer science domain, gathered in various online digital libraries. For evaluation metrics the authors chose to use recall to evaluate the generated key-phrases, and the F_1-score to evaluate present key-phrases. For recall, the cut-off points R_{10} and R_{20} were chosen, whilst for the F_1-score they used the values $F_{1,5}$-score, $F_{1,10}$-score and $F_{1,O}$-score, where O corresponds to defining as cut-off point the same amount of key-phrases as the number of ground truth labels.

As baselines, the authors compared AutoKeyGen to five different unsupervised methods: TF-IDF (Jones, 1972), TextRank (Mihalcea and Tarau, 2004), SingleRank (Wan and Xiao, 2008), PositionRank (Florescu and Caragea, 2017) and the aforementioned EmbedRank method (Bennani-Smires et al., 2018b). To test ablations, they also introduced two variants of AutoKeyGen: AutoKeyGen-OnlyBank, which only uses partial matches between the phrase bank and the input document to extract candidates without using the Seq2Seq model; and AutoKeyGen-OnlyEmbed, which only uses the semantic similarity score to rank candidate key-phrases, forgoing the lexical similarity.

Noteworthy results were gathered both in the extraction and generation of key-phrases. For present key-phrases, AutoKeyGen achieved the best $F_{1,5}$-score, $F_{1,10}$-score and $F_{1,O}$-score values across all tested methods, outperforming even the baseline EmbedRank approach in all metrics by a significant margin. As for the absent key-phrases, AutoKeyGen also had the best results in all metrics, leading the authors to argue that the model has the potential to derive high-quality absent key-phrases under an unsupervised setting. When compared to the ablated variants, AutoKeyGen outperformed both of them, even though in some metrics not by a very significant margin.

2.2.3 Multi-Document Key-Phrase Extraction

MDKPE can be described as a very similar task to KPE, where the goal is to retrieve a small set of phrase that encapsulate the core concepts within a collection of documents that discuss a common topic. This

task has only been sporadically researched, despite ist value for gleaning high-level depictions on sets of related documents. Previous approaches have tried to merge and re-rank results from SDKPE (Berend and Farkas, 2013a; Bayatmakou et al., 2017a), or to find correlations between word occurrences in document sets (Hammouda et al., 2005a), both without great success.

Recently, Shapira et al. (2021) wrote a literature review on MDKPE, explaining previous work that was done and the main detractors from further pursuit. To their knowledge, Hammouda et al. (2005b) seem to be the first to explore this tasks, evaluating it in the web-document domain. Term frequency, term position and usage in titles or headlines were the factors analysed to perform key-phrase ranking from within documents of the same topic, evaluating 10 sets of 30 documents. For the key-phrase extraction segment, the word sequences that were common to all documents within a set were collected.

Berend and Farkas (2013b) proposed a re-ranking approach for the MDKPE task, merging key-phrase lists from each individual document from a given topic. Since a specific dataset for multi-document tasks did not exist, the authors used scientific papers from ACL workshops, paired with their respective call-for-papers website section. The re-ranking operation consisted in using information gain metrics on the unified list of key-phrases to extract the top-15 from a given topic set, and then applying word-level cosine similarity comparing each key-phrase with each call-for-papers website section. Further key-phrase analysis was also regulated by NLP experts, that assessed if the key-phrase lists did indeed correspond to the corresponding topics.

Another noteworthy contribution was the work by Bayatmakou et al. (2017b), which also approached the problem from a single to multi document approach. They created document topics by using search queries on a large dataset of scientific abstracts [9], and extracted key-phrases from them using RAKE (Rose et al., 2010). The key-phrases were then re-ranked to a unified list according to word co-occurrences within the key-phrase set and term frequency within the document topic, weighted by document salience in regards to the search query.

From all these works that were covered, a glaring limiting factor on research was the absence of a common MDKPE dataset, not allowing comparisons to be established between methods. Shapira et al. (2021) identified this problem as the main detractor from the pursuit of MDKPE. In order to improve this aspect, the authors proposed the first dataset for multi-document key-phrase extraction, titled MK-DUC-01, which builds upon the DUC-2001 dataset proposed by Wan and Xiao (2008) and is composed of 308 news articles from 30 different topics.

In order to adapt the DUC-2001 to a multi-document setting, the authors conducted an automatic merging and re-ranking process, following that with a manual refinement procedure. The end result produced a suitable benchmark for MDKPE, which they tested with ten different existing SDKPE algorithms and a singular MDKPE one, in order to provide base reference values for all future work. As a final note,

[9] https://www.webofknowledge.com

	DUC			NUS			Inspec			SemEval		
Model	$F1_5$	$F1_{10}$	$F1_{15}$	$F1_5$	$F1_{10}$	$F1_{15}$	$F1_5$	$F1_{10}$	$F1_{15}$	$F1_5$	$F1_{10}$	$F1_{15}$
Multipartite	21.70	24.10	23.62	6.17	8.57	10.82	13.41	18.18	20.52	10.13	12.91	13.24
EmbedRank	21.75	25.09	24.68	2.13	2.94	3.56	14.51	21.02	23.79	9.63	13.90	14.79
SIFRank	**24.30**	**27.60**	**27.96**	3.01	5.34	5.86	29.38	**39.12**	**39.82**	11.16	16.03	**18.42**
MDERank	13.05	17.31	19.13	15.24	18.33	**17.95**	26.17	33.81	36.17	10.16	15.32	17.76
AutoKeyGen	-	-	-	**21.80**	**23.30**	-	**30.30**	34.50	-	**18.70**	**24.00**	-

Table 2.1: Results of state-of-the-art unsupervised key-phrase extraction methods.

the authors mentioned that the concepts of MDKPE are already indirectly being used in multi-document summarization research, and the pursuit of better results can assist the summarization community.

2.3 Chapter Overview

Section 2.1 of this Chapter expands upon the foundational knowledge required to understand the exposed KPE methods, allowing Section 2.2 to focus on providing a broad overview on what the state-of-the-art methods for this task are, how they function and the influence they had on future work. A particular emphasis was deposited in unsupervised methods, as that is the main focus of this work, whilst briefly mentioning supervised methods and a similar yet novel task, MDKPE.

Table 2.1 gives an overview on how the different methods compare against each other, using results present on their original publications. With this comparison established, the remainder of this document pivots to a discussion about new methods and future work, using these results as a cornerstone.

3

Single-Document Key-Phrase Extraction

Contents

This chapter details the proposed approaches for SDKPE, also reviewing the previous models that were adapted and extended in this context. Section 3.1 portrays the architecture of the proposed approach, its multiple configurations and how previous work was influential to the ideas that were used. Section 3.2 describes all datasets and metrics used to evaluate the approaches, as well as associated results. Lastly, Section 3.3 provides a conclusion based on the obtained results, proposing future extensions to the developed approaches.

3.1 Overview on the Proposed Approaches

Even though the task definition of KPE was previously covered, it is important to accurately define it in mathematical terms: given a text document $\boldsymbol{d}$ belonging to a dataset $\mathfrak{D}$, the goal is to extract a set $\mathfrak{C}$ of candidate phrases $\boldsymbol{c}$ that correspond to the relevant key-phrases for that document $\boldsymbol{d}$. In order to do so, firstly a wide set $\mathfrak{C}$ is extracted and afterwards these candidate phrases are ranked, outputting the top-k candidates within the set as the chosen key-phrases.

Seeking to expand on previous studies covered in Chapter 2, namely EmbedRank (Bennani-Smires et al., 2018a) and MDERank (Zhang et al., 2021), the proposed approaches handle three different shortcomings of these studies: providing support for multilingual texts; performing comparisons between documents and candidate phrases in a computationally efficient manner; and, finally adequately handling the processing of large documents.

The candidate phrase extraction is done using models from the spaCy[1] library, according to the language of the documents within dataset $\mathfrak{D}$, to tokenize and perform part-of-speech tagging of each document $\boldsymbol{d}$. The regular expression `<PROPN|NOUN|ADJ>*<PROPN|NOUN>+<ADJ>*` over sequences of universal part-of-speech tags is used as a heuristic method to extract candidate phrases, relying only on a simple tagset that is common to different languages (Petrov et al., 2011). Lemmatization is performed to join candidates with slight differences into a single representation, through the simplemma [2] library which offers complete multilingual options. A mapping is kept between each possible candidate form and the corresponding lemmatized candidates, so that matches in the text can be aggregated into the lemmatized versions.

3.1.1 Obtaining Text Representations

Regarding the ranking task, suitable representations for both the source document and the candidate phrases need to be found, as even though Transformer encoder models like BERT (Devlin et al., 2018) and RoBERTa (Liu et al., 2019) can produce effective text representations, they are also too computa-

[1] https://spacy.io/models/
[2] https://github.com/adbar/simplemma/

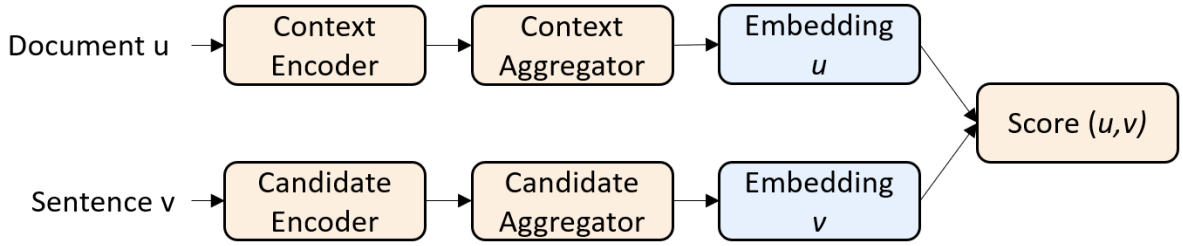

Figure 3.1: General bi-encoder architecture.

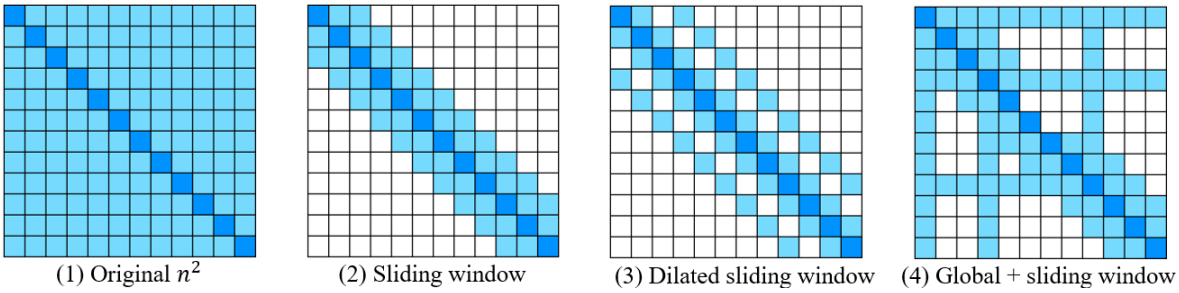

Figure 3.2: Self-attention in a standard Transformer model, versus attention patterns in a Longformer model.

tionally demanding. They can be used as cross-encoders to assess the similarity between a pair of input texts (i.e., processing the concatenation of both texts, and directly outputting a similarity score), but a more efficient approach is to instead consider a bi-encoder setting, in which the texts to be compared are modeled separately, and then a similarity score is computed over aggregates (e.g., token averages) from the resulting representations – see Figure 3.1.

Moreover, these models also struggle when processing long documents, due to the quadratic complexity associated to the self-attention mechanism computed over all pairs of positions from input sequences. Recent approaches such as the Longformer (Beltagy et al., 2020) or Bigbird (Zaheer et al., 2020) address this limitation, changing the self-attention operations in order to limit how the different positions interact – see the illustration on Figure 3.2.

Text representations are obtained with a multilingual model based on RoBERTa, adapted from a model in the Sentence-Transformers library [3] and pre-trained as a bi-encoder for assessing sentence similarity (Reimers and Gurevych, 2019). The RoBERTa-based model was adapted into a Longformer without any additional training, extending the input sequence limit to 4096 tokens (i.e., initializing the additional position embeddings by copying the first position embeddings) and changing the implementation of the self-attention operations within the different layers, while keeping the model parameters.

In brief, Sentence-Transformers bi-encoders process strings independently through the same Transformer encoder, followed by mean pooling aggregation to create fixed-sized sentence embeddings.

[3] https://www.sbert.net/

These models are trained either to directly predict sentence similarity scores as given in training data corresponding to annotated sentence pairs, or to predict similarity relations between sentences (e.g., given an anchor sentence a, a positive sentence p with high similarity towards a, and a negative sentence n, a loss function can be considered that tunes the network such that the distance between a and p is smaller than the distance between a and n). The process was specifically started from the Sentence-Transformers model named `paraphrase-multilingual-mpnet-base-v2`, built from a multilingual RoBERTa and trained to mimic the results of another monolingual Sentence-Transformers bi-encoder, through a knowledge distillation objective (Reimers and Gurevych, 2020). This model was adapted through the procedure described in the Longformer paper [4] to build a Long Document Transformer starting from a RoBERTa checkpoint.

As shown in Figure 3.2, three attention patterns are combined within the Longformer architecture: sliding window, focusing on the local context and examining a fixed-size window w around each token; dilated sliding window, which adds a gap of size d between each token considered in the sliding window, with d varying across layers and attention heads; and global attention, in which some specific input locations (e.g., the initial `[CLS]` token) will attend to (and be attended by) all other tokens.

In the remaining parts of this thesis, the proposed text representation model is referred to as the Multilingual Sentence-Longformer (MSL). This model employs a sliding window attention with size of 512, thus behaving like RoBERTa when the input has less than 512 tokens. One additional global attention pattern was also added, in which the specific positions corresponding to the occurrences of the key-phrase candidates have global attention, when representing both candidates and documents. The rationale behind the global attention addition is to emphasize the impact candidates have on the document representation, thus yielding greater importance between different candidate mentions.

Using these word representations, different approaches were built to address the candidate ranking problem for KPE.

3.1.2 LMEmbedRank

Longformer Multilingual EmbedRank (LMEmbedRank) corresponds to an adaptation of EmbedRank (Bennani-Smires et al., 2018a) that represents documents through MSL embeddings, and candidate phrases as the average of all MSL token embeddings that form the multiple occurrences of the candidate (first averaging the token representations from each occurrence, and then averaging across occurrences). As can be seen in Figure 3.3, where colored words represent the tokens that are considered, LMEmbedRank averages all the token representations in order to create the embedding representation of a document, whilst to embed a candidate phrase (e.g., *core concepts*) it performs an average pooling operation over all tokens that form each occurrence over the document, and then averages over

[4] https://github.com/allenai/longformer/blob/master/scripts/convert_model_to_long.ipynb

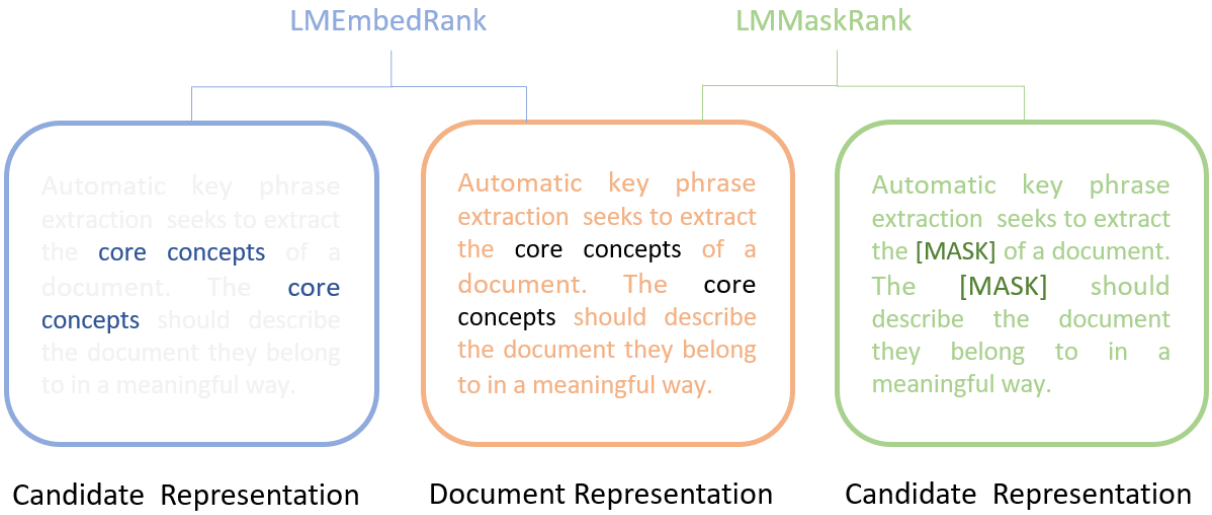

Figure 3.3: Overview on the LMEmbedRank and LMMaskRank representing the candidate *core concepts*.

all occurrences. For candidates that only occur after the first 4096 tokens (i.e., the Longformer limit), representations are generated using a back-off procedure that processes the candidate string alone.

After the candidate representations are generated, they are ranked using the cosine similarity measure to the document representation, in descending order of similarity.

3.1.3 LMMaskRank

Longformer Multilingual MaskRank (LMMaskRank) corresponds to an adaptation of MDERank (Zhang et al., 2021) that also represents documents and candidate phrases through MSL embeddings. As represented in Figure 3.3, where colored words represent considered tokens, in order to create the embedding representation of a document LMMaskRank uses the same mechanism as LMEmbedRank. On the other hand, to create a candidate phrase representation (e.g., *core concepts*), all occurrences of that specific candidate phrase are replaced by the `[MASK]` token, afterwards using MSL to embed the masked document.

As described in the original paper, candidate phrases are then ranked using the cosine distance measure, calculated by $1 - cosine\_similarity$, in descending order of distance (i.e., candidate representations that are further away from that of the original document are preferred).

It is interesting to note that the computationally most expensive operation, in both LMEmbedRank and LMMaskRank, corresponds to obtaining the MSL embeddings (i.e., one forward pass over the Longformer model). LMMaskRank is thus much more demanding, given that LMEmbedRank only needs to compute MSL embeddings once, while LMMaskRank needs a separate computation for each candidate (i.e., replacing the candidate occurrences with `[MASK]` tokens, before computing the representations).

		True Label	
		Key-phrase	Non Key-phrase
Predicted Label	Key-phrase	True Positive (TP)	False Positive (FP)
	Non Key-phrase	False Negative (FN)	True Negative (TN)

Table 3.1: Confusion matrix for the key-phrase extraction task.

3.1.4 Combining Both Ranking Approaches

Longformer Multilingual Rank (LMRank) corresponds to a hybrid approach that uses weighted averages of scores obtained by both previous methods, based on the hypothesis that each method would be better suited to handle different types of documents, and thus combined they could achieve better results. This general approach can be implemented through any number of different combination schemes. Empirically, the unnormalized arithmetic average and the harmonic average of LMEmbedRank and LMMaskRank scores were tested and yielded good results.

3.2 Experimental Evaluation

This subsection starts by introducing the evaluation metrics that were used during the course of this dissertation, alongside the datasets that the approaches were experimented in. It then follows with an overview on the experimental results across all datasets, providing a comparison with previous methods, as well as against ablated versions of the proposed approaches.

3.2.1 Evaluation Metrics

As previously stated, the two main segments of this KPE task, namely the extraction of candidate phrases and the ranking of the extracted candidate key-phrases, were evaluated separately with different evaluation metrics and objectives in mind.

For the first segment, the evaluation was performed with an unordered set of results in mind, as the objective of candidate extraction is not to rank but only to identify key-phrase candidates. In order to evaluate this task, the system was assessed in terms of Precision (P), Recall (R) and F_1-score (F_1), stemming the candidates in order to ensure minimal mismatches. These metrics are computed based on a confusion matrix such as that presented in Table 3.1, according to the following equations:

$$P = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}. \tag{3.1}$$

$$R = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}. \tag{3.2}$$

$$F_1 = \frac{2\mathrm{PR}}{\mathrm{P}+\mathrm{R}}. \tag{3.3}$$

The most important measure to the quality of the candidate-phrase extraction component is the Recall value, as without a high Recall it will be impossible to accurately rank key-phrases, as correct key-phrases will not be available to be ranked. It is interesting to note that there is an upper bound on the possible Recall value, as the candidate extraction method is unable to find correct key-phrases that do not appear within the text document (although the lemmatization operation does help in this regard). The candidate extraction Recall, for each dataset, is shown in Table 3.2, coupled with the percentage of absent key-phrases from each dataset, allowing a more accurate analysis of the limitations of the extraction method.

In the ranking segment, the result space was handled as an ordered set with a specific cut-off point k defined for each metric, comparing the top k ranked candidates of the system with the top k true label key-phrases. To follow common evaluation metrics for KPE systems that allow establishing comparisons between the proposed approach and others, the performance was evaluated on the ranking task by using Precision (P_k), Recall (R_k) and F_1-score ($F_{1,k}$), at the cut-off points $k = \{5, 10, 15\}$. These metrics were calculated exactly as explained previously, but only considering the top k retrieved candidates.

Furthermore, in order to have a more nuanced evaluation, Mean Average Precision (MAP) and Normalized Discounted Cumulative Gain (nDCG) were also used as ranking evaluation metrics. As the name suggests, MAP calculates the mean of the average precisions across a set of documents (in their domain). With $\mathfrak{D}$ as the set of all evaluated documents, $\mathfrak{R}_\mathfrak{d}$ as the set of true relevant key-phrases on a document d and $rel\,(i,d)$ as a binary function that, for each key-phrase, takes the value of 1 if it matches with a true relevant key-phrase for d, and 0 otherwise, the MAP is calculated as:

$$\begin{aligned} MAP &= \frac{\sum_{d\in\mathfrak{D}} \mathrm{AveP}\,(d)}{|\mathfrak{D}|}, \\ \mathrm{AveP}\,(d) &= \frac{\sum_{k=1}^{n} P_\mathrm{k}\,(d)\; rel\,(k,d)}{|\mathfrak{R}|}. \end{aligned} \tag{3.4}$$

As for the nDCG, it is a measure commonly used in IR tasks to measure ranking quality based on the position of each relevant result in the ranking. With $\mathfrak{R}_{k,d}$ as the set of true label key-phrases at cut-off point k for a document d, when sorted according to relevance, nDCG at the same cut-off is given by:

$$DCG_\mathrm{k}\,(d) = \sum_{i=1}^{k} \frac{rel\,(i,d)}{\log_2(i+1)}, \tag{3.5}$$

$$IDCG_\mathrm{k}\,(d) = \sum_{i=1}^{|\mathfrak{R}_k|} \frac{2^{rel\,(i,d)}-1}{\log_2(i+1)}, \tag{3.6}$$

$$nDCG_{\mathrm{k}}\left(d\right) = \frac{\mathrm{DCG_k}}{\mathrm{IDCG_k}}. \tag{3.7}$$

3.2.2 Datasets

To evaluate the performance of our proposed approaches concerning different languages and domains, several pre-established datasets were used. Breaking them down by languages, five datasets in English were chosen, namely: DUC-2001 (DUC) (Wan and Xiao, 2008), composed by news articles extracted from TREC-9 that could be categorized in 30 different topics, with manually annotated key-phrases for each document; NUS (Nguyen and Kan, 2007), containing long scientific conference papers, with manually assigned key-phrases; Inspec (Hulth, 2003b), comprised of abstracts from scientific papers found in journals, on the disciplines of Computers and Control and Information Technology; SemEval-2010 (SemEval) (Kim et al., 2010), composed by conference and workshop papers from the ACM Digital library, with carefully chosen key-phrases from both the authors and readers of those papers; and PubMed (Aronson et al., 2000), created from medical publications collected from PubMed Central, with manually assigned key-phrases. A single Portuguese dataset, 110-PT-BN-KP (PT-KP) (Marujo et al., 2013), constituted by expertly annotated documents from the ALERT corpus which contains solely news broadcasts. Two Spanish datasets, CACIC (Aquino and Lanzarini, 2015), formed by a set of documents published by the Argentine Congress of Computer Science, expertly annotated; and WICC (Aquino and Lanzarini, 2015), which entails scientific articles published by the Workshop of Researchers in Computer Science, with author assigned key-phrases. A French dataset, WikiNews (FR-Wiki) (Bougouin et al., 2013b), created from news publications from the French version of WikiNews, with each document annotated by at least three different students. Finally, a German dataset, TeKET (Rabby et al., 2020), consisting of articles collected from various open score research databases, using author assigned key-phrases as ground truth.

Additionally, in order to further expand research on key-phrase extraction in the epidemiological domain, a new dataset was created, named ResisBank. ResisBank is a collection of 400 articles in English retrieved from ResistanceBank.org [5], an open access repository for surveys and maps of antimicrobial resistance in animals. These articles are available in plain text form, each with manually assigned key-phrases from the authors combined with complementary key-phrases from the repository itself, e.g. the names of the animals covered or the microorganisms it refers to. In order to offer a realistic scenario for KPE, explicit mentions to the author assigned key-phrases were removed from the articles, which does increase the difficulty of the task. ResisBank is publicly available online [6] for free open-source usage.

Statistical characterization information is presented in Table 3.2, including the average number of

[5] https://resistancebank.org/
[6] https://github.com/araag2/ResistanceBank-Dataset

Dataset	Language	Avg. #KPs	Ex. Recall	Absent KPs	Avg. #Words	#Docs
DUC	EN	8	87.22%	6.77 %	740	308
NUS	EN	11	88.19%	4.30 %	5201	209
Inspec	EN	10	58.70%	35.57 %	128	2000
SemEval	EN	16	95.00%	3.22 %	8332	243
PubMed	EN	15	80.20%	15.80 %	3992	1320
ResisBank	EN	5	93.45%	3.31 %	5616	400
PT-KP	PT	24	53.56%	5.21 %	304	110
CACIC	ES	5	72.34%	7.33 %	3985	888
WICC	ES	5	74.32%	5.96 %	1955	1640
FR-WIKI	FR	12	79.12%	4.37 %	293	100
TeKET	DE	5	93.50%	0.00 %	11524	10

Table 3.2: Statistics for the considered datasets.

ground truth key-phrases per document, the number of ground truth key-phrases that are absent from their respective documents, the average number of words per document, and the number of documents in each dataset.

3.2.3 Experimental Results

Evaluating the candidate extraction segment, with the Recall statistics presented in Table 3.2, the high Recall objective was mostly achieved throughout all datasets, with the two notable exceptions being the Inspec and the PT-KP datasets. For the Inspec dataset, noting the 35.57% absent key-phrases, the extraction method performed quite well given the theoretical maximum of 64.431% extraction Recall, and the obtained results seem well within the limits of sole candidate extraction. The same cannot be said for the PT-KP dataset, which presents a very low candidate absence of 5.21% and the lowest value in extraction Recall. I theorize this is due to the composition of the dataset, being a small dataset in terms of number of documents (100) and in average number of words (304). However, it presents, by far, the highest number of key-phrases per document (24), which leads to the regular expression pattern being unable to find many of the ground truth key-phrases, which do not always correspond to noun phrases.

Table 3.3 presents experimental results over the multiple datasets that were also used in previous studies, comparing the alternatives discussed in Section 3.1. The lines named LMRank$_{avg_a}$ and LMRank$_{avg_h}$ correspond to using an arithmetic or harmonic average of LMEmbedRank and LMMaskRank scores, as described in Subsection 3.1.4.

Although the different evaluation metrics mostly agree on how the methods should be ranked according to result quality, different methods can perform slightly better on some of the datasets:

- LMEmbedRank, which is also the more effective method computationally, performs clearly better than LMMaskRank on the DUC, NUS, Inspec, SemEval, and CACIC datasets.

Model	DUC				NUS				Inspec				SemEval				PubMed			
	nDCG	$F1_5$	$F1_{10}$	$F1_{15}$	nDCG	$F1_5$	$F1_{10}$	$F1_{15}$	nDCG	$F1_5$	$F1_{10}$	$F1_{15}$	nDCG	$F1_5$	$F1_{10}$	$F1_{15}$	nDCG	$F1_5$	$F1_{10}$	$F1_{15}$
LMEmbedRank	**45.01**	**30.28**	**33.72**	**34.10**	28.55	20.61	22.87	23.12	**45.95**	**31.25**	**36.97**	**38.05**	22.16	14.49	18.74	**20.93**	13.14	7.94	10.22	10.28
LMMaskRank	37.00	24.41	28.31	28.15	30.53	18.58	20.01	17.38	41.19	27.75	33.31	34.78	21.10	15.38	17.48	18.10	27.72	17.36	18.93	18.14
LMRank$_{avg_a}$	39.57	25.58	29.96	31.66	28.16	22.76	24.49	24.86	43.54	29.66	35.34	36.45	19.78	14.59	16.22	16.27	23.65	15.1	16.46	**20.35**
LMRank$_{avg_h}$	40.98	29.68	31.29	30.05	**38.11**	**28.17**	**32.69**	**31.79**	43.43	30.10	34.67	35.14	**22.97**	**16.69**	**19.17**	18.25	**29.48**	**17.94**	**20.15**	18.74
Model	PT-KP				CACIC				WICC				FR-WIKI				TeKET			
	nDCG	$F1_5$	$F1_{10}$	$F1_{15}$	nDCG	$F1_5$	$F1_{10}$	$F1_{15}$	nDCG	$F1_5$	$F1_{10}$	$F1_{15}$	nDCG	$F1_5$	$F1_{10}$	$F1_{15}$	nDCG	$F1_5$	$F1_{10}$	$F1_{15}$
LMEmbedRank	36.78	23.91	32.91	37.29	**40.15**	**26.49**	**16.93**	13.20	24.88	15.31	14.85	13.97	43.64	24.15	32.92	26.50	6.48	0.00	6.95	6.15
LMMaskRank	39.90	25.96	34.38	37.63	22.52	16.75	14.72	14.07	27.50	17.76	16.48	14.31	49.62	37.06	37.65	34.83	**17.65**	10.44	**10.95**	**10.75**
LMRank$_{avg_a}$	**41.26**	28.04	**36.62**	**39.59**	17.68	11.09	13.13	**15.37**	17.99	13.20	12.37	11.20	50.88	37.53	**39.97**	**36.19**	15.87	**10.66**	10.28	10.15
LMRank$_{avg_h}$	41.09	**28.06**	35.79	38.91	25.64	19.11	15.98	14.77	**31.43**	**20.57**	**17.29**	**15.01**	**49.95**	**37.93**	38.43	35.86	**17.65**	10.44	**10.95**	**10.75**

Table 3.3: Key-phrase extraction results on each dataset and for each of the proposed methods.

- In turn, LMMaskRank performs better than LMEmbedRank on Pubmed, PT-KP, WICC, FR-WIKI, and TeKET.
- Datasets like NUS, SemEval, PubMed, and particularly TeKET, feature very long documents, beyond the 4096 token limit in Longformer. In these cases, LMMaskRank tends to perform better, although the relation between result quality and the characteristics of the documents (e.g., size, language, or candidate Recall) is not entirely clear. Note that LMMaskRank is biased towards preferring candidates occurring in the first 4096 tokens, since occurrences beyond this limit will not impact the representations (i.e., the representations for these candidates are exactly equal to those from the documents, and hence they will be ranked below other candidates).
- The combination of both approaches, particularly when considering the harmonic mean, is beneficial in most cases. In the NUS, SemEval, PubMed, WICC, and FR-WIKI datasets, the best results are achieved with a combined method. On the other datasets, the combination performs similarly to the best method, improving over the LMEmbedRank or LMMaskRank strategies.

Table 3.4 presents results for ablated versions of the LMEmbedRank, LMMaskRank, and LMRank$_{avg_h}$ methods, specifically assessing the impact of the different ideas introduced in our proposal. The following alternatives were tested on 5 of the datasets also seen in Table 3.3:

- Using the regular Sentence-Transformers model based on a multilingual RoBERTa, instead of converting the model into a Longformer. In the case of LMEmbedRank, the candidates that occur after the maximum token limit of the model were also represented with the back-off procedure that processes the candidate string alone, without any document context.
- Using a standard English Longformer[7], instead of the Sentence-Transformers model pre-trained only for MLM. This way, we can assess the impact of model pre-training with sentence similarity

[7] https://huggingface.co/allenai/longformer-large-4096

Model	DUC				NUS				Inspec				SemEval				PT-KP			
	nDCG	$F1_5$	$F1_{10}$	$F1_{15}$	nDCG	$F1_5$	$F1_{10}$	$F1_{15}$	nDCG	$F1_5$	$F1_{10}$	$F1_{15}$	nDCG	$F1_5$	$F1_{10}$	$F1_{15}$	nDCG	$F1_5$	$F1_{10}$	$F1_{15}$
LMEmbedRank	45.01	30.28	33.72	34.10	28.55	20.61	22.87	23.12	45.95	31.25	36.97	38.05	22.16	14.49	18.74	20.93	36.78	23.91	32.91	37.29
- Longformer	40.92	27.56	31.26	31.23	18.12	11.14	13.52	12.49	44.51	29.70	35.40	36.46	12.35	1.82	8.69	12.05	34.82	21.41	30.95	35.37
- Sentence-BERT	26.86	15.75	22.10	22.50	17.54	12.23	12.57	11.90	23.68	17.54	19.80	18.27	10.27	2.03	7.07	11.32	-	-	-	-
- Lematization	42.10	29.09	31.85	31.93	25.79	19.31	20.72	21.14	43.42	29.88	35.01	36.31	19.21	12.78	16.48	18.88	33.84	22.43	30.89	35.28
- Global Attention	44.15	29.59	33.16	33.37	27.56	19.93	22.22	22.41	45.23	30.88	36.57	37.6	21.18	13.77	18.05	20.34	35.87	23.03	32.32	36.64
LMMaskRank	37.00	24.41	28.31	28.15	30.53	18.58	20.01	17.38	41.19	27.75	33.31	34.78	21.10	15.38	17.48	18.10	39.90	25.96	34.38	37.63
- Longformer	33.89	22.46	25.85	24.82	20.79	9.75	10.72	7.78	39.83	26.34	31.75	33.11	12.21	2.99	7.91	8.69	36.42	22.75	30.95	35.03
- Sentence-BERT	32.92	22.39	27.71	26.50	23.74	12.36	14.09	11.83	27.09	20.45	22.82	23.03	15.64	10.93	12.99	14.21	-	-	-	-
- Lemmatization	34.23	23.1	26.19	26.25	27.55	17.49	18.09	15.28	38.29	26.32	31.47	32.87	18.13	13.90	15.22	16.5	33.65	20.56	28.82	30.97
- Global Attention	36.04	23.50	27.37	27.01	29.60	18.00	19.25	16.61	40.58	27.46	32.94	34.27	20.26	14.36	16.96	17.44	39.05	23.71	31.55	35.07
LMRank$_{avg_h}$	40.98	29.68	31.29	30.05	38.11	28.17	32.69	31.79	43.43	30.10	34.67	35.14	22.97	16.69	19.17	18.25	41.09	28.06	35.79	38.91
- Longformer	37.62	27.22	29.14	27.14	29.37	21.40	27.32	25.38	41.25	28.86	32.87	33.21	15.27	8.92	11.28	11.91	37.87	25.15	33.33	36.13
- Lemmatization	38.32	28.14	29.49	28.13	36.09	27.09	30.88	29.95	41.31	28.91	32.78	33.31	20.67	15.04	17.44	16.51	33.39	20.71	29.63	31.58
- Global Attention	39.68	28.63	30.14	28.78	37.12	26.92	31.65	30.47	42.60	28.72	33.90	34.30	22.10	15.66	18.48	17.07	40.13	26.42	34.79	37.64
+ Weighting	40.57	29.06	30.82	29.33	38.81	28.65	33.23	32.38	42.64	29.62	34.14	34.52	22.36	16.32	18.85	17.65	41.83	28.45	36.25	39.53

Table 3.4: Results with ablated versions of the proposed key-phrase extraction methods.

tasks, noting also that previous studies such as MDERank (Zhang et al., 2021) have only explored the use of regular Transformer encoders pre-trained for MLM.

- Removing the lemmatization procedure that aggregates similar candidates appearing with a slightly different surface form.
- Removing the Longformer attention pattern that considers a global attention for the tokens that correspond to candidate occurrences, instead leaving only the `[CLS]` token with the global attention over all other tokens.

The results show that all the four previous aspects, and particularly the pre-training over sentence similarity tasks (i.e., using an adapted Sentence-Transformers model) and the conversion of the Sentence-Transformers model into a Longformer, contribute to improved results. Higher differences in the result quality are also seen in the datasets involving longer documents.

Besides the aforementioned ablations, extensions over the methods in Table 3.3 were also considered, leveraging ideas advanced in previous studies. These included: (a) post-processing the document/candidate embeddings prior to computing similarities (Sajjad et al., 2021; Artetxe et al., 2018; Jégou and Chum, 2012; Su et al., 2021), or (b) weighting the individual tokens when computing the representations within LMEmbedRank, e.g. proportionally to attention scores (Ding and Luo, 2021). Still, results were consistently worse, and I decided not to report these scores.

One particular extension that was tested involved weighting the scores of the LMEmbedRank and LMMaskRank methods prior to their combination, in an attempt to further improve results. In a first step towards doing this, the distribution of the similarity scores between candidate and document representations was analyzed, for the two different methods (i.e., LMEmbedRank and LMMaskRank) and over the

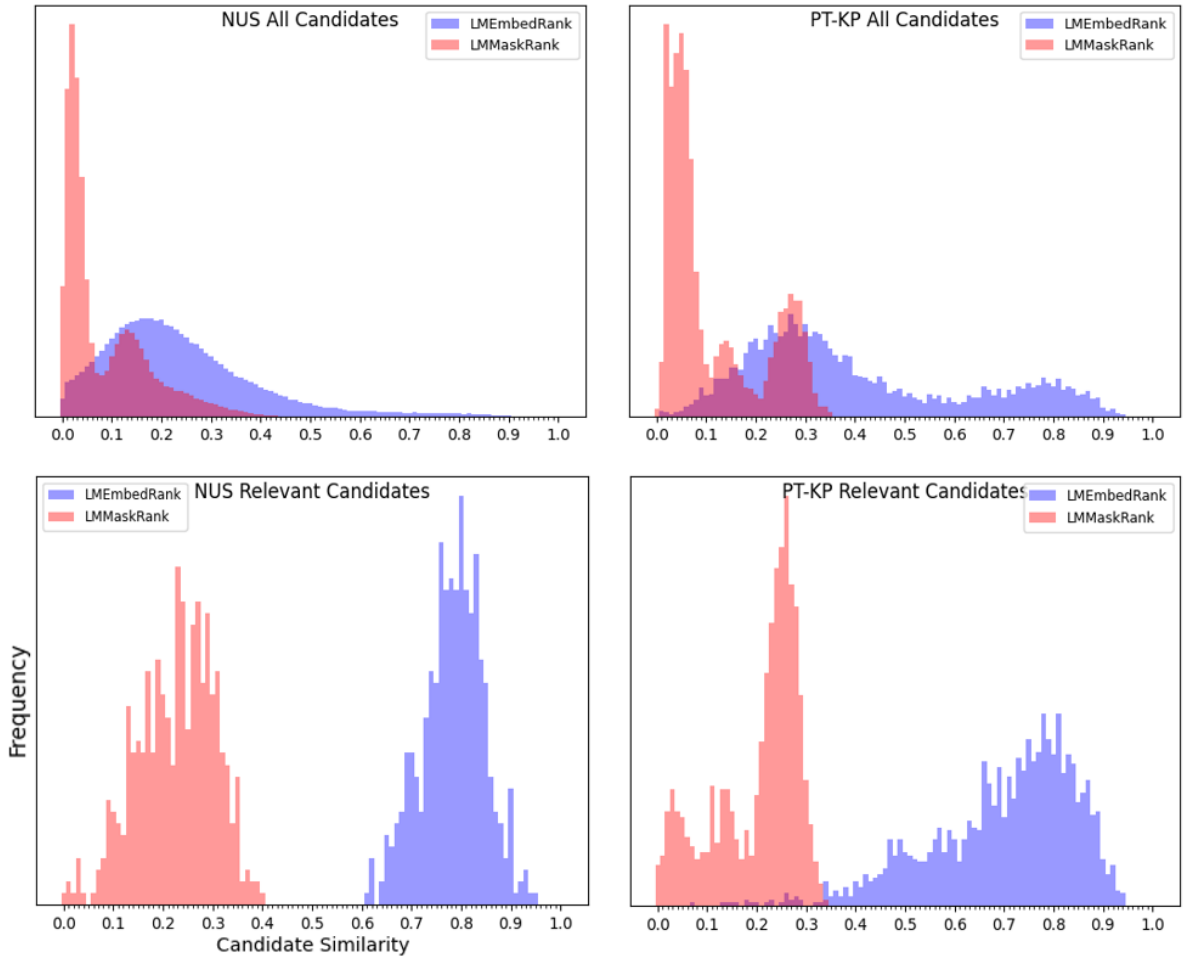

Figure 3.4: Distribution of similarity scores between candidates and documents, considering all candidates and documents for two different datasets (on top), or only the subsets of relevant candidates (at the bottom).

different datasets. Figure 3.4 illustrates the results of this analysis, specifically for the NUS and PT-KP datasets. Similar patterns could also be observed for the other datasets.

The results showed that LMEmbedRank produces similarity scores that are more evenly spread, whereas LMMaskRank mostly produces results in the interval $[0, 0.5]$. Both methods produce two peaks in the distribution for the similarity values, corresponding to a good distinction between the relevant and the irrelevant candidates (i.e., the top charts in Figure 3.4 correspond to the entire sets of candidates, whereas the bottom charts show only the similarity scores for the subset of relevant candidates).

Based on this analysis, a combination method was tested in which a constant of 0.5 was added to the scores from LMMaskRank procedure, prior to the combination with LMEmbedRank. The results are shown in the bottom row from Table 3.4, although no noticeable improvements were seen.

With regards to previous work, Table 3.5 compares the best proposed methods, specifically LMEmbedRank (i.e., the simplest and fastest method) and LMRank$_{avg_h}$, against the results reported in publications pre-

Model	DUC			NUS			Inspec			SemEval			PT-KP			TeKET		
	$F1_5$	$F1_{10}$	$F1_{15}$	$F1_5$	$F1_{10}$	$F1_{15}$	$F1_5$	$F1_{10}$	$F1_{15}$	$F1_5$	$F1_{10}$	$F1_{15}$	$F1_5$	$F1_{10}$	$F1_{15}$	$F1_5$	$F1_{10}$	$F1_{15}$
TF-IDF	9.21	10.63	11.60	11.60	14.20	12.50	24.20	28.00	24.80	16.10	16.70	15.30	-	17.9	-	7.50	8.60	9.60
TopicRank	19.97	21.73	20.97	4.54	7.93	9.37	12.20	17.24	19.33	9.93	12.52	12.26	-	14.80	-	6.30	7.60	8.10
EmbedRank	21.75	25.09	24.68	2.13	2.94	3.56	14.51	21.02	23.79	9.63	13.90	14.79	-	-	-	-	-	-
SIFRank	24.30	27.60	27.96	3.01	5.34	5.86	29.38	**39.12**	**39.82**	11.16	16.03	18.42	-	-	-	-	-	-
MDERank	13.05	17.31	19.13	15.24	18.33	17.95	26.17	33.81	36.17	10.16	15.32	17.76	-	-	-	-	-	-
AttentionRank	-	-	-	-	-	-	24.45	32.15	34.49	11.39	15.12	16.66	-	-	-	-	-	-
YAKE!	11.99	14.18	14.18	7.85	11.05	13.09	8.02	11.47	13.65	6.82	11.01	12.55	-	10.70	-	8.83	**12.30**	**13.80**
Multipartite	21.70	24.10	23.62	6.17	8.57	10.82	13.41	18.18	20.52	10.13	12.91	13.24	12.05	15.60	-	7.10	9.10	9.70
CDKGen	-	-	-	41.20	38.10	-	33.10	34.70	-	34.20	35.50	-	-	-	-	-	-	-
SEG-Net	-	-	-	39.60	-	-	21.60	-	-	28.30	-	-	-	-	-	-	-	-
SKE-Base-Rank	-	-	-	38.90	36.50	-	28.90	32.10	-	35.40	33.70	-	-	-	-	-	-	-
LMEmbedRank	**30.28**	**33.72**	**34.10**	20.61	22.87	23.12	**31.25**	36.97	38.05	14.49	18.74	**20.93**	23.91	32.91	37.29	0.0	6.95	6.15
LMRank$_{avg_h}$	29.68	31.29	30.05	**28.17**	**32.69**	**31.79**	30.10	34.67	35.14	**16.69**	**19.17**	18.25	**28.06**	**35.79**	**38.91**	**10.44**	10.95	10.75

Table 3.5: Comparison between our best key-phrase extraction methods and previously published results.

senting and using previous methods (including results for the original EmbedRank (Bennani-Smires et al., 2018a) and MDERank (Zhang et al., 2021) methods). Results are presented for the datasets over which more previous methods have been tested (i.e., mostly by re-using results presented on previous comparisons (Zhang et al., 2021)), also including results for some recent supervised approaches (i.e., the second set of rows in Table 3.5).

The results in Table 3.5 show that the proposed approaches are very competitive within the realm of unsupervised KPE, outperforming most unsupervised methods in the majority of the datasets and often by a very large margin, while simultaneously being simple, multilingual, and thus easy to generalize to different types of applications.

Notable exceptions correspond to the Inspec and the TeKET datasets. In the specific case of Inspec, SIFRank (Sun et al., 2020) outperforms the proposed methods in $F1_{10}$ and $F1_{15}$, likely due to the small size of the documents (128 words on average) which offset the positive Longformer effect. On TeKET, YAKE! (Campos et al., 2018) outperforms the proposed approaches also in $F1_{10}$ and $F1_{15}$, but in this case it is difficult to draw many conclusions because the dataset only features 10 very long documents (11524 words on average), and hence the results can be very noisy.

It is also important to notice that the differences towards recent supervised methods are still very significant. Previous methods such as CDKGen (Diao et al., 2020), SEG-NET (Ahmad et al., 2021), or SKE-Base-Rank (Mu et al., 2020) are, usually, still significantly better than the best unsupervised approaches, although this also varies depending on characteristics of the datasets (e.g., on Inspec, the best results in terms of $F1_{10}$ and $F1_{15}$ are obtained with unsupervised methods).

Finally, Table 3.6 presents the results of using the proposed methods on the newly created dataset for this work, i.e. ResisBank. These results offer a future benchmark for analysis in this dataset, allowing future studies to compare themselves with the obtained results.

The hybrid approaches seem to consistently outperform the non-hybrid ones, with each weighting

Model	ResisBank										
	MAP	nDCG	P_5	R_5	$F1_5$	P_{10}	R_{10}	F_{10}	P_{15}	R_{15}	$F1_{15}$
LMEmbedRank	3.25	8.93	4.81	5.20	4.92	5.01	10.75	6.75	4.38	13.83	6.59
LMMaskRank	15.01	27.76	19.40	20.47	19.68	15.10	31.47	20.19	11.73	36.33	17.59
LMRank$_{avg_a}$	16.86	30.57	**22.40**	**23.23**	**22.58**	**16.00**	**33.30**	**21.44**	12.40	38.36	18.63
LMRank$_{avg_h}$	**18.23**	**32.48**	21.60	22.23	21.72	15.80	33.03	21.19	**13.06**	**40.50**	**19.63**

Table 3.6: Benchmark key-phrase extraction results on the proposed ResisBank dataset.

scheme having several best scores. LMEmbedRank is vastly outclassed by LMMaskRank, presenting much worse results, which I theorize is an effect of the large average document size (5616 words).

3.3 Conclusions and Future Work

In this Chapter I proposed new unsupervised methods for key-phrase extraction, extending the previous EmbedRank (Bennani-Smires et al., 2018a) and MDERank (Zhang et al., 2021) approaches in different directions. The proposed approaches were tested over multiple datasets, with results showing a very competitive performance against state-of-the-art unsupervised methods, while also generalizing across different languages and domains.

A new dataset targeting on researching KPE in the epidemiological domain was also created, titled ResisBank, on which a very solid benchmark analysis was conducted envisioning future usage.

For future work, other methods for handling long inputs besides the Longformer (e.g., memory efficient attention implementations (Rabe and Staats, 2021)) can be considered. In the post-processing segment, even though some techniques were tested that had success in previous work (Sajjad et al., 2021; Artetxe et al., 2018; Jégou and Chum, 2012; Su et al., 2021) and obtained worse results with the proposed methods, other post-processing techniques can be explored that might be better suited to the models used. The random attention patterns proposed by Zaheer et al. (2020) or other novel attentions patterns, like the Hypercube Transformer Wang et al. (2022) can also be further investigated, as new ideas might offer better results. Finally, the combination weighting schemes used in the hybrid approaches also warrants further analysis, as it is possible that with some fine-tuning or with non-linear combination schemes better results might be obtained.

4

Multi-Document Key-Phrase Extraction

Contents

This chapter describes the proposed approaches for MDKPE. Similarly to chapter 3, Section 4.1 starts by describing the proposed approaches. Section 4.2 provides an overview on all datasets and metrics used to evaluate the work, as well as preliminary results. Finally, Section 4.3 offers insight on future work to be done in order to extend the implementation of the proposed approaches, and conclusions found from the work completed until now.

4.1 Proposed Approaches

Albeit sharing core similarities with KPE, it is crucial to formally distinguish MDKPE from it: given a topic t composed by several text documents d and belonging to a dataset $\mathfrak{D}$, the goal is to extract a set $\mathfrak{C}$ of candidate phrases c that correspond to the relevant key-phrases for that topic t. In order to do so, firstly a wide set $\mathfrak{C}$ is extracted and afterwards these candidate phrases are ranked, outputting the top-k candidates within the set as the chosen key-phrases.

The proposed KPE approaches, which were detailed in Section 3.1, can be broken down in three main steps: (1) extracting candidates from each input document corresponding to noun phrases, (2) representing the input document and its candidates as embedding vectors, and (3) ranking the candidates through the cosine similarity measure between representations (i.e., between each candidate embedding and the input document embedding). In the case of MDKPE, the input consists of a set of documents sharing a common topic, instead of an individual document. The procedure thus requires some adaptations, which I propose to do through small changes in steps (2) and (3) from the previous enumeration.

Adapting the proposed LMEmbedRank approach, the candidates are similarly extracted from each of the documents within an input set of documents t that correspond to a given topic, and then, in step (2) when representing candidate phrases through embeddings, the multiple occurrences of each candidate phrase within that set are considered (i.e., first averaging the token representations from each occurrence, then averaging across all occurrences within each document, and finally averaging across all the documents in which the candidate occurs). For ranking the candidates in step (3), which in this case correspond to candidates appearing in any of the documents within that set, the cosine similarity between each candidate embedding and each document embedding is firstly measured, including the documents in which the candidate does not occur. Then, the final score for each candidate is calculated as the average score across all documents, even the ones they do not occur in.

As for LMMaskRank, the changes to step (2) were different. Since the representation of each candidate is not the average of each of their occurrences, but a single embedding with each candidate occurrence replaced with the `[MASK]` token, the candidate embeddings are calculated by averaging the masked document embeddings from all documents within the set that contain the candidate. This

adaptation has two major flaws: firstly, it averages the masked document embeddings of different documents, which creates a new embedding that is much more similar to the average embeddings of the original documents than to a candidate representation; and secondly, it creates an enormous bias towards candidates that do not occur in multiple documents, as those candidates will be less similar to multiple documents and thus, on average, have a better classification. Step (3) works identically to the aforementioned LMEmbedRank adaptations.

4.2 Experimental Evaluation

This subsection starts by introducing the specific evaluation metrics that were used to evaluate the MDKPE task, alongside the dataset it was experimented in. An overview on the experimental results obtained on the dataset follows, providing a comparison with previous methods.

4.2.1 Evaluation Metrics

The chosen metrics to evaluate results in the MDKPE task are identically to the ones for SDKPE, which were detailed in Section 3.2. Following the same scheme, evaluation will be divided in two segments: the extraction of candidate phrases, which will focus on having a high Recall value to allow the methods to have accurate key-phrases in their candidate sets; and the ranking of said candidate phrases, which will use the aforementioned Precision (P_k), Recall (R_k) and F_1-score ($F_{1,k}$), at the cut-off points $k = \{5, 10, 15\}$, coupled with the MAP and nDCG measures to provide a more nuanced analysis.

Additionally, in order to compare the results with the benchmark results obtained by Shapira et al. (2021), the same exact metrics will be used, the F_1-score ($F_{1,k}$) at the cut-off points $k = \{1, 5, 10, 20\}$.

4.2.2 Datasets

The evaluation for MDKPE uses a single English dataset, the MK-DUC-01 dataset introduced by Shapira et al. (2021). MK-DUC-01 was built from the DUC (Wan and Xiao, 2008) SDKPE dataset, which in turn used the documents from the DUC-2001 multi-document summarization dataset, consisting of 30 topics with an average of 10.27 related news articles per topic (308 in total). To restructure the dataset for the multi-document setting, Shapira et al. (2021) used an automatic merging and re-ranking process (i.e., merging the lists of key-phrases from the individual documents within each topic, removing phrase duplicates, ranking the key-phrases according to document frequency, and using heuristics to remove overly generic or repetitive key-phrases), followed by a manual refinement procedure (i.e., manually revising the top 30 candidates from the previous step, and keeping the best 20 items for each topic, as the ground truth results). Additional information on this dataset can be found in Table 4.1.

Dataset	Language	#Topics	Avg. #KPs	Ex. Recall	Absent #Kps	Avg. #Words	#Docs
MK-DUC-01	EN	30	44	96.14%	3.09%	704	308

Table 4.1: Statistics for the considered datasets.

Model	MK-DUC-01										
	MAP	nDCG	P_5	R_5	$F1_5$	P_{10}	R_{10}	F_{10}	P_{15}	R_{15}	$F1_{15}$
LMEmbedRank	**15.87**	**59.02**	**24.00**	**6.82**	**8.79**	**22.23**	**9.48**	**12.44**	**22.88**	**12.47**	**15.71**
LMMaskRank	5.20	45.45	12.66	5.56	5.14	9.33	6.40	6.20	7.55	6.93	6.22
LMRank$_{avg_a}$	6.33	48.52	15.03	5.82	6.76	9.56	7.01	7.83	14.57	9.43	11.07
LMRank$_{avg_h}$	6.57	50.02	15.25	5.87	7.04	9.88	7.13	7.92	14.84	9.59	11.90

Table 4.2: Benchmark key-phrase extraction results on the MK-DUC-01 dataset.

A slightly different version of this MK-DUC-01 dataset was also used in its original work to better measure obtained results, entitled the $Trunc - 20$ version. The only difference this version presents is the ground truth key-phrases being truncated to the top 20 key-phrases, in order to establish a more representational task-setting since each key-phrase will be more salient to their own topics.

4.2.3 Experimental Results

Looking at the obtained results in the candidate extraction segment, with its Recall statistics presented in Table 4.1, a very high value of 96.14% was achieved, just $\sim 1\%$ short of the theoretical maximum for an extraction method based on regular expressions.

In the candidate ranking segment, Table 4.2 contains all experimental results obtained from adapting all four of the SDKPE methods. All evaluation metrics agree that LMEmbedRank appears to be by far the best method in the MK-DUC-01 dataset, obtaining the best results across the board. I theorize this is due to the fact that in its current form LMMaskRank is ill-suited for the task at hand, as a great bias is present towards candidates that appear in single documents within a topic, which is a great hindrance to the overall results. The combined approaches mirror these findings, as LMMaskRank appears to be greatly weighting down LMEmbedRank.

Table 4.3 compares the proposed approaches with baseline results presented by Shapira et al. (2021) on the MK-DUC-01 article. The main takeaway from this comparison is that LMEmbedRank is very competitive with past methods, achieving the best results in $F1_1$ and $F1_5$ and the second best results in $F1_{10}$, providing a robust approach from which to extend new methods from. As for other approaches, their results are far below average even across pre-existing baselines, reaffirming that LMMaskRank must not be suited for the task at hand. It is also important to note that these results were attained by using the $Trunc - 20$ version of the MK-DUC-01 dataset, which seems to favour LMEmbedRank and to damage other results.

Model	MK-DUC-01 ($Trunc - 20$)			
	$F1_1$	$F1_5$	$F1_{10}$	$F1_{20}$
Tf-Idf	0.63	1.60	2.44	4.50
KPMiner	1.59	5.34	7.56	10.84
YAKE	2.86	5.87	8.23	10.84
TextRank	2.54	9.88	13.79	17.17
SingleRank	2.86	8.80	14.23	18.51
TopicRank	4.13	10.94	16.01	18.68
TopicalPageRank	2.86	9.08	15.56	19.84
PositionRank	3.17	8.53	15.56	19.51
MultipartiteRank	4.13	11.21	**17.34**	**21.34**
CollabRank	2.86	9.61	14.90	17.84
multi-doc	0.09	0.93	1.46	1.88
LMEmbedRank	**4.95**	**12.69**	16.41	19.61
LMMaskRank	2.02	3.65	3.40	4.48
LMRank$_{avg_a}$	2.18	4.07	5.01	7.20
LMRank$_{avg_h}$	2.27	4.86	5.73	8.43

Table 4.3: Baseline results for the MK-DUC-01 ($Trunc - 20$) dataset.

4.3 Conclusions and Future Work

I proposed adapting the unsupervised approaches for key-phrase extraction to a multi-document scenario. The resulting adaptations showed a very competitive performance against state-of-the-art unsupervised methods in the MK-DUC-01 dataset.

As for future work, a problem that needs to be addressed is the poor performance of LMMaskRank. A way to counteract the bias this method presents would be to change the way candidates are ranked, where instead of calculating similarities between each candidate representation and each document, the approach could average all document representations into a single one and afterwards calculate the candidate similarity to that joint representation. In short, alternatively to calculating an average of similarities to each document, a similarity to the average of all documents would be calculated. Going beyond the aforementioned adaption it is possible to create variants that combine the semantic compatibility scores computed from the embeddings with two other factors: (1) the number of documents a candidate appears in, and (2) geo-spatial association measures capturing the locations that are strongly linked to the candidates that exist. A geoparser like Mordecai (Halterman, 2017), i.e. an open source tool which is able to resolve entities to geographic coordinates, can be used in order to improve candidate extraction by predicting similarities in geo-spatial regions (Gritta et al., 2018; Cardoso et al., 2022; Kulkarni et al., 2021), or by using known spatial association statistics (Moran, 1950; Cliff and Ord, 1981; Geary, 1954; Getis and Ord, 2010).

Furthermore, in order to widen the range of available datasets to test MDKPE in, and to address this task in a different domain, the creation of a dataset composed of articles extracted from the Wikipedia

page on epidemics [1] was started, focused on providing a resource with geo-referenced documents within the realm of epidemiology, with its progress publicly available online [2].

[1] https://en.wikipedia.org/wiki/List_of_epidemics
[2] https://github.com/araag2/WikipediaEpidemics-dataset

5

Conclusions and Future Work

Contents

This chapter presents the closing arguments of this dissertation, as well as an overarching view of the accomplished work, stating the main conclusions that can be drawn from the experimental results obtained and how the work this dissertation entails can be expanded upon in the future.

5.1 Conclusion

Over the course of this thesis, work was presented concerning two different key-phrase extraction tasks, one handling a single-document scenario and the other a multi-document one. In the first task, the KPE one, the main contribution was the development of new unsupervised approaches for key-phrase-extraction, extending previous methods namely EmbedRank (Bennani-Smires et al., 2018a) and MDERank (Zhang et al., 2021). The proposed approaches seek to address shortcomings of both these works, and obtained results that are on par with state-of-the-art unsupervised methods over multiple datesets, while also generalizing across different languages and domains.

Another noteworthy contribution of this work was the creation of a new dataset targeting research on the KPE task in the epidemiological domain, entitled ResisBank [1]. Using the aforementioned approaches, experiments were conducted in order to obtain benchmark results for future dataset usage.

Shifting the focus to the MDKPE task, the proposed KPE approaches were adapted to this multi-document scenario, using re-ranking operations over the single-document results. Albeit a very simple adaptation, the obtained results on the MK-DUC-01 dataset showed that this idea offers a very competitive performance against other novel unsupervised approaches.

Overall, the main takeaway from the concluded research is that it is possible to create unsupervised key-phrase extraction multilingual methods that offer similar results to ones restricted to specific languages or domains, motivating research on the limitations and the applicability of these results in real world scenarios, for example within the epidemiological domain or the summarization community.

5.2 Future Work

Even though the developed approaches showed promising results, there is significant room for improvement and experimentation. In the KPE task, other ways for handling long inputs besides the Longformer, like what Rabe and Staats (2021) proposed, can be considered, as well as a deeper exploration of post-processing techniques like Whitening (Artetxe et al., 2018; Jégou and Chum, 2012; Su et al., 2021) and Z-Score Normalization (Sajjad et al., 2021). Conjointly, other attention patterns can also be explored to further improve how the longformer is used, like the random attention process conducted by the BigBird (Zaheer et al., 2020) model or the novel Hypercube Transformer (Wang et al., 2022).

[1] https://github.com/araag2/ResistanceBank-Dataset

As for MDKPE, only a very surface level exploration was conducted, allowing future work to use the benchmark results and techniques as a basis for improvement. Further work can be done in order to better adapt KPE methods to the MDKPE task, like for example averaging all document representations into a single one to counteract existing biases. Other avenues such as using geo-spatial referencing (Gritta et al., 2018; Cardoso et al., 2022; Kulkarni et al., 2021) coupled with spatial association statistics (Moran, 1950; Cliff and Ord, 1981; Geary, 1954; Getis and Ord, 2010) can be considered, or even simpler ideas can be explored like taking into consideration the number of documents a given candidate appears in. A distinct lack of datasets remains a looming difficulty, which increases the importance of creating new datasets in order to further test and improve MDKPE.

Bibliography

Ahmad, W., Bai, X., Lee, S., and Chang, K.-W. (2021). Select, extract and generate: Neural keyphrase generation with layer-wise coverage attention. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Aquino, G. O. and Lanzarini, L. C. (2015). Keyword identification in Spanish documents using neural networks. *Journal of Computer Science & Technology*, 15.

Aronson, A. R., Bodenreider, O., Chang, H. F., Humphrey, S. M., Mork, J. G., Nelson, S. J., Rindflesch, T. C., and Wilbur, W. J. (2000). The NLM Indexing Initiative. In *Proceedings of the American Medical Informatics Association Symposium*.

Artetxe, M., Labaka, G., and Agirre, E. (2018). Generalizing and improving bilingual word embedding mappings with a multi-step framework of linear transformations. In *Proceedings of the Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence*.

Augenstein, I., Das, M., Riedel, S., Vikraman, L., and McCallum, A. (2017). SemEval 2017 task 10: ScienceIE-extracting keyphrases and relations from scientific publications. In *Proceedings of the International Workshop on Semantic Evaluation*.

Bayatmakou, F., Ahmadi, A., and Mohebi, A. (2017a). Automatic query-based keyword and keyphrase extraction. In *Proceedings of the IEEE Artificial Intelligence and Signal Processing Conference*.

Bayatmakou, F., Ahmadi, A., and Mohebi, A. (2017b). Automatic query-based keyword and keyphrase extraction. In *Procedings of the IEEE Artificial Intelligence and Signal Processing Conference*.

Beltagy, I., Peters, M. E., and Cohan, A. (2020). Longformer: The long-document transformer. *arXiv preprint 2004.05150*.

Bennani-Smires, K., Musat, C., Hossmann, A., Baeriswyl, M., and Jaggi, M. (2018a). Simple unsupervised keyphrase extraction using sentence embeddings. *arXiv preprint 1801.04470*.

Bennani-Smires, K., Musat, C., Hossmann, A., Baeriswyl, M., and Jaggi, M. (2018b). Simple unsupervised keyphrase extraction using sentence embeddings. In *Proceedings of the Conference on Computational Natural Language Learning*.

Berend, G. and Farkas, R. (2013a). Single-document keyphrase extraction for multi-document keyphrase extraction. *Computación y Sistemas*, 17(2):179–186.

Berend, G. and Farkas, R. (2013b). Single-document keyphrase extraction for multi-document keyphrase extraction. *Computación y Sistemas*, 17(2).

Boudin, F. (2015). Reducing over-generation errors for automatic Keyphrase Extraction using integer linear programming. In *Proceedings of the ACL Workshop on Novel Computational Approaches to Keyphrase Extraction*.

Boudin, F. (2018). Unsupervised keyphrase extraction with multipartite graphs. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*.

Bougouin, A., Boudin, F., and Daille, B. (2013a). TopicRank: Graph-based topic ranking for keyphrase extraction. In *Proceedings of the International Joint Conference on Natural Language Processing*.

Bougouin, A., Boudin, F., and Daille, B. (2013b). Topicrank: Graph-based topic ranking for keyphrase extraction. In *Proceedings of the International Joint Conference on Natural Language Processing*.

Campos, R., Mangaravite, V., Pasquali, A., Jorge, A. M., Nunes, C., and Jatowt, A. (2018). YAKE! collection-independent automatic keyword extractor. In *Proceedings of the European Conference on Information Retrieval*.

Carbonell, J. and Goldstein, J. (1998). The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Cardoso, A. B., Martins, B., and Estima, J. (2022). A novel deep learning approach using contextual embeddings for toponym resolution. *ISPRS International Journal of Geo-Information*, 11(1):28.

Clark, K., Luong, M.-T., Le, Q. V., and Manning, C. D. (2020). ELECTRA: Pre-training text encoders as discriminators rather than generators. *arXiv preprint 2003.10555*.

Cliff, A. D. and Ord, J. K. (1981). *Spatial processes: models & applications*. Taylor & Francis.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint 1810.04805*.

Diao, S., Song, Y., and Zhang, T. (2020). Keyphrase generation with cross-document attention. *arXiv preprint 2004.09800*.

Ding, H. and Luo, X. (2021). Attentionrank: Unsupervised keyphrase extraction using self and cross attentions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Florescu, C. and Caragea, C. (2017). A position-biased PageRank algorithm for keyphrase extraction. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Geary, R. C. (1954). The contiguity ratio and statistical mapping. *The incorporated statistician*, 5(3).

Gers, F. A. and Schmidhuber, E. (2001). LSTM recurrent networks learn simple context-free and context-sensitive languages. *IEEE Transactions on Neural Networks*, 12(6):1333–1340.

Getis, A. and Ord, J. K. (2010). The analysis of spatial association by use of distance statistics. In *Perspectives on spatial data analysis*. Springer.

Goldberg, Y. (2017). Neural network methods for natural language processing. *Synthesis Lectures on Human Language Technologies*, 10(1):1–309.

Gritta, M., Pilehvar, M. T., and Collier, N. (2018). Which melbourne? augmenting geocoding with maps. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Halterman, A. (2017). Mordecai: Full Text Geoparsing and Event Geocoding. *The Journal of Open Source Software*, 2(9).

Hammouda, K. M., Matute, D. N., and Kamel, M. S. (2005a). Corephrase: Keyphrase extraction for document clustering. In *International workshop on machine learning and data mining in pattern recognition*. Springer.

Hammouda, K. M., Matute, D. N., and Kamel, M. S. (2005b). Corephrase: Keyphrase extraction for document clustering. In *International workshop on machine learning and data mining in pattern recognition*, pages 265–274. Springer.

Hasan, K. S. and Ng, V. (2014). Automatic keyphrase extraction: A survey of the state of the art. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

He, P., Liu, X., Gao, J., and Chen, W. (2020). DeBERTa: Decoding-enhanced BERT with disentangled attention. *arXiv preprint 2006.03654*.

Hulth, A. (2003a). Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Hulth, A. (2003b). Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Jégou, H. and Chum, O. (2012). Negative evidences and co-occurences in image retrieval: The benefit of PCA and whitening. In *Proceedings of the European Conference on Computer Vision*.

Jones, K. S. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1).

Joshi, M., Chen, D., Liu, Y., Weld, D. S., Zettlemoyer, L., and Levy, O. (2020). SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Kim, S. N., Medelyan, O., Kan, M.-Y., and Baldwin, T. (2010). SemEval-2010 task 5: Automatic keyphrase extraction from scientific articles. In *Proceedings of the International Workshop on Semantic Evaluation*, pages 21–26.

Kingman, D. and Ba, J. (2015). Adam: A method for stochastic optimization. conference paper. In *Proceedings of the International Conference for Learning Representations*.

Krapivin, M., Autaeu, A., and Marchese, M. (2009). Large dataset for keyphrases extraction. *Technical Report DISI-09-055*.

Kulkarni, M., Mahata, D., Arora, R., and Bhowmik, R. (2021). Learning rich representation of keyphrases from text. *arXiv preprint 2112.08547*.

Lau, J. H. and Baldwin, T. (2016). An empirical evaluation of Doc2Vec with practical insights into document embedding generation. *arXiv preprint 1607.05368*.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint 1907.11692*.

Liu, Z., Huang, W., Zheng, Y., and Sun, M. (2010). Automatic keyphrase extraction via topic decomposition. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Marujo, L., Gershman, A., Carbonell, J., Frederking, R., and Neto, J. P. (2013). Supervised topical key phrase extraction of news stories using crowdsourcing, light filtering and co-reference normalization. *arXiv preprint 1306.4886*.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4).

Meng, R., Yuan, X., Wang, T., Zhao, S., Trischler, A., and He, D. (2020). An empirical study on neural keyphrase generation. *arXiv preprint 2009.10229*.

Meng, R., Zhao, S., Han, S., He, D., Brusilovsky, P., and Chi, Y. (2017). Deep keyphrase generation. *arXiv preprint 1704.06879*.

Mihalcea, R. and Tarau, P. (2004). TextRank: Bringing order into text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Moran, P. A. P. (1950). Notes on continuous stochastic phenomena. *Biometrika*, 37(1/2):17–23.

Mu, F., Yu, Z., Wang, L., Wang, Y., Yin, Q., Sun, Y., Liu, L., Ma, T., Tang, J., and Zhou, X. (2020). Keyphrase extraction with span-based feature representations. *arXiv preprint 2002.05407*.

Nguyen, T. D. and Kan, M.-Y. (2007). Keyphrase extraction in scientific publications. In *Proceedings of the International Conference on Asian Digital Libraries*.

Pagliardini, M., Gupta, P., and Jaggi, M. (2017). Unsupervised learning of sentence embeddings using compositional n-gram features. *arXiv preprint 1703.02507*.

Papagiannopoulou, E. and Tsoumakas, G. (2018). Local word vectors guiding keyphrase extraction. *Information Processing & Management*, 54(6):888–902.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. *arXiv preprint 1802.05365*.

Petrov, S., Das, D., and McDonald, R. (2011). A universal part-of-speech tagset. *arXiv preprint 1104.2086*.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, 14(3).

Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151.

Rabby, G., Azad, S., Mahmud, M., Zamli, K. Z., and Rahman, M. M. (2020). TeKET: a tree-based unsupervised keyphrase extraction technique. *Cognitive Computation*, 12(4).

Rabe, M. N. and Staats, C. (2021). Self-attention does not need $o(n^2)$ memory. *arXiv preprint 2112.05682*.

Reimers, N. and Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using siamese BERT-networks. *arXiv preprint 1908.10084*.

Reimers, N. and Gurevych, I. (2020). Making monolingual sentence embeddings multilingual using knowledge distillation. *arXiv preprint arXiv:2004.09813*.

Rose, S., Engel, D., Cramer, N., and Cowley, W. (2010). Automatic keyword extraction from individual documents. *Text mining: applications and theory*, 1.

Rosenblatt, F. (1957). *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint 1609.04747*.

Sajjad, H., Alam, F., Dalvi, F., and Durrani, N. (2021). Effect of post-processing on contextualized word representations. *arXiv preprint 2104.07456*.

Shapira, O., Pasunuru, R., Dagan, I., and Amsterdamer, Y. (2021). Multi-document keyphrase extraction: A literature review and the first dataset. *arXiv prepint 2110.01073*.

Shen, X., Wang, Y., Meng, R., and Shang, J. (2021). Unsupervised Deep Keyphrase Generation. *arXiv preprint 2104.08729*.

Sterckx, L., Demeester, T., Deleu, J., and Develder, C. (2015). Topical word importance for fast keyphrase extraction. In *Proceedings of the International Conference on the World Wide Web*.

Su, J., Cao, J., Liu, W., and Ou, Y. (2021). Whitening sentence representations for better semantics and faster retrieval. *arXiv preprint 2103.15316*.

Sun, S., Xiong, C., Liu, Z., Liu, Z., and Bao, J. (2020). Joint keyphrase chunking and salience ranking with BERT. *arXiv preprint 2004.13639*.

Sun, Y., Qiu, H., Zheng, Y., Wang, Z., and Zhang, C. (2020). SIFRank: A new baseline for unsupervised keyphrase extraction based on pre-trained language model. *IEEE Access*, 8:10896–10906.

Sun, Z., Tang, J., Du, P., Deng, Z.-H., and Nie, J.-Y. (2019). DivGraphPointer: A graph pointer network for extracting diverse keyphrases. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *arXiv preprint 1706.03762*.

Wan, X. and Xiao, J. (2008). Single document keyphrase extraction using neighborhood knowledge. In *Proceedings of the Conference of the Association for the Advancement of Artificial Intelligence*.

Wang, R., Liu, W., and McDonald, C. (2014). Corpus-independent generic keyphrase extraction using word embedding vectors. In *Proceedings of the Software Engineering Research Conference*.

Wang, X., Song, X., Li, B., Guan, Y., and Han, J. (2020). Comprehensive named entity recognition on CORD-19 with distant or weak supervision. *arXiv preprint 2003.12218*.

Wang, Y., Lee, C.-T., Guo, Q., Yin, Z., Zhou, Y., Huang, X., and Qiu, X. (2022). What dense graph do you need for self-attention?

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint 1609.08144*.

Xiong, L., Hu, C., Xiong, C., Campos, D., and Overwijk, A. (2019). Open domain web keyphrase extraction beyond language modeling. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Yuan, X., Wang, T., Meng, R., Thaker, K., Brusilovsky, P., He, D., and Trischler, A. (2018). One size does not fit all: Generating and evaluating variable number of keyphrases. *arXiv preprint 1810.05241*.

Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., et al. (2020). Big bird: Transformers for longer sequences. *Proceedings of the Annual Meeting on Neural Information Processing Systems*.

Zhang, L., Chen, Q., Wang, W., Deng, C., Zhang, S., Li, B., Wang, W., and Cao, X. (2021). MDERank: A masked document embedding rank approach for unsupervised keyphrase extraction.

Zhao, J., Bao, J., Wang, Y., Wu, Y., He, X., and Zhou, B. (2021). Sgg: Learning to select, guide, and generate for keyphrase generation. *arXiv preprint 2105.02544*.