KeyGames: A Game Theoretic Approach to Automatic Keyphrase Extraction

Arnav Saxena[*], **Mudit Mangal**[*] and **Goonjan Jain**
Department of Applied Mathematics, Delhi Technological University
New Delhi, India
{arnavsaxenadtu,muditmangaldtu}@gmail.com, goonjan_jain@hotmail.com

Abstract

In this paper, we introduce two advancements in the automatic keyphrase extraction (AKE) space - KeyGames and pke+. KeyGames is an unsupervised AKE framework that employs the concept of evolutionary game theory and consistent labelling problem to ensure consistent classification of candidates into keyphrase and non-keyphrase. Pke+ is a python based pipeline built on top of the existing pke library to standardize various AKE steps, namely candidate extraction and evaluation, to ensure truly systematic and comparable performance analysis of AKE models. In the experiments section, we compare the performance of KeyGames across three publicly available datasets (Inspec 2001, SemEval 2010, DUC 2001) against the results quoted by the existing state-of-the-art models as well as their performance when reproduced using pke+. The results show that KeyGames outperforms most of the state-of-the-art systems while generalizing better on input documents with different domains and length. Further, pke+'s pre-processing brings out improvement in several other system's quoted performance as well.

1 Introduction

Automatic Keyphrase Extraction (AKE) is a task of identifying important words and phrases that best describe a given text document. AKE makes searching and indexing large digital collections of text feasible and finds use in a variety of Natural Language Processing (NLP) and Information Retrieval (IR) tasks ((Zhang et al., 2004), (Hulth and Megyesi, 2006), (Berend, 2011)). Owing to its widespread use, Keyphrase Extraction has emerged as a fundamental NLP task, improvements in which could lead to improvements in higher-level applications that build upon it (Danesh et al., 2015). Due to this importance, many approaches to keyphrase extraction have been proposed in the literature, majorly along two research lines: supervised and unsupervised (Hasan and Ng, 2014). However, the field faces two specific problems. Firstly, with state-of-the-art performance on keyphrase extraction being much lower than on the many core NLP tasks, there is still a fair amount of improvement possible in this space (Papagiannopoulou and Tsoumakas, 2020). Secondly, there is no effective way to compare and analyze various past AKE systems since most of them use different experimental setups. Different works have used different pre-processing and evaluation to demonstrate their performance. These discrepancies in the past studies make assessing the progress of the AKE space very challenging.

To deal with the first issue, we recognised the need to approach this challenging problem from a fresh perspective. We thus propose a game-theoretic framework for AKE - KeyGames. Players, strategies, and payoffs form an integral part of any game. In KeyGames, the candidate keyphrases act as players having two strategies each: being a keyphrase and being a non-keyphrase. Each phrase plays a game with all the other phrases before deciding upon its strategy. Next, the definition of keyphrases is paraphrased, such that it paves the way for us to develop heuristics which are then formulated into mathematical payoffs. As per our definition, a candidate keyphrase becomes a keyphrase if it satisfies the following two conditions: (a) it should be related to the main themes of the document (b) it should be related to other important phrases

[*] Equal Contribution

Proceedings of the 28th International Conference on Computational Linguistics, pages 2037–2048
Barcelona, Spain (Online), December 8-13, 2020

in the document. To capture (a) in our payoffs, we use semantic relatedness between the phrases and the theme of the document (we call this confidence - higher this metric, more confident the candidate is of becoming a keyphrase). Later, we also bias a candidate's confidence based on its positional information and frequency of occurrence. To capture (b), we use semantic relatedness between the different phrases. The construction of these payoffs is discussed in detail in section 5.4. Once we have the payoffs in place, we apply evolutionary game algorithm to score and rank keyphrases.

Inspired by Tripodi and Pelillo (2017), our reasons for exploring Game theory were two fold; firstly, game theory and NLP have a intuitionistic parallel wherein both have entities interacting with each other thereby influencing each other's behaviour. Second and more important was the ability of game-theoretic frameworks to perform consistent labelling of entities under contextual constraints. We discuss this property in detail in section 3 and further demonstrate consistent labelling in action when we incorporate a preliminary constraint in KeyGames's payoffs to tackle overgeneration[1] in extracted keyphrases (KeyGames+, section 5.4). Through this effort, we reduce the overgeneration error by ~10% points across datasets.

To deal with the second issue, we built pke+[2] as an extension to pke library [3] (Boudin, 2016), Pke is a python based tool kit containing implementation of various existing state-of-the-art AKE systems. Under pke+, we have provided a pipeline with standardized modules for candidate extraction and evaluation. All of pke's implemented systems can be plugged-in to this pipeline to extract keyphrases under similar experimental setup, allowing for a fair comparision (we have implemented KeyGames simply as another system under pke's setting).

To the best of our knowledge, using a game-theoretic approach for AKE has not been proposed before and our work is first such attempt. In fact, the application of game theory to NLP is still in its infancy. We believe that our work would spur additional research in this regard.

2 Game Theory

Game Theory is a branch of applied mathematics that provides predictive power in interactive decision-making situations (Tripodi and Pelillo, 2017). A typical game consists of a finite set of players, $P = (P_1, P_2, .., P_n)$, a set of pure strategies for each player i, $\beta_i = (\beta_1, ..., \beta_n)$ and a utility or payoff function $U_i : \beta_1 \times ..\beta_n \rightarrow R$ which associates a combination of strategies played by various players to the payoffs they receive. Each player aims to settle on a strategy which maximizes its payoff. The concept of payoff, in general, refers to the satisfaction that a player derives from the outcome of a game while interacting with other players. Furthermore, in a two player game, payoffs of a player i when it plays the game with player j can be represented in the payoff matrix A_{ij}, wherein values in each cell of the matrix indicates the payoff player i gets when opting the strategy corresponding to the row represented by that cell, in the event when the j^{th} player chooses the strategy corresponding to the column represented by that cell. Nash equilibria are those strategy profiles which are best response to the strategy of the co-player and no player has the incentive to unilaterally deviate from his strategy, because there is no way to do better.

In Evolutionary Game Theory (EGT), instead of choosing a strategy with certainty as in classical game theory, players choose strategies with certain probability. We thus use the concept of mixed strategy space for each player in EGT. Mixed strategy space is defined as a probability distribution over the pure strategies a player is allowed to play. It is represented as a vector $x_i = (x_i^1, ..., x_i^m)$ where m is the number of pure strategies and each component x_i^h denotes the probability that player i chooses its h^{th} pure strategy. Note that:

$$\sum_{h=1}^{m} x_i^h = 1 \tag{1}$$

[1]Overgeneration errors occur when a system correctly predicts a candidate as a keyphrase because it contains a word that appears frequently in the associated document, but at the same time erroneously outputs other candidates as keyphrases because they contain the same word

[2]https://github.com/mangalm96/KeyGames-pke

[3]https://github.com/boudinfl/pke

An important aspect we observe within the framework of EGT is the inductive learning process wherein the players play the game repeatedly till equilibrium is achieved and keep on updating their beliefs on the state of the game and choose their strategy accordingly (Tripodi and Pelillo, 2017). The steps below are followed to model this learning process:

Step 1: The expected payoff of every pure strategy each player i can obtain (u_i^1) in a single two player game is obtained using the equation:

$$u_i^1 = A_{ij}x_j \tag{2}$$

Step 2: The expected payoff obtained by each player i in a single game (u_i^2) is calculated using:

$$u_i^2 = x_i^T A_{ij} x_j = x_i^T u_i^1 \tag{3}$$

Step 3: Discrete time version of the replicator dynamic equation (Erdem and Pelillo, 2012) is used to update the mixed strategy space of a player i after the end of each iteration as follows:

$$x_i^h(t+1) = x_i^h(t) * u_i^1/u_i^2 \tag{4}$$

where t denotes each t^{th} iteration of a set of two player games.

3 Motivation

An important factor that motivated us to pursue game theoretic models was their ability to solve Consistent Labelling Problem wherein labelling of objects is done while satisfying certain constraints as per the context of the problem (Haralick and Shapiro, 1979). We observed that AKE can be modeled as a Consistent Labelling Problem as well wherein a candidate keyphrase is to be labelled either a keyphrase or a non-keyphrase abiding to certain constraints majorly based on the definition of a keyphrase. This distinguishes our approach from the existing AKE algorithms. A game-theoretic algorithm allows us not only to exploit the contextual information from an input document like various previous works do but also lets us design constraints and heuristics and ensures that classification of candidates into keys and non-keys remains consistent with these heuristics. In fact, it has been shown that, in some cases, using only contextual information without the imposition of constraints can lead to inconsistencies in the assignment of labels to related linguistic entities (Tripodi and Pelillo, 2017). A clear example of such a case is seen while extracting keyphrases from short texts where we have fewer candidates than the number of keyphrases to be extracted. Any other scoring algorithm would classify all the candidates as keys. KeyGames, however, will classify only those candidates into keys that remain consistent with the definition of a keyphrase and will classify the rest as non-keys.

A Consistent Labelling Problem can be defined as tuple $\langle X, L, C\rangle$ where $X = \{X_1, X_2...X_n\}$ is a set of n objects, $L = \{l_1, l_2 \ldots l_m\}$ is a set of m labels and C is a real-valued, $n^2 * m^2$ matrix of compatibility coefficients also called compatibility model, $C = \{C_{ij}(\lambda, \mu) \mid \lambda, \mu \in L\}$. $C_{ij}(\lambda, \mu)$ is the measure of strength of compatibility of two hypothesis "X_i is labeled λ" and "X_j is labeled μ". C is used to impose constraints on labelling so that each label assignment is consistent. The problem is to find a label for each object such that the resulting set of object-label pairs is consistent with the constraints of the compatibility model (Haralick and Shapiro, 1979). A. Miller and Zucker (1991) observed that a consistent labelling problem is equivalent to a polymatrix game. They demonstrated that the objects are equivalent to players in game theory, the labels are strategies each player can opt for, the compatibility model is the payoff matrix designed for the game and a consistent labelling solution is the Nash Equilibrium. In our model, the candidate keyphrases serve as the objects (players), "keyphrase" and "non-keyphrase" serve as the only two labels (strategies) and finally, the mathematical signals formulated using definition of keyphrases serve as the compatibility model (payoff matrix). These signals simply imply that two candidates, one of which adheres to the definition of a keyphrase and another one which doesn't, cannot be assigned the same labels. Clearly, the first one is to be labelled a keyphrase while the second one is to be labelled a non-keyphrase.

4 Related work

Existing studies present keyphrase extraction in two steps. Firstly, a list of candidate keyphrases is generated using heuristic rules such as extracting important noun phrases or extracting phrases that match specific POS (part-of-speech) tags (Bougouin et al., 2013; Mahata et al., 2018). In the next step, ranking mechanisms are often used to select the keyphrases from among the candidates. Both supervised and unsupervised methods are used here. Detailed surveys on these are available (Hasan and Ng, 2014; Papagiannopoulou and Tsoumakas, 2020). Here, we focus on unsupervised approaches due to their many benefits (no training data, domain independence). In unsupervised approaches, many techniques have been explored including, graph-based ranking, clustering and statistical methods. Many statistical based approaches like TFIDF, KPMiner(El-Beltagy and Rafea, 2009) have been proposed. YAKE (Campos et al., 2020) recently used a combination of statistical metrics to capture context information and the spread of the terms in the document.

In graph-based approaches, a document is represented as a graph and word-word interactions are scored using graph centrality measures such as PageRank (Brin and Page, 1998) and its variants. First introduced as TextRank (Mihalcea and Tarau, 2004), words are here represented by nodes and two words are connected by an edge if they co-occur in given window size. Nodes are then ranked using PageRank. Wan and Xiao (2008) extended TextRank to SingleRank by incorporating weights to edges, using number of co-occurences between words as weights. Another graph based model, PositionRank (Florescu and Caragea, 2017) demonstrated the importance of capturing the positions and frequency of a word in a biased PageRank algorithm, preferring words that appeared earlier and more frequently in the text. There have also been attempts at extracting keyphrases that cover all the main topics of the document. This has been done either by using a clustering based approach (Liu et al., 2009) to cluster words into specific topics or by using Latent Dirichlet Allocation(LDA) model (Blei et al., 2003) to obtain the topic distribution (Liu et al., 2010; Sterckx et al., 2015). The LDA based methods require training data, making the systems corpus dependent. Clustering based approaches were first used in a graph by Bougouin et al. (2013). In their proposed system, TopicRank, candidates are clustered depending on the percentage of shared words using hierarchical clustering. A graph is constructed where each node represents a cluster and edges are weighted based on the phrases' offset position in the text. This work was extended to MultiPartiteRank (Boudin, 2018) by encoding topical and positional information within a multipartite graph structure.

With recent advances in deep learning, word embeddings are increasingly being used to study word associations in NLP. Wang et al. (2015) first proposed usage of word embedding vectors for keyphrase extraction when they used the semantic information supplied by pretrained word embeddings to formulate a weighted scheme which is then used on a weighted Page Rank algorithm. Later using a similar approach, Mahata et al. (2018) proposed a corpus dependent method making use of their domain specific trained fasttext embeddings(Bojanowski et al., 2017) for extracting keyphrases from scientific articles. Recently in a departure from graph-based approaches, Bennani-Smires et al. (2018) proposed EmbedRank where they use sentence embeddings (Sent2vec) to represent both the candidate phrases and the document in the same high-dimensional vector space. Their system then ranks the candidate phrases using cosine similarity (semantic relatedness) between the embedding of the candidate phrase and the document embedding.

KeyGames combines many of the approaches discussed above. Here we develop payoffs by capturing word-word and topic-word interactions using fasttext embeddings and later bias these interactions using frequency and positional information of a candidate.

5 Proposed Methodology

5.1 Candidate Extraction

In this work, we use Spacy's[4] noun phrases chunking parser to extract the noun phrases from the input document. Various pre-processing steps (comprising of noun chunk cleaning) are then employed on these chunks to get a list of n-grams (including unigrams). Finally, Spacy's POS tagger is used to remove those singular tokens, which are not Proper Noun or unknown entities (X). Figure 1 illustrates this entire

[4]https://spacy.io

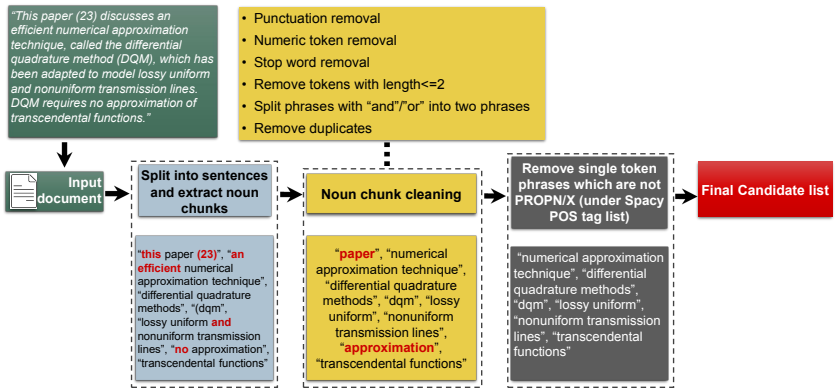

Figure 1: Candidate Extraction Methodology

candidate extraction methodology in detail with an example. This methodology forms a part of the candidate extraction module of the pke+ pipeline.

5.2 Semantic Relatedness using Embeddings

As mentioned in Section 1, we use semantic relatedness in our payoffs. To capture this relatedness, we use the concept of word embeddings. In particular, we use the pre-trained fastText embeddings to get a vector representation of our candidate keyphrases. Semantic relatedness between the candidates is then calculated using the cosine similarity between the phrase vectors. The chosen pre-trained embeddings[5] contain 1 million word vectors trained with subword information on Wikipedia 2017, UMBC web corpus, and statmt.org news dataset. The semantic relatedness between two phrases P_i and P_j is represented as S_{ij}. We choose fastText for its ability to capture both semantic and morphological similarities between words and also the similarity for Out of Vocabulary (OOV) words (Bojanowski et al., 2017).

5.3 Confidence Score

As per our definition, a keyphrase that is more related to the theme of the document is more confident to become a keyphrase. For this preliminary work, we assume that major themes of a document can be inferred from the document title. Thus we simply extract noun phrases from the input document's title (T = $\{t_1, t_2 \ldots t_m\}$), each of which acts as a thematic phrase for our algorithm. The thematic confidence score of the candidates is then calculated using the following equation:

$$Ct_i = S_{i1} + S_{i2} \ldots + S_{im} = \sum_{k=1}^{m} S_{ik} \tag{5}$$

where each ($S_{i1}, S_{i2}, ...S_{im}$) is the semantic relatedness between each keyphrase P_i and all the thematic phrases $\{t_1, t_2 \ldots t_m\}$. These individual scores (Ct_i) are then normalised after division with the highest confidence value (Ct_{max}).

Taking inspiration from previous works (Florescu and Caragea, 2017), we bias the confidence score by assigning more weight to candidate phrases that are found early in a document and are frequent. We weigh each phrase with its inverse position in the candidate list. The frequency of a phrase is taken into account by summing over all its position weights. For eg. for a phrase in the 2nd, 6th and 10th position, the positional confidence is $1/2 + 1/5 + 1/10 = 0.8$. The normalised positional confidence Cp_i, then added to our thematic confidence Ct_i gives us the final confidence score C_i for a candidate phrase P_i.

5.4 Evolutionary Game Algorithm

The candidate keyphrases participate as players ($P_i \in$ P) in our games, having two strategies each - to be a keyphrase (K) or to be a non-keyphrase (NK). Initially, all the players would have equal probabilities (0.5 each) of being a keyphrase or non-keyphrase and hence the initial mixed strategy space(x_i) for each

[5]https://fasttext.cc/docs/en/english-vectors.html

player P_i, would be

$$x_i = [0.5, 0.5]^T \tag{6}$$

As the games proceed, the probabilities in the mixed strategy space adjust appropriately following the inductive learning process after the replicator equation (Equation 4) is applied at the end of each iteration. Also, the two probabilities in the mixed strategy space of every player will always add to 1 (Equation 1). Whenever for any candidate keyphrase P_i, the probability of being a keyphrase is greater than 0.5, we say that the Phrase P_i is tending to be a keyphrase (K). Conversely, whenever the probability is lower than 0.5 (and hence the probability of being a non-keyphrase is higher than 0.5), we say that the Phrase P_i is tending to be a non-keyphrase (NK). Next, we describe the formulation of payoffs that players will receive while deciding upon their strategy (key vs. non key). Note that they will ultimately settle for a strategy for which their average payoff is maximum.

Payoffs 1.0: To construct the payoffs, we first need to understand how the candidates would select their strategies as they interact with one another. Phrase P_i while playing the game with Phrase P_j takes a decision based on the following logics:

1. If P_j is tending to be a keyphrase and also possesses a very high confidence value (C_j), P_i presumes that P_j has a very strong chance to become a keyphrase. Thus, P_i selects its strategy based on its semantic relatedness with P_j. If P_i and P_j have high semantic relatedness (S), then P_i chooses to be a keyphrase (K) as it believes it will have high contextual coherence with the input document just like the semantically similar and confident Phrase P_j. Otherwise, it tends to be a non-keyphrase.

2. If P_j is tending to be a non-keyphrase and has a very low confidence value ($\overline{C_j}$), P_i presumes that P_j will surely become a non-keyphrase and again selects its strategy based on its semantic relatedness with P_j. In this case, if P_i and P_j have high semantic relatedness (S) then P_i chooses to be a non-keyphrase as it gets reasons to believe that it will not be contextually coherent with the document just like the semantically similar and non-confident Phrase P_j. Otherwise, it tends to be a keyphrase.

3. If P_j is tending to be a keyphrase and has a very low confidence value ($\overline{C_j}$) or if P_j is tending to be a non-keyphrase with a very high confidence value (C_j), P_i realizes that P_j is not sure about its decision and hence bases its strategy on its own confidence value. If P_i has high confidence (C_i), then it chooses to be a keyphrase as high confidence indicates that the Phrase P_i is contextually coherent with the document and hence is capable of explaining the main topics of the document.

Payoffs 2.0: In a bid to bias our AKE system against producing over-generated keyphrases, we introduce a factor of fuzzy string match, "F" (based on the Levenshtein distance) on top of Payoff 1.0 built in the last discussion. The core idea is that if two candidates have similar spelling or similar set of tokens, then the system should deter from labelling both of them as keyphrases together. (only the candidate which performs well on other heuristics now stands a chance to become a keyphrase). We use fuzz.token set ratio() function from fuzzywuzzy[6], a python based library to calculate this factor F. A few examples of this function in action-
F(lossy transmission, transmission) = 100
F(pade approx, rational approx) = 84
F(numerical methods,transmission lines) = 34
To bring out this impact of fuzzy match, we tweak the first condition discussed under payoff 1.0 as follows while keeping the other two conditions unchanged:

1. If P_j is tending to be a keyphrase and also possesses a very high confidence value (C_j), P_i presumes that P_j has a very strong chance to become a keyphrase. Thus, P_i selects its strategy based on its semantic relatedness *and its fuzzy similarity* with P_j. If P_i and P_j have high semantic relatedness (S) *and low fuzzy similarity* ($\overline{F}$), then P_i chooses to be a keyphrase as it believes it will have high

[6] 1. https://github.com/seatgeek/fuzzywuzzy

Phrase P_i \ Phrase P_j	Strategies	**KEY**	**NON KEY**
	KEY	Payoff α_1 • P_j has high confidence and P_i is contextually similar to P_j(C_j * S) • P_j has low confidence and P_i has high confidence ($\overline{C_j}$* C_i)	Payoff α_2 • P_j has high confidence and P_i has high confidence (C_j * C_i) • P_j has low confidence and P_i is dissimilar to P_j ($\overline{C_j}$* $\overline{S}$)
	NON KEY	Payoff α_3 • P_j has high confidence and P_i is dissimilar to P_j (C_j*$\overline{S}$) • P_j has low confidence and P_i has low confidence ($\overline{C_j}$* $\overline{C_i}$)	Payoff α_4 • P_j has high confidence and P_i has low confidence (C_j * $\overline{C_i}$) • P_j has low confidence and P_i is similar to P_j ($\overline{C_j}$* S)

Table 1: Strategies of Phrase P_i summarised depending upon strategies of Phrase P_j.

Phrase P_i \ Phrase P_j	Strategies	Key	Non Key
	Key	C_jS +$\overline{C_j}C_i$	C_jC_i+$\overline{C_j}$ $\overline{S}$
	Non Key	$C_j\overline{S}$+ $\overline{C_j}$ $\overline{C_i}$	$C_j\overline{C_i}$ +$\overline{C_j}S$

Table 2: Payoff 1.0 Matrix. C and S denote the Confidence and Similarity indicators respectively.

Phrase P_i \ Phrase P_j	Strategies	Key	Non Key
	Key	$C_jS\overline{F}$ +$\overline{C_j}C_i$	C_jC_i+$\overline{C_j}$ $\overline{S}$
	Non Key	$C_j\overline{S}$+ $\overline{C_j}$ $\overline{C_i}$ + C_jSF	$C_j\overline{C_i}$ +$\overline{C_j}S$

Table 3: Payoff 2.0 Matrix. C,S and F denote the Confidence, Similarity and Fuzzy Match indicators respectively.

contextual coherence with the input document just like the semantically similar and confident Phrase P_j while the low F_{ij} ensures that it will not be causing over-generation. Otherwise, it tends to be a non-keyphrase.

As in any two-by-two game with players having two strategies each, four scenarios are possible when two phrases are involved in our games: (I) both phrases opt for becoming keyphrase (II) both phrases opt for becoming non-keyphrase (III) phrase one opts to become a keyphrase while phrase two doesn't (IV) phrase two opts to become a keyphrase while phrase one does not. For each of these four scenarios, we develop four mathematical signals (α_1, α_2, α_3, α_4) which ultimately become our payoffs. These four signals employ the use of indicators S, $\overline{S}$, C_i, C_j, $\overline{C_i}$, $\overline{C_j}$, F, $\overline{F}$ to model the three decision logics discussed above. We further demonstrate the formulation of Payoffs 1.0 in Table 1.The Payoff 1.0 matrix and Payoff 2.0 matrix are shown in Table 2 and Table 3 respectively.

Lastly, we implement evolutionary game algorithm (Algorithm 1) to score and rank keyphrases. KeyGames (KG) use Payoffs 1.0 and KeyGames+ (KG+) use Payoffs 2.0. Note that with regards to Algorithm 1, equilibrium is the state when the probability of becoming a keyphrase for all candidates remains constant across two consecutive iterations. We simulate this by letting the algorithm run for a large enough number of iterations(i.e. 50).

Algorithm 1 KeyGames Algorithm

Input: Candidates list ck; Confidence score C; Semantic Relatednes S
Init Mixed Strategy Space, x = [ ]
Init Weight of a candidate i, $W(i)$ = 0
for each phrase i in ck **do**
    $x(i)$ = [P(K)=0.5, P(NK)=0.5]^T
while (equilibrium not achieved) **do**
    for each phrase i in ck **do**
        u_1 = $[0, 0]^T$
        u_2 = 0
        for each phrase j in ck **do**
            payoffMatrix$[0, 0]$ = C_jS +$\overline{C_j}C_i$
            payoffMatrix$[0, 1]$ = $C_jC_i + \overline{C_j}$ $\overline{S}$
            payoffMatrix$[1, 0]$ = $C_j\overline{S}$+ $\overline{C_j}$ $\overline{C_i}$
            payoffMatrix$[1, 1]$ = $C_j\overline{C_i} + \overline{C_j}S$
            $temp$ = payoffMatrix $*x(j)$
            $u_2 += x(i)^T * temp$
            $u_1 += temp$
        $x_{(t+1)}(i) = x_t(i) * u_1/u_2$
        $ck[i].W(i) += x_{(t+1)}(i)[0]$
sort(ck) by descending W and extract topn keys
Output: List of Topn keyphrases

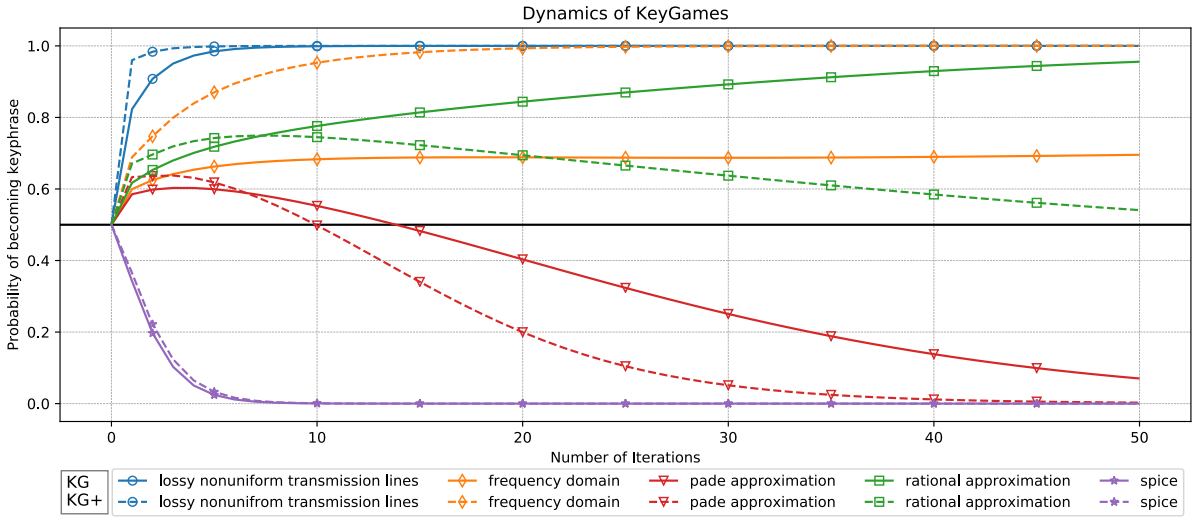

Figure 2: Using KG, 'lossy nonuniform transmission lines', 'rational approximation' and 'frequency domain', all end up in the extracted keyphrase list at the equilibrium position, but 'rational approximation' and 'frequency domain' don't make it to the top 10 keyphrase list as they achieve equilibrium at a very slow rate. Under KG+, however, overgenerated phrases like 'pade approximation' and 'rational approximation' both move towards becoming non-key at a faster pace then their KG counterparts. Also note that KG+, while punishing over-generated candidates, rewards non-overgenerated phrases like 'frequency domain' and it ends up as top 10 keyphrase (partially matching with a ground truth keyphrase) thereby increasing diversity in the KG+ extracted keyphrase list.

Title: Accurate modeling of **lossy nonuniform transmission lines** by using **differential quadrature methods**.
Input Text: This paper discusses an efficient **numerical approximation technique**, called the **differential quadrature method** (DQM), which has been adapted to model lossy uniform and nonuniform transmission lines... Using the DQM, the frequency-domain Telegrapher's partial differential equations for transmission lines... DQM reduces interconnects into **multiport models** whose port voltages and currents are related by rational formulas in the frequency domain. The rationalization process in DQM is comparable with the Pade approximation of asymptotic waveform evaluation (AWE) applied to transmission lines. AWE employs a complex moment-matching process to obtain rational approximation... Due to global sampling of points in the DQM approximation, it requires far fewer grid points in order to build accurate discrete models than other numerical methods do. The DQM-based time-domain model can be readily integrated in a circuit simulator like SPICE.
Assigned Keyphrases: lossy nonuniform transmission lines, differential quadrature method, numerical approximation technique, frequency-domain Telegrapher PDE, partial differential equations, multiport models, multiconductor transmission lines, rationalization process
KeyGames Keyphrases: lossy nonuniform transmission lines, differential quadrature methods, numerical approximation technique, transmission lines, multiport models, accurate discrete models, numerical methods, rational formulas
KeyGames+ Keyphrases: lossy nonuniform transmission lines, differential quadrature methods, numerical, approximation technique, multiport models, global sampling, small set, frequency domain, accurate discrete models

Table 4: An example of keyphrases extracted using KG and KG+ from a sample Inspec article (397.txt)

5.5 Keyphrase Ranking:

Keyphrases are ranked in the order in which they achieve equilibrium. We measure this rate for a candidate by summing over its probability of becoming a keyphrase at the end of each iteration (Algorithm 1: Line 18). The earlier a candidate becomes a keyphrase, the faster its probability of becoming a keyphrase (P(K)) reaches one. Hence, its summation of P(K) across all iterations would be higher. Table 4 shows ranked keyphrases extracted from a sample Inspec article. Figure 2 illustrates the dynamics of a typical KeyGame for selected phrases of the sample.

6 Experimentation and Results

6.1 Datasets and Evaluation Metrics

To evaluate the performance of KeyGames, we carry out experiments on three publicly available English datasets. The first dataset, Inspec (Hulth, 2003), consists of short documents from scientific journal

Method	Inspec				SemEval				DUC			
	P	R	F_1	$F_1{}^{o}$	P	R	F_1	$F_1{}^{o}$	P	R	F_1	$F_1{}^{o}$
SingleRank	40.31	36.41	38.26	29.69*	6.8	4.71	5.57	1.99*	22.48	27.47	24.73	27.2
TopicRank	41.02	34.69	37.59	27.9	16.1	11.16	13.18	12.1	19.31	23.6	21.24	21.39*
PositionRank	40.56	36.61	38.48	27.49*	10.6	7.35	8.68	7.85*	23.33	28.52	25.67	27.01*
MultipartiteRank	41.15	37.2	39.07	30.6	18.2	12.61	**14.9**	14.5	22.21	27.19	24.45	23.41*
EmbedRank	41.44	37.43	39.33	35.8*	6.2	4.29	5.08	3.58*	22.71	27.71	24.96	29.4*
YAKE	36.7	32.8	34.63	31.6	0.6	0.4	0.5	12.25*	3.95	4.84	4.35	14.18*
KeyGames(KG)	43.38	38.79	**40.48**†	-	17.57	12.13	14.35	-	25.71	31.43	**28.28**†	-
KeyGames+(KG+)	43.15	38.73	40.87	-	17.17	11.85	14.02	-	23.89	29.21	26.28	-

Table 5: Comparison of KG and KG+ with some unsupervised systems. Precision (P), Recall (R), and F-score (F_1) for Top 10 Keyphrases (using pke+) are reported. † indicates significance at the 0.01 level using Student's t-test over the next best system. For a fair comparison, the original reported F-scores ($F_1{}^{o}$) of the existing systems are provided. Missing results are accounted for by reimplementation using pke (indicated by *)

abstracts. Like previous works (Mihalcea and Tarau, 2004; Bougouin et al., 2013), we use the test set of 500 documents and the uncontrolled set of annotated keyphrases as ground truth for our analysis. The second dataset, SemEval 2010 (Kim et al., 2010) contains ACM full length papers. In our experiments, we use the 100 documents from the test set and the combined set of annotated keyphrases. The third dataset, DUC 2001 (Wan and Xiao, 2008), is a collection of 308 medium length news articles. Similar to candidate extraction, we use common evaluation to transparently compare our approach with other systems. We compute exact matched true positives between the assigned and extracted keys (both stemmed) and evaluate performances in terms of the micro-averaged Precision, Recall, and F_1 measure for top 10 keyphrases.

6.2 Performance Comparision

In Table 5, we compare KG (with Payoffs 1.0) and KG+ (with Payoffs 2.0) to existing unsupervised state-of-the-art systems - SingleRank (Wan and Xiao, 2008), TopicRank (Bougouin et al., 2013), PositionRank (Florescu and Caragea, 2017), MultipartiteRank (Boudin, 2018), EmbedRank[7] (Bennani-Smires et al., 2018) and YAKE[8] (Campos et al., 2020). Direct comparisons are done on the scores (P,R,F_1) obtained using the pke+ implementation. We also mention the original scores($F_1{}^{o}$) as reported in the respective papers to compare it with our pke+ implementations.

We observe that KeyGames consistently outperforms most of the existing systems across the three datasets, each with different document length, covering two different domains. Moreover, the pke+ pipeline (more specifically, the pre-processing) helps most systems improve/retain performances as originally quoted. This further indicates the effectiveness of the new candidate extraction. Among existing systems, EmbedRank performs admirably on short texts (Inspec/DUC) but fails to replicate it on long texts (as also reported by the authors themselves). Only MPRank gives consistent results across the datasets but does not match the effectiveness of KeyGames, especially on Inspec and DUC, where the improvements are statistically significant while the difference on SemEval is statistically insignificant. The performance on longer texts like SemEval has historically been lower, often attributed to the hypothesis that it is more difficult to extract keyphrases correctly from longer documents because of a much bigger search space (Hasan and Ng, 2014).

Compared to KG, the KG+ system, which incorporates the over-generation constraint, shows a dip in performance even though it results in keys with less overgeneration (Table 6) (and hence more diversity). We calculate this over-generation percentage as the the sum of the number of phrases having more than 75% fuzzy match with any other keyphrase in the extracted/assigned keyphrase list as

Dataset	Inspec	SemEval	DUC
Assigned Keys	29.59%	46.99%	22.71%
KG	35.49%	63.4%	30.68%
KG+	26.12%	53.5%	20.92%

Table 6: Over-generation in Assigned keys & Extracted Keys of KG and KG+

[7]EmbedRank's reported scores are macro-averaged. Hence, we mention their reimplemented scores

[8]YAKE's original reported score are based on the entire SemEval dataset (and not just the test set)

a share of the total number of keyphrases extracted/assigned across the entire dataset -

$$og\% = 100\,\frac{\sum_{i \in N} \text{count(overgenerated keys)}}{\sum_{i \in N} \text{count(total keys)}} \tag{7}$$

where *overgenerated keys* are keyphrases with more than *75% fuzzy match(F)* score with any other key in the assigned/extracted keyphrase list of doc i, *total keys* are the total number of assigned/extracted keyphrases of doc i and N is the number of documents in a dataset.

Bennani-Smires et al. (2018) also observed such a dip in results on these diverse and less overlapping keys in their experiments. It can perhaps be explained by high over-generation present in the assigned keys themselves (Table 6). In Table 5 we highlight the results of the KG system and not KG+ because of its more consistent performance across datasets.

7 Conclusion and Future Work

In this paper, we discussed our work on a novel, unsupervised game-theoretic approach to automatic keyphrase extraction after modeling the task as a consistent labelling problem. We paraphrased the definition of a keyphrase, conceptualized logics to model the same, and ultimately designed keyphrase games by quantifying these logics into mathematical payoffs. For carrying out experiments, we used pke to construct a new pipeline pke+, standardizing candidate extraction, and evaluation steps across different systems. Our proposed method consistently outperforms most of the existing systems on three datasets with different domains and lengths, indicating the potential of game theory as an alternative. In the future, we plan on working with Infection and Immunization Dynamics (InImDyn) (Rota Bulò et al., 2011), which approximately simulates replicator dynamics in linear time, thereby reducing time-complexity of KeyGames. We plan to experiment with different heuristics in our payoffs as well. We would also like to explore other methods like document embedding to extract the theme of the document.

References

Douglas A. Miller and Steven Zucker. 1991. Copositive-plus lemke algorithm solves polymatrix games. *Operations Research Letters*, 10:285–290, 07.

Kamil Bennani-Smires, Claudiu Musat, Andreea Hossmann, Michael Baeriswyl, and Martin Jaggi. 2018. Simple unsupervised keyphrase extraction using sentence embeddings. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 221–229. Association for Computational Linguistics.

Gábor Berend. 2011. Opinion expression mining by exploiting keyphrase extraction. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1162–1170. Asian Federation of Natural Language Processing.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Florian Boudin. 2016. pke: an open source python-based keyphrase extraction toolkit. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, pages 69–73, Osaka, Japan, December.

Florian Boudin. 2018. Unsupervised keyphrase extraction with multipartite graphs. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 667–672. Association for Computational Linguistics.

Adrien Bougouin, Florian Boudin, and Béatrice Daille. 2013. Topicrank: Graph-based topic ranking for keyphrase extraction. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 543–551. Asian Federation of Natural Language Processing.

S. Brin and L. Page. 1998. The anatomy of a large-scale hypertextual web search engine. In *Seventh International World-Wide Web Conference (WWW 1998)*.

Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Mário Jorge, Celia Nunes, and Adam Jatowt. 2020. Yake! keyword extraction from single documents using multiple local features. *Inf. Sci.*, 509:257–289.

Soheil Danesh, Tamara Sumner, and James H. Martin. 2015. Sgrank: Combining statistical and graphical methods to improve the state of the art in unsupervised keyphrase extraction. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, pages 117–126. Association for Computational Linguistics.

Samhaa R. El-Beltagy and Ahmed A. Rafea. 2009. Kp-miner: A keyphrase extraction system for english and arabic documents. *Inf. Syst.*

Aykut Erdem and Marcello Pelillo. 2012. Graph transduction as a noncooperative game. *Neural Computation*, 24(3):700–723.

Corina Florescu and Cornelia Caragea. 2017. PositionRank: An unsupervised approach to keyphrase extraction from scholarly documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vancouver, Canada. Association for Computational Linguistics.

R. M. Haralick and L. G. Shapiro. 1979. The consistent labeling problem: Part i. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):173–184.

Kazi Saidul Hasan and Vincent Ng. 2014. Automatic keyphrase extraction: A survey of the state of the art. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1262–1273. Association for Computational Linguistics.

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

Anette Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, EMNLP '03, pages 216–223, Stroudsburg, PA, USA. Association for Computational Linguistics.

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

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

Zhiyuan Liu, Wenyi Huang, Yabin Zheng, and Maosong Sun. 2010. Automatic keyphrase extraction via topic decomposition. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 366–376. Association for Computational Linguistics.

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

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*.

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

Samuel Rota Bulò, Marcello Pelillo, and Immanuel Bomze. 2011. Graph-based quadratic optimization: A fast evolutionary approach. *Computer Vision and Image Understanding*.

Lucas Sterckx, Thomas Demeester, Johannes Deleu, and Chris Develder. 2015. Topical word importance for fast keyphrase extraction. In *Proceedings of the 24th International Conference on World Wide Web*, page 121–122. Association for Computing Machinery.

Rocco Tripodi and Marcello Pelillo. 2017. A game-theoretic approach to word sense disambiguation. *Computational Linguistics*, 43(1):31–70.

Xiaojun Wan and Jianguo Xiao. 2008. Single document keyphrase extraction using neighborhood knowledge. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2*, AAAI'08, pages 855–860. AAAI Press.

Rui Wang, Wei Liu, and Chris McDonald. 2015. Using word embeddings to enhance keyword identification for scientific publications. In *Australasian Database Conference*, pages 257–268. Springer.

Yongzheng Zhang, Nur Zincir-Heywood, and Evangelos Milios. 2004. World wide web site summarization. *Web Intelli. and Agent Sys.*, 2(1):39–53, January.