# Incorporating Multimodal Information in Open-Domain Web Keyphrase Extraction

**Yansen Wang***, **Zhen Fan***, **Carolyn P. Rosé**
Language Technologies Institute, Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh PA, 15213
{yansenwa, zhenfan, cp3a}@andrew.cmu.edu

## Abstract

Open-domain Keyphrase extraction (KPE) on the Web is a fundamental yet complex NLP task with a wide range of practical applications within the field of Information Retrieval. In contrast to other document types, web page designs are intended for easy navigation and information finding. Effective designs encode within the layout and formatting signals that point to where the important information can be found. In this work, we propose a modeling approach that leverages these multi-modal signals to aid in the KPE task. In particular, we leverage both lexical and visual features (e.g., size, font, position) at the micro-level to enable effective strategy induction, and meta-level features that describe pages at a macro-level to aid in strategy selection. Our evaluation demonstrates that a combination of effective strategy induction and strategy selection within this approach for the KPE task outperforms state-of-the-art models. A qualitative post-hoc analysis illustrates how these features function within the model.

## 1 Introduction

We present a novel multi-modal approach to **K**ey**P**hrase **E**xtraction (KPE), which is the task of automatically extracting salient phrases from a given document. The KPE task is a foundational task, which plays a facilitating role in many Information Retrieval (IR) tasks, including classification, summarization, and document indexing (Hasan and Ng, 2014). Specifically, the KPE task requires accurate selection of the phrases that best capture the web document's topic. Well-performing approaches take advantage of document structure and entity co-occurrences.

Over the history of work in this area, there have been a variety of benchmarks (Medelyan and Witten, 2002; Nguyen and Kan, 2007; Wan and Xiao, 2008; Meng et al., 2017) and an equally wide variety of both non-neural (Grineva et al., 2009; Liu et al., 2009, 2010) and neural modeling approaches (Meng et al., 2017; Zhang et al., 2017; Chen et al., 2018). The earliest KPE approaches were mainly limited to domain-specific keyphrase extraction. The recent release of **OpenKP** (Xiong et al., 2019), a large-scale feature-rich dataset specifically developed for open-domain web-page keyphrase extraction, has encouraged further research related to the KPE task. A novel characteristic of this data set is the inclusion of features related to visual properties.

Visual properties of words and web page layout offer a KPE model utility in at least two respects. First, **micro-level features** operating at the word level, including lexical features as well as features related to size, font, color, and position of words, signal relative importance of words within extended texts. Intuitively, texts that are highlighted with colored, bold or bigger font or placed in more obvious places within a web page are more likely to be important, and should correspondingly be given higher probability as a keyphrase. Next, **macro-level features** describing the layout and type of page (e.g., News, Storefront, etc.) are related to the distribution of keyphrases. For example, for a news or blog website, the title or the first paragraph of its main content are likely locations for finding keyphrases, while for an Indexing page, the important phrases will more likely be found in lists. Important information may also be listed underneath or beside pictures.

Based on these insights, we propose a multi-modal framework, **S**trategy-based **M**ultimodal **AR**chi**T**ecture for **K**ey**P**hrase **E**xtraction (SMART-KPE), addressing the web KPE task in two steps: **Multimodal Strategy Induction** to apply specific extraction tactics with a refined use of micro-level features and **Strategy Selection**

* Equally contributed.



*Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 1790–1800,
November 16–20, 2020. ©2020 Association for Computational Linguistics

to choose results from different tactics using macro-level features. In our evaluation, we compare SMART-KPE with several state-of-the-art baselines, where SMART-KPE shows its better ability to locate and extract keyphrases. We offer post-hoc case studies and ablation studies to illustrate model strengths and weaknesses. In addition to the improvement over SOTA baselines for the KPE task, to the best of our knowledge, Strategy-based Multimodal Architecture for Keyphrase Extraction is the most comprehensive treatment of multimodality in open-domain KPE.

## 2 Related Work

### 2.1 Development of Open-domain Web Keyphrase Extraction

Originally, the concept *keyphrase* was first used by authors of scientific papers when they indicated by hand a few phrases they decided best summarized their paper (Çano and Bojar, 2019). The first corpora for automated keyphrase extraction were likewise assembled out of publications from scientific fields including technical reports (Witten et al., 1999), paper abstracts (Hulth, 2003), and scientific papers (Nguyen and Kan, 2007; Medelyan et al., 2009; Kim et al., 2010). To this day, scientific publications still serve as a fundamental fixed-domain benchmark for neural KPE methods (Meng et al., 2017; Alzaidy et al., 2019; Sahrawat et al., 2019) due to the availability of ample data of this kind. However, experiments have revealed that KPE methods trained directly on such corpora do not generalize well to other web-related genres or other types of documents (Chen et al., 2018; Xiong et al., 2019), where there may be far more heterogeneity in topics, content and structure, and there may be more variation in terms of where a key phrase may appear.

Past researchers have collected corpora for KPE in Internet and social media environments, including web pages (Yih et al., 2006; Hammouda et al., 2005), blogs (Grineva et al., 2009), email (Dredze et al., 2008), news articles (Wan and Xiao, 2008; Hulth and Megyesi, 2006) and live chats (Kim and Baldwin, 2012), but most of these existing corpora fall prey to similar problems with respect to robust model training for neural models due to data sparsity and lack of representativeness in topic distribution. The recently released OpenKP (Xiong et al., 2019) is the first large-scale KPE dataset with a broad distribution of topic domains. This recent dataset facilitates work on model generalization and the opportunity to develop nuanced models that can adapt their performance based on the type of document they are applied to. This property of the dataset has inspired our proposed method where strategies are selected based on the detected type of document using macro-level features.

### 2.2 Neural Keyphrase Extraction Approaches

The earliest neural KPE models treat the KPE task as a standard encoder-decoder task, which first creates an encoding of the input using an RNN (Meng et al., 2017) or CNN (Zhang et al., 2017), and then decodes the predicted keyphrases. These early approaches were strictly limited to textual data representations.

The release of OpenKP (Xiong et al., 2019) has introduced opportunities for research in multimodal KPE. OpenKP is now a recently added branch of MS-MARCO(Nguyen et al., 2016), with a public leaderboard for the KPE task held by Microsoft[1]. Built from Web data, it serves as the first large-scale benchmark for open-domain neural keyphrase extraction. In addition to providing the raw *text* of each document, OpenKP also includes various *visual features* associated with each text term, such as position, size, font, etc. Along with OpenKP, Xiong et al. (2019) also proposed BLING-KPE, the first neural model baseline for open-domain keyphrase extraction using visual features along with text. BLING-KPE first generates a hybrid embedding for each term by concatenating: (1) the ELMo (Peters et al., 2018) representation of the term, (2) standard sinusoidal position embedding (Vaswani et al., 2017), and (3) 18 of the 20 **visual features** of the term available in the OpenKP dataset. It models n-grams using multiple CNNs, and utilizes a Transformer (Vaswani et al., 2017) layer and feed-forward layer for scoring. This approach represents the first attempt at multimodal KPE.

Recently, Sun et al. (2020) achieved greater success on the OpenKP task by modeling keyphrase extraction as multiple traditional text tasks, including sequence labeling, chunking, salience ranking, etc. From this work we adopt the idea of modeling KPE as a sequence labeling task, where one of five tags is assigned to each document term: N**O**n

[1] https://microsoft.github.io/msmarco/

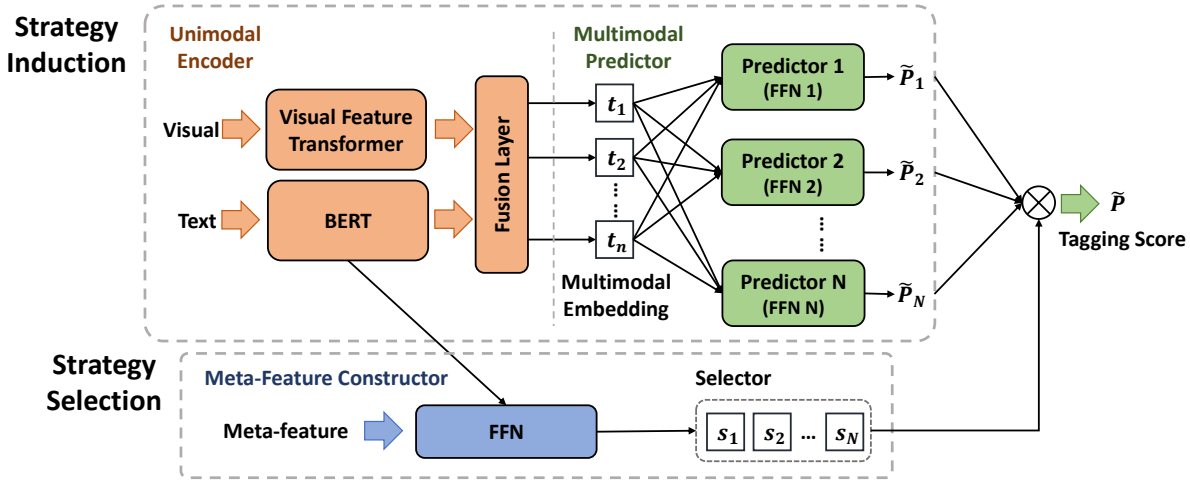



Figure 1: The SMART-KPE model architecture. Arrow between *BERT* and *FFN* represents the overall textual representation $\vec{w}_{CLS}$.

keyphrase, **B**egin word of the keyphrase, M**I**ddle word of the keyphrase, **E**nd word of the keyphrase, and **U**ni-word keyphrase.

However, despite multiple recent efforts on this newly proposed dataset, published work so far in multimodal KPE has either omitted available features (Sun et al., 2020), or has adopted a brute force approach to feature encoding (direct concatenation of raw features) (Xiong et al., 2019). Therefore, in this work we strive for a more nuanced approach to leveraging available features for multimodal KPE and offer a uniquely comprehensive approach.

# 3 Model

## 3.1 Task Definition

Here we formalize the keyphrase extraction task (KPE) under the web page setting: Given a document $D = \{W, V, M\}$, where $W = \{w_1, w_2, \ldots, w_n\}$ are the text terms of the web page with length $n$, $V = \{v_1, v_2, \ldots, v_n\}$ are the respective visual features of each term and $M$ is the set of macro-level meta-features describing the document, we aim to find the set of word sub-sequences $S = \{S_1, S_2, ..., S_K\}$ where $S_i = \{w_{j_i}, \ldots, w_{j_i+k_i-1}\}$ are the extracted keyphrases from the document text that are the most salient and best represent the keypoints of the document.

We adopt the method used by Sun et al. (2020)(**BERT2Tag**), where KPE is modeled as a **sequence labeling** problem on text terms $W$. For a given document, each term is assigned one of five labels: namely, $\{O, B, I, E, U\}$, which represent n**O**n keyphrase, **B**egin word of the keyphrase, M**i**ddle word of the keyphrase, **E**nd word of the keyphrase and **U**ni-word keyphrase respectively. In the following sections, we elaborate on the details of training and testing for this sequence labeling task.

## 3.2 Model Structure

We divide the web KPE task into two steps: **multimodal strategy induction**, where specific tactics are applied to micro-level multimodal features, and **strategy selection**, where macro-level features are used to choose the best available strategy matching the form of the current page. We devise **S**trategy-based **M**ultimodal **AR**chi**T**ecture for **K**ey**P**hrase **E**xtraction (SMART-KPE), a multimodal framework that extends the sequence labeling foundation.

Figure 1 illustrates the architecture of SMART-KPE. Specifically, it incorporates three components: Unimodal Encoder, Multimodal Predictor and Meta-Feature Constructor.

### 3.2.1 Multimodal Strategy Induction

In this step, strategies are learned from selected subsets of available micro-level features for the KPE task. Specifically, our model first generates contextualized embeddings for both textual and visual features separately in the Unimodal Encoder. These embeddings are fused and subsequently fed into several distinct strategy-specific sequence labeling networks in the Multimodal Predictor, each generating a separate probability distribution of each token's possible tag.

The micro-level features used during this step are provided by the dataset, specified in Section 4.1.

**Unimodal Encoder** The Unimodal Encoder is designed to build each term's multimodal representation based on both textual and visual modalities. We use a pretrained uncased BERT model (Devlin et al., 2018) to generate the contextualize term embeddings. Similarly, we apply a separate transformer (Vaswani et al., 2017) to the visual features. This can be formulated as:

$$\vec{w}_{[CLS]}, \vec{w}_1, ..., \vec{w}_n = \text{BERT}(W), \quad (1)$$

$$\vec{v}_1, ..., \vec{v}_n = \text{Vis\_Transformer}(V). \quad (2)$$

where $\vec{w}_i$ and $\vec{v}_i$ are the contextualized text and visual embeddings of word $w_i$ respectively. $\vec{w}_{CLS}$ is the BERT representation of the [CLS] token, which we use later as a representation of the whole document in the Meta-Feature Constructor.

We motivate our handling of visual features using two intuitions. First, similar to text, visual features can be expected to achieve completely different visual effects depending on the page context: e.g., a word with font size 20 might be commonly used in one web page while it could mark the largest word in another. Second, the behavior and characteristics of visual and text modality features are different from one another. Therefore the first-step self-attention should be modeled in separate networks for text and visual features.

At the end of the Unimodal Encoder, textual and visual embeddings are concatenated as the final representation of a term and fused globally with another transformer:

$$\vec{e}_i = \vec{w}_i || \vec{v}_i, \quad (3)$$

$$\vec{t}_1, ..., \vec{t}_n = \text{Transformer}(\vec{e}_1, ..., \vec{e}_n). \quad (4)$$

**Multimodal Predictor** The Multimodal Predictor consists of $N$ label predictors representing $N$ different extraction strategies. Each predictor takes the textual and visual embeddings given by Eq. 4 and generates a tag score distribution $\mathcal{P}_{i,k}$ for the term $i$ independently using a 2-layer feed-forward network:

$$\mathcal{P}_{i,k} = softmax(\text{FFN}_k(\vec{t}_i)). \quad (5)$$

Note that all the strategies here are defined implicitly, which means we do not introduce any human assigned prior related to identification of page type. Rather, we allow latent information on page types to emerge and differentiate between strategies as the network's learning process finds through optimization. This avoids a time-consuming labeling process and also mitigates the risk that types that seem intuitive from a cognitive standpoint might nevertheless not offer utility within the optimization.

### 3.2.2 Strategy Selection

In order to obtain the overall sequence labeling result, the Meta-Feature Constructor encodes the macro-level meta-features to perform a weighted selection upon predictors in the Multimodal Predictor. In this work, we use the following kinds of macro-level meta-features:

**Whole-Text Representation** Here we use $\vec{w}_{CLS}$, the representation of the [CLS] token of our BERT model in the Unimodal Encoder as an estimated overall textual representation of both the website document and the title.

**Snapshot** The visual embedding extracted from the snapshot of the original web page using Resnet-152 (He et al., 2016). This is a novel aspect of our work that offers the model the opportunity to identify characteristics of the overall layout and appearance of the web page.

This architecture is designed to flexibly include more meta-features given extra data or resources. All the meta-features are concatenated within a meta-feature embedding $\vec{m}$, after which a feed-forward network is applied to generate a normalized $N$-dimensional selector vector:

$$\vec{S} = (s_1, ..., s_N) = softmax(\text{FFN}(\vec{m})), \quad (6)$$

$$s_k = \underset{k}{softmax}(s'_1, ..., s'_N). \quad (7)$$

We use the selector vector $s_k$ as the weights for the $N$ sequence label predictors in Multimodal Predictor to generate the overall probability that each tag is assigned to a term.

$$\tilde{\mathcal{P}}_i = \sum_{k=1}^{N} s_k \mathcal{P}_{i,k}, \quad (8)$$

$$p_{i,T} = \tilde{\mathcal{P}}_i(T), \quad (9)$$

where $p_{i,T}$ is the probability of term $i$ being labeled as tag $T \in \{O, B, I, E, U\}$, as described in Section 3.1.

### 3.3 Training and Keyphrase Prediction

The SMART-KPE model is trained in an end-to-end way using term-wise cross-entropy loss:

$$loss = -\sum_{i=1}^{n} \log \tilde{\mathcal{P}}_i(y_i), \tag{10}$$

where $y_i$ is the correct tagging label for word $w_i$ according to its relation to the golden keyphrases.

For keyphrase prediction, after SMART-KPE predicts the tag probability distribution for each term, the score of each phrase is calculated as follows:

$$Score(S_{i:i+k}) = \min_{j \in [i,i+k)} p_{j,T'_j},$$

where $T'_j$ is the corresponding tag for term $w_j$ given $S_{i:i+k}$ is a keyphrase, and $p_{i,T'_i}$ is the predicted probability of $w_j$ being labeled as $T'_j$. Min-pooling is used here to enable keyphrases to be treated in a comparable fashion regardless of length.

## 4 Experimental Methodology

### 4.1 Dataset

We set OpenKP as the main dataset for our task. OpenKP consists of ∼150K documents sampled from the Bing search engine, within which neither the domain nor type of original web pages are restricted.

For each document, the following information is given:

- **URL:** The link to the respective web page.
- **Text:** Cleaned body text of a document.
- **Visual DOM features:** A set of vectors representing the visual characteristics of text terms, listed in Table 1. For each term, features describing itself and its parent block in the DOM Tree are included.

The keyphrases for each document in the given dataset were labeled by expert annotators, with each document assigned 1-3 keyphrases. As a requirement, all the keyphrases were ones that appeared in the original document. The detailed statistics of OpenKP are displayed in Table 2.

In the original dataset, no meta-features are provided except the website URL. We downloaded the title of each website and concatenated the titles with the cleaned body text as the text input of our model. We also took a snapshot in the Google Chrome Browser for the web page with display size 600 × 800 and all the element shrunk to 50% in order to get a more comprehensive view of the website. [2]

| Feature Name | Dimension |
|---|---|
| Block Position | 2×2 |
| Block Size | 2×2 |
| Font Size | 1×2 |
| Is Bold | 1×2 |
| Is Heading Element | 1×2 |
| Is Block Element | 1×2 |
| Is Inline Element | 1×2 |
| Is Leaf Element | 1×2 |

Table 1: Visual features in OpenKP

| Property | Value |
|---|---|
| # of Training Docs | 134,894 |
| # of Validation Docs | 6,616 |
| # of Test Docs | 6,614 |
| Average Doc Length | 900.4 |
| Average KPs per Doc | 1.8 |
| Average KP Length | 2.0 |
| Doc Vocab size | 1.5M |
| KP vocab size | 62K |

Table 2: Statistics of OpenKP

### 4.2 Baselines and Evaluation Metrics

We compare SMART-KPE with the following baselines on the OpenKP dataset:

**BLING-KPE** (Xiong et al., 2019) *Beyond Language UnderstandING KeyPhrase Extraction*, the official baseline proposed with the OpenKP dataset. It concatenates ELMo, visual and positional features and uses a CNN structure to generate k-gram phrase embeddings for binary prediction.

**BERT2{Span,Chunk,Tag,Rank,Joint}** (Sun et al., 2020) A set of models applying different keyphrase extraction methods (Span prediction, Chunking, Tagging, Ranking and Joint method) upon BERT and its variants. Note that for fair comparison, we report all results obtained by BERT-based baseline models and compare them

[2] Data and codes are available at https://github.com/victorywys/SMART-KPE.

| Model | P@1 | R@1 | F@1 | P@3 | R@3 | F@3 | P@5 | R@5 | F@5 |
|---|---|---|---|---|---|---|---|---|---|
| BLING-KPE | 0.404 | 0.220 | 0.285* | 0.248 | 0.390 | 0.303* | 0.188 | 0.481 | 0.270* |
| BERT2Span** | 0.480 | 0.260 | 0.324 | 0.287 | 0.443 | 0.335 | 0.211 | 0.533 | 0.293 |
| BERT2Chunk** | 0.518 | 0.280 | 0.349 | 0.313 | 0.480 | 0.365 | 0.225 | 0.564 | 0.312 |
| BERT2Tag** | 0.516 | 0.280 | 0.348 | 0.316 | 0.485 | 0.368 | 0.230 | 0.578 | 0.319 |
| BERT2Rank** | 0.517 | 0.280 | 0.348 | 0.321 | 0.495 | 0.375 | 0.235 | 0.590 | 0.326 |
| BERT2Joint** | 0.520 | 0.282 | 0.350 | 0.324 | 0.498 | 0.378 | 0.235 | 0.590 | 0.326 |
| SMART-KPE-Skeleton*** | 0.560 | 0.300 | 0.375 | 0.335 | 0.511 | 0.390 | 0.241 | 0.603 | 0.334 |
| SMART-KPE-Micro*** | 0.563 | **0.305** | **0.380** | 0.342 | 0.524 | 0.399 | 0.244 | 0.613 | 0.339 |
| SMART-KPE-Macro | 0.565 | **0.305** | **0.380** | 0.340 | 0.519 | 0.395 | 0.244 | 0.611 | 0.338 |
| SMART-KPE-Full | **0.567** | 0.304 | **0.380** | **0.344** | **0.525** | **0.401** | **0.248** | **0.620** | **0.344** |
| RoBERTa2Joint** | 0.546 | 0.294 | 0.366 | 0.336 | 0.516 | 0.392 | 0.244 | 0.611 | 0.338 |
| SMART-KPE+R2J | **0.567** | **0.307** | **0.381** | **0.348** | **0.532** | **0.405** | **0.250** | **0.625** | **0.347** |

* These numbers are not included in the original paper and are estimated with Precision and Recall.
** These results are re-evaluated on the updated official dataset.
*** Since no macro-level meta-features are used in these model variants, the number of Predictors is set to 1.

Table 3: Model performances on the OpenKP development set. F1@3 is the main metric for this task. SMART-KPE-Full is the complete model and Skeleton, Micro and Macro denote for ablations where no additional features, only micro-level visual features, or only macro-level features are introduced respectively. SMART-KPE+R2J is our complete model equipped with the state-of-the-art extracting method (RoBERTa2Joint).

with the basic version of SMART-KPE which also uses BERT. For further comparison, we also report the strongest baseline result (RoBERTa2Joint) presented by Sun et al. (2020), and compare it to a RoBERTa-based variant of SMART-KPE in Section 5.1.

For evaluation of our generated keyphrases, we follow the official MS-MARCO guide and evaluation code [3]. Retrieval metrics include Precision, Recall and F1 at positions 1, 3 and 5, of which F1@3 is considered the main metric.

### 4.3 Implementation and Training Details

The model was implemented in Pytorch (Paszke et al., 2019) using the huggingface reimplementation of BERT (Wolf et al., 2019). Table 4 lists the parameters of our model. The maximum document length is 512 due to BERT limitations, and documents are zero-padded or truncated to this length. The model contains approximately 120M trainable parameters and was trained on a single GeForce GTX 1080 Ti GPU for around 12 hours to achieve best performance.

[3] https://github.com/microsoft/OpenKP

| Hyperparameter | Dimension or Value |
|---|---|
| BERT Embedding | 768 |
| Visual Feature | 18 |
| Snapshot Embedding | 512 |
| Visual Transformer | 3-head, 2-layer, 18-d hidden |
| Predictor Transformer | 6-head, 2-layer, 786-d hidden |
| Predictor Number | 4 |
| Predictor FFN | 128-ReLU-5 |
| Meta-feature FFN | 256-ReLU-4 |
| Optimizer | AdamW |
| Learning Rate | $1 \times 10^{-5}$ |
| Batch Size | 32 |

Table 4: Parameters used for training SMART-KPE.

## 5 Experimental Results and Analysis

### 5.1 Evaluation Results

Experimental results on the OpenKP dataset are listed in Table 3. We perform experiments on the full SMART-KPE model and its 3 variants, where *only micro-level visual features* (SMART-KPE-Micro), *only macro-level meta-features* (SMART-KPE-Macro), and *neither set of features* (SMART-KPE-Skeleton) are applied respectively. We see that all variants of BERT-based SMART-KPE outperform BERT2Tag and BERT2Joint on all metrics, suggesting the effectiveness of feature construction and strategy selection.

| Website Url | Plain Texts | Golden | Predicted |
|---|---|---|---|
| https://genius.com/Old-crow-medicine-show-wagon-wheel-lyrics | Wagon Wheel Old Crow Medicine Show Produced by David Rawlings Album Old Crow Medicine Show ... Wagon Wheel Lyrics ... Verse 1 Headed down south to the land of the pines ... | **Golden:**<br>Wagon Wheel<br>Old Crow Medicine Show<br>David Rawlings | **SMART-KPE-Skeleton:**<br>Wagon Wheel<br>Old Crow Medicine Show<br>Lyrics<br>**SMART-KPE-Full:**<br>Wagon Wheel<br>Old Crow Medicine Show<br>David Rawlings |
| https://www.proshareng.com/newscategory/%20Bonds%20&%20Fixed%20Income/1 | Bonds & Fixed Income ... Despite Year End Funding Pressures CBN Maintains OMO and FX Interventions ... Naira Crashes Below N360 Per Dollar at the Parallel Market as External ... | **Golden:**<br>Bonds<br>Fixed Income | **SMART-KPE-Skeleton:**<br>Bonds<br>CBN<br>Nigeria<br>**SMART-KPE-Full:**<br>CBN<br>Fixed Income<br>Bonds |
| https://jojo.fandom.com/wiki/Category:Part_4_Characters/ | Part 4 Characters ... These are the characters featured in Diamond Is Unbreakable. TRENDING PAGES Jotaro Kujo ... | **Golden:**<br>Diamond is Unbreakable<br>JoJo's Bizarre | **SMART-KPE-Skeleton:**<br>Characters<br>Diamond<br>Jojo<br>**SMART-KPE-Full:**<br>Jotaro Kujo<br>Characters<br>Joseph Joestar |

Table 5: Case Study of 3 web pages. Part of the original text, all golden keyphrases and top 3 predicted keyphrases are presented for each case. The correctly predicted keyphrases are highlighted in red. The snapshots of these 3 web pages are shown in Figure 2. Note that in the original data, punctuation is absent and keyphrases are case-insensitive. The text snippets we show here are restored from the original websites.

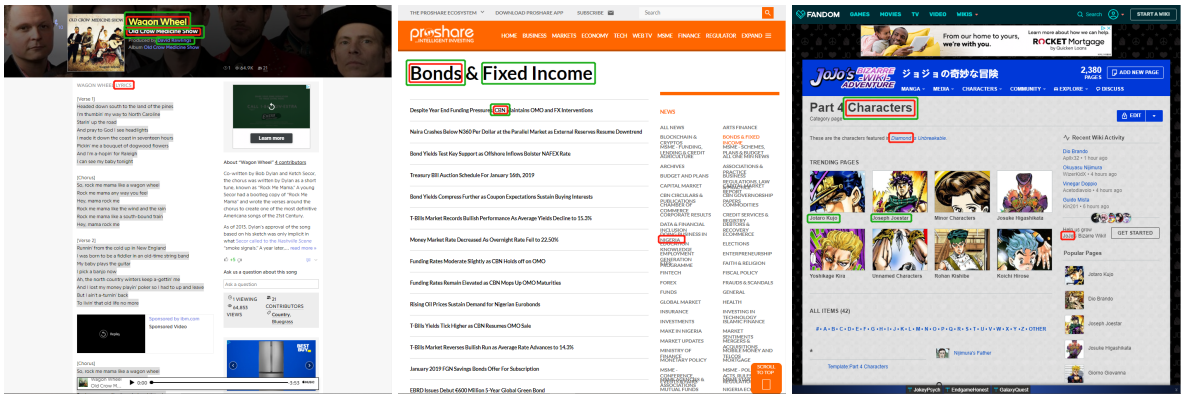
We further explore how different kinds of features function when extracting keyphrases. SMART-KPE-Skeleton outperforms the baselines even without the use of micro-level visual features and macro-level meta-features. This is due to the addition of title information as well as a more effective model structure. We observe further improvements as each multimodal feature component is added upon SMART-KPE-Skeleton, with the full model being the best combination. This analysis reveals the separate and joint effects of using meta-features and performing strategy selection, which formulates a complete approach to extract keyphrases.

We also change the BERT base in Unimodal Encoder to RoBERTa and replace the predictors in Multimodal Predictor from the tagging method to the joint extraction method(Sun et al., 2020) to fairly compare with the baseline with the best performance, RoBERTa2Joint. Experiment results are listed in the last two rows of Table 3. RoBERTa-based SMART-KPE outperforms both the baseline model and SMART-KPE-Full in all metrics, further demonstrating our model's flexibility to benefit from a more advanced text-based extraction backbone.

## 5.2 Case Study of Visual Feature Usage

We demonstrate the effects of introducing micro-level visual features by showing 3 cases from the validation set of OpenKP in Table 5. We present prediction results from SMART-KPE-Skeleton (SMART-KPE using only text features) and the complete SMART-KPE model. Snapshots of the original web pages are presented in Figure 2.

Cases #1 and #2 show how micro-level visual features help find the correct keyphrases. In Case #1, all the keyphrases are in the middle part of the webpage and more obvious than other words with different colors. The full model, successfully utilizing micro-level visual features, focuses on all of the keyphrases, while the skeleton model chooses a topic word in the first paragraph "Lyrics", resulting in a mistake. Case #2 shows a similar situation where the model with visual features finds the proper keyphrases that are much larger in font size, while the text-only model selects nouns elsewhere. On the other hand, Case #3 demonstrates a typical kind of web page where visual features can be misleading: an indexing page. In this kind of page, words larger in size and in bold are mostly

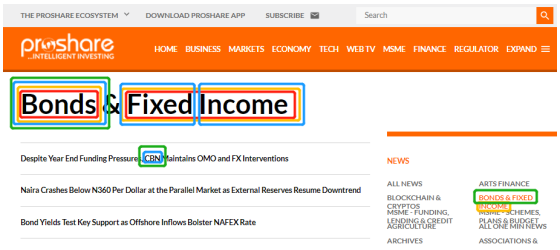



(a) Case #1 (b) Case #2 (c) Case #3

Figure 2: The snapshot of the websites listed in Table 5. Texts with green bounding boxes are keyphrases predicted by SMART-KPE-Full, and texts with red bounding boxes are keyphrases predicted by SMART-KPE-Skeleton. We can see the preference of the former model to focus on larger, bold and colorful texts.

entries to detailed contents, while the summarizing words before them, which are sometimes hidden in the small contents and easy to be omitted, are more likely to become keyphrases.


(a) Case #1




(b) Case #2

Figure 3: The predicted keyphrases of different predictors. Each color represents a specific predictor.

### 5.3 Analysis of Prediction Strategy Selection

In order to confirm that our predictors apply different tactics rather than simply duplicating each other, we calculate the overlap rate among the extracted keyphrases of different predictors.

We define the overlap rate for $n$ sets as:

$$O.R. = \frac{\sum_{k=1}^{n}(k-1)n_k}{(n-1)\sum_{k=1}^{n} n_k}, \tag{11}$$

where $n_k$ is the number of elements shared by $k$ different sets. An overlap rate of 0 indicates that no elements exist in more than one set while 1 indicates totally identical sets. As shown in Table 6, the overlap rates are low at all positions, which means the predictors are generating diverse keyphrases, leaving space for the selector to pick up final results with macro-level features.

| | O.R.@1 | O.R.@3 | O.R.@5 |
|---|---|---|---|
| SMART-KPE | 0.252 | 0.284 | 0.320 |

Table 6: Overlap rate of SMART-KPE predictions.

We also looked back into the data to see the keyphrases in context. Figure 3 presents the locations of extracted keyphrases by different predictors. We can generally conclude the difference as the result of: (1) Focus bias of modalities. Some multimodal predictors behave similarly to a purely textual predictor while others focus more on distinctly visual features. (2) Chunking bias of words. Different predictors will chunk words in different ways. (3) Ranking bias of keyphrases. Even if the same keyphrases are selected in different instances, the influence of their scores depends on other scores being compared. The more keyphrases being compared, the less utility an individual score has. We see an increase in overlap rate when more keyphrases are considered.

### 5.4 Paragraph Length

We investigate whether paragraph length or segmented length affects performance. In the experiments section, we truncate the document and only use the first 512 words. Apart from statistical results indicating that the majority of keyphrases appear rather near the front for most web pages, we calculated P@3 and F1@3 for different maximum document lengths to see how the extent of truncation affects the extent of the model's ability to generate reasonable keyphrases. The results are presented in Figure 4.

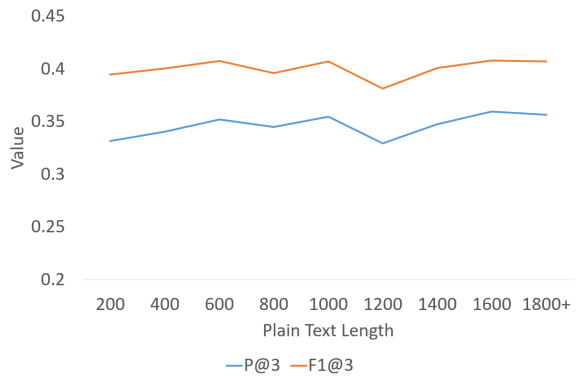



Figure 4: P@3 and F1@3 score from SMART-KPE for websites of different lengths.

From the figure, we can conclude that in our experiment, P@3 and F1@3 values are not heavily influenced by the length of the documents or amount of truncation. This is consistent with the intuition that keyphrases often appear at the top of the document(title, heading, first paragraph). Although there are advantages of increasing maximum text length, it is very resource-intensive and increases the difficulty of training because the amount of noise relative to the amount of signal increases as the maximum length increases.

## 6 Conclusions and Future Work

In this work we propose a Strategy-based Multimodal Architecture for Keyphrase Extraction (SMART-KPE) as a new state-of-the-art method for multimodal web-page keyphrase extraction. Different from traditional keyphrase extraction models mainly focusing on text, SMART-KPE illustrates the advantage of incorporating other modalities to help keyphrases location and salience prediction. Our proposed model outperforms several state-of-the-art baselines with the introduction of multimodal information. Through several case studies, we further illustrate how micro and macro-level features lead to the model's correct or incorrect selections.

As a first attempt to introduce macro-level meta-features for strategy selection, we believe there is much potential to refine and improve our approach. One high-level idea is to add further supervision to the current selector model based on empirical web page clustering, to better train the model to develop a set of more distinct keyphrase prediction strategies, and more effectively adjust the respective selector weights. We also plan to add more types of meta-features to generate richer multimodal representations. Furthermore, the SMART-KPE framework can be easily adapted to other NLP tasks, and we believe there is much potential in combining SMART-KPE with different models to further boost performance on open-domain KPE and other web-related tasks.

## Acknowledgments

This work was funded in part by NSF grants IIS 1822831 and 1917955 and funding from Microsoft.

## References

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

Erion Çano and Ondřej Bojar. 2019. Keyphrase generation: A text summarization struggle. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 666–672.

Jun Chen, Xiaoming Zhang, Yu Wu, Zhao Yan, and Zhoujun Li. 2018. Keyphrase generation with correlation constraints. *arXiv preprint arXiv:1808.07185*.

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

Mark Dredze, Hanna M Wallach, Danny Puller, and Fernando Pereira. 2008. Generating summary keywords for emails using topics. In *Proceedings of the 13th international conference on Intelligent user interfaces*, pages 199–206.

Maria Grineva, Maxim Grinev, and Dmitry Lizorkin. 2009. Extracting key terms from noisy and multitheme documents. In *Proceedings of the 18th inter-*

*national conference on World wide web*, pages 661–670.

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

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.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

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

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 the 44th annual meeting of the Association for Computational Linguistics*, pages 537–544. Association for Computational Linguistics.

Su Nam Kim and Timothy Baldwin. 2012. Extracting keywords from multi-party live chats. In *Proceedings of the 26th Pacific Asia Conference on Language, Information, and Computation*, pages 199–208.

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.

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.

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: Volume 1-Volume 1*, pages 257–266. Association for Computational Linguistics.

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

Olena Medelyan and Ian Witten. 2002. Thesaurus based automatic keyphrase indexing. In *Proceedings of the 6th ACM/IEED-CS joint conference on Digital libraries*, pages 296–297.

Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. 2017. Deep keyphrase generation. *arXiv preprint arXiv:1704.06879*.

Thuy Dung Nguyen and Min-Yen Kan. 2007. Key phrase extraction in scientific publications. In *Proceeding of International Conference on Asian Digital Libraries*, pages 317–326.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: a human-generated machine reading comprehension dataset.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237.

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

Si Sun, Chenyan Xiong, Zhenghao Liu, Zhiyuan Liu, and Jie Bao. 2020. Joint keyphrase chunking and salience ranking with bert.

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

Xiaojun Wan and Jianguo Xiao. 2008. Collabrank: Towards a collaborative approach to single-document keyphrase extraction. In *Proceedings of 22nd International Conference on Computational Linguistics*, pages 969–976, Manchester, UK.

Ian Witten, Gordon Paynter, Eibe Frank, Car Gutwin, and Graig Nevill-Manning. 1999. Kea:practical automatic key phrase extraction. In *Proceedings of the fourth ACM conference on Digital libraries*, pages 254–256.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Lee Xiong, Chuan Hu, Chenyan Xiong, Daniel Campos, and Arnold Overwijk. 2019. Open domain web keyphrase extraction beyond language modeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5178–5187.

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

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