Open Domain Web Keyphrase Extraction
Beyond Language Modeling

Lee Xiong   Chuan Hu   Chenyan Xiong   Daniel Campos   Arnold Overwijk
Microsoft AI and Research
Redmond, WA 98052, USA
`{lexion, chuah, cxiong, dacamp, arnoldo}@microsoft.com`

Abstract

This paper studies keyphrase extraction in real-world scenarios where documents are from diverse domains and have variant content quality. We curate and release OpenKP, a large scale open domain keyphrase extraction dataset with near one hundred thousand web documents and expert keyphrase annotations. To handle the variations of domain and content quality, we develop BLING-KPE, a neural keyphrase extraction model that goes beyond language understanding using visual presentations of documents and weak supervision from search queries. Experimental results on OpenKP confirm the effectiveness of BLING-KPE and the contributions of its neural architecture, visual features, and search log weak supervision. Zero-shot evaluations on DUC-2001 demonstrate the improved generalization ability of learning from the open domain data compared to a specific domain.

1 Introduction

Automatically extracting keyphrases that are salient to the document meanings is an essential step to semantic document understanding. An effective keyphrase extraction (KPE) system can benefit a wide range of natural language processing and information retrieval tasks (Turney, 2001; Hasan and Ng, 2014). Recent neural methods formulate the task as a document-to-keyphrase sequence-to-sequence task. These neural KPE models have shown promising results compared to previous systems (Chen et al., 2018; Meng et al., 2017; Ye and Wang, 2018).

Noticeably, the recent progress in neural KPE is mostly observed in documents originating from the scientific domain (Meng et al., 2017; Augenstein et al., 2017). Perhaps because the scientific domain has sufficient training data for these neural methods: Authors are in the practice of as-

signing keyphrases to their publications. In real-world scenarios, most potential applications of KPE deal with diverse documents originating from sparse sources that are rather different from scientific papers. They often include a much diverse document structure and reside in various domains whose contents target much wider audiences than scientists. It is unclear how well the neural methods trained in the scientific domain generalize to other domains and in real-world scenarios.

This paper focuses on the task of open domain web keyphrase extraction, which targets KPE for web documents without any restriction of the domain, quality, nor content of the documents. We curate and release a large scale open domain KPE dataset, OpenKP, which includes about one hundred thousand web documents with expert keyphrase annotations.[1] The web documents are randomly sampled from the English fraction of a large web corpus and reflect the characteristics of typical web pages, with large variation in their domains and content qualities. To the best of our knowledge, this will be the first publicly available open domain manually annotated keyphrase extraction dataset at this scale.

This paper develops BLING-KPE, **B**eyond **L**anguage Understand**ING K**ey**P**hrase **E**xtraction, that tackles the challenges of KPE in documents from variant domains and content qualities. BLING-KPE uses a convolutional transformer architecture to model the language properties in the document, while also goes beyond by introducing the visual representation of the document and weak supervision from search user clicks.

The *visual presentations* of the document, including the location, size, font, and HTML structure of each text piece in the document, are integrated as visual features to the word embeddings

---

[1]The dataset, resources, and future updates are available at `aka.ms/BLING`.

in BLING-KPE. BLING-KPE learns to model the visual representations together with the document language in its network.

The *weak supervision from search clicks* is formulated as a pre-training task: Query Prediction. It trains the model to predict which phrase in the document has been used as a "click query", a query that a user issued to search and click on the document. The click queries on a document reflect the user's perceptions of the relatedness and importance when searching the document and can be considered as pseudo keyphrases. Pre-training on this weak supervision brings in training signals available at scale in commercial search systems.

Our experiments on OpenKP demonstrate the effectiveness of BLING-KPE. It outperforms standard KPE baselines, recent neural approaches and a highly optimized commercial KPE system by large margins. Ablation studies show the contributions of the neural architecture, visual features, and search weak supervision to BLING-KPE; removing any of them significantly reduces its accuracy.

Another advantage of learning from real-world open domain documents is improved generalization ability. We conduct zero-shot evaluations on the DUC-2001 news KPE datasets (Wan and Xiao, 2008b), where neural KPE systems are evaluated without seeing any labels from their news articles. BLING-KPE trained on OpenKP is the only neural method that outperforms traditional non-neural KPE methods, while neural KPE systems trained on the scientific documents do not generalize well to the news domain due to the domain differences.

2   Related Work

The classic keyphrase extraction systems typically include two components: *candidate keyphrase extraction* and *keyphrase importance estimation* (Hasan and Ng, 2014). The candidate keyphrases are often extracted by heuristic rules, for example, finding phrases following certain POS tag sequences (Wan and Xiao, 2008b; Liu et al., 2009a; Mihalcea and Tarau, 2004), predefined lexical patterns (Nguyen and Phan, 2009; Medelyan et al., 2009), or using entities as candidate phrases (Grineva et al., 2009).

The importance of the candidate keyphrases can be estimated by unsupervised or supervised methods. The unsupervised methods leverage the graph structures between phrases in the doc-

ument (Mihalcea and Tarau, 2004; Wan and Xiao, 2008a,b), and topic information from topic modeling (Grineva et al., 2009; Liu et al., 2009b, 2010). The supervised keyphrase selection methods formulate a classification or ranking task and combine features from phrase frequencies (Witten et al., 2005), document structures (Chen et al., 2005; Yih et al., 2006), and external resources such as Wikipedia (Medelyan et al., 2009) and query log (Yih et al., 2006).

Recently, neural techniques have been applied to keyphrase tasks. Meng et al. formulate a seq2seq learning task that learns to extract and generate the keyphrase sequence from the document sequence; they incorporate a copy mechanism to the seq2seq RNN to extract phrases in the generation process (CopyRNN) (Meng et al., 2017). Improving this seq2seq setup has been the focus of recent research, for example, adding diverse constraints to reduce the duplication of produced keyphrases (Yuan et al., 2018; Chen et al., 2018), bringing auxiliary tasks to reduce the needs of training data (Ye and Wang, 2018), and adding title information to improve model accuracy (Chen et al., 2019).

The recent neural KPE methods have shown strong performances on the scientific domain, where large scale training data is available from the author assigned keyphrases on papers (Meng et al., 2017). Such specific domain training data limits the model generalization ability. Chen et al. show the seq2seq keyphrase generation models trained on scientific papers do not generalize well to another domain (Chen et al., 2018).

In general, previous research finds automatic keyphrase extraction a challenging task: its state-of-the-art accuracy is much lower than other language processing tasks, while supervised methods do not necessarily outperform simple unsupervised ones. Hasan and Ng (2014) pointed out potential ways to improve automatic keyphrase extraction, including better incorporation of *background knowledge*, better handling *long documents*, and better *evaluation* schemes. BLING-KPE aims to address these challenges by incorporating pre-training as a form of *background knowledge*, visual information to improve *long document* modeling, and OpenKP as a large scale open domain *evaluation* benchmark.

3 Open Domain Keyphrase Benchmark

This section describes the curation of `OpenKP` and its notable characteristics.

3.1 Data Curation

Documents in `OpenKP` include about seventy thousand web pages sampled from the index of Bing search engine.[2] The sampling is conducted on the pool of pages seen by United State users between Nov 2018 and Feb 2019.

There is no restriction on the domain or type of documents. They can be content-oriented pages like news articles, multi-media pages from video sites, or indexing pages with many hyperlinks. `OpenKP` is designed to reflect the diverse properties of web documents in the internet.

Keyphrase Labels are generated by our expert annotators. For each document, they examine the rendered web page and manually label 1-3 keyphrases following these definitions:

- *Salience:* A keyphrase captures the essential meaning of the page with no ambiguity.

- *Extraction:* The keyphrase has to appear in the document.

- *Fine-Grained:* The keyphrase cannot be general topics, such as "Sports" and "Politics".

- *Correct & Succinct:* The keyphrase has to form a correct English noun phrase, while also cannot be clauses or sentences.

We use the extraction setting to ensure labeling consistency and to increase annotation speed, which is around 42 pages per hour.

Expert Agreements. Our annotation experts are trained employees dedicated to providing high-quality annotations on web documents. We follow standard practice in generating annotations for production systems, which included regular touch-points to understand the confusion, as well as updates on the judgment guidelines to resolve ambiguities.

To study the task difficulty, we had five judges each annotate the same 50 random URLs. We measure the pairwise agreements between experts at different depths by Exact Match on the whole keyphrase, as well as the overlap between select keyphrases' unigrams. The agreement between judges is listed in Table 1.

Table 1: The agreements between pairs of expert judges at different annotation depth. Exact and Unigram show the percentage of judge agreement on exact keyphrases and overlapped unigrams.

Judge Depth	Exact Match	Unigram Match
Keyphrase@1	64.74%	64.74%
Keyphrase@2	48.30%	63.12%
Keyphrase@3	43.51%	57.66%

The results confirm that open domain keyphrase extraction is not an easy task. When measuring agreement for the top 3 keyphrases, our expert judges completely agree on about 43% of keyphrase pairs. Compared to the previous small scale annotations, for example, on DUC-2001's news articles (Wan and Xiao, 2008b), annotating web pages with diverse contents and domains are harder.

We manually examined these annotations and found two sources of disagreement: Chunking Variances and KP Choices.

Chunking Variances we define as two judges pick different boundaries of the same concept. For example, one judge may select "Protein Synthesis" as the keyphrase, and others may select "Protein" and "Synthesis" as two separate keyphrases. We found Chunking Variances consist of about 20% of disagreements. As shown in Table 1, the judge agreements is substantially higher on Unigram overlaps than on Exact matches, indicating that they may select chunks that overlap with each other but not exactly the same.

KP Choices we define as two judges pick different keyphrases. The judges agree mostly (64%) on the first entered keyphrase, as shown in Table 1. The variations on the second and third keyphrases are larger. However, we found the variations are more about which keyphrases they choose to enter, *not about whether a phrase is a keyphrase or not*. The variations on judge labels mostly reflect the missing positives in `OpenKP`; most of the keyphrases annotated by judges are correct. We can reduce the missing positives by a deeper annotation, i.e. ten keyphrases per document, or by labeling all candidate phrases with classification labels (Liu et al., 2018). However, that will significantly reduce the number of total documents in `OpenKP`, as each document costs much more to annotate. We chose the current design choice of `OpenKP` to favor a larger amount of training labels, which, in our experience, is more effective in
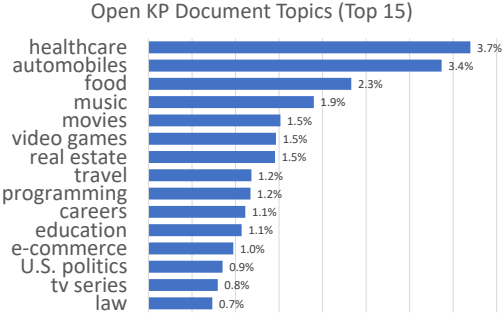
Figure 1: The most popular topics in OpenKP .

Table 2: Statistics of OpenKP used in our experiments. The new version on MSMARCO include 150K documents.

Statistics	Mean	STD
Doc Length	900.4	1494.4
# of KPs per Doc	1.8	0.8
Keyphrase Length	2.0	0.96
Doc Vocabulary Size	1.5M	n.a.
KP Vocabulary Size	62K	n.a.
# of Documents	68K	n.a.
# of Unique KPs	99.6K	n.a.

training deep neural models.

3.2 Data Characteristics

Table 2 lists the statistics of OpenKP. The document length is the length of the text parsed from the HTML of the web page, using a production HTML parser. The parsed texts will be released with the dataset. These statistics reflect the large variations in the document contents; their length varies a lot and share little common keyphrases, as shown by a large number of unique keyphrases.

We also leverage a production classifier to classify OpenKP documents into 5K predefined domains. The top 15 most popular classes and their distributions are shown in Figure 1. As expected, these documents have a large variation in their topic domains. The most popular domain, "healthcare", only covers 3.7% documents; the tenth most popular topic only covers 1% of documents. Moreover, the top 15 classes make up less than 25% of the entire dataset which showcases what a domain diverse dataset OpenKP is.

4 Keyphrase Extraction Model

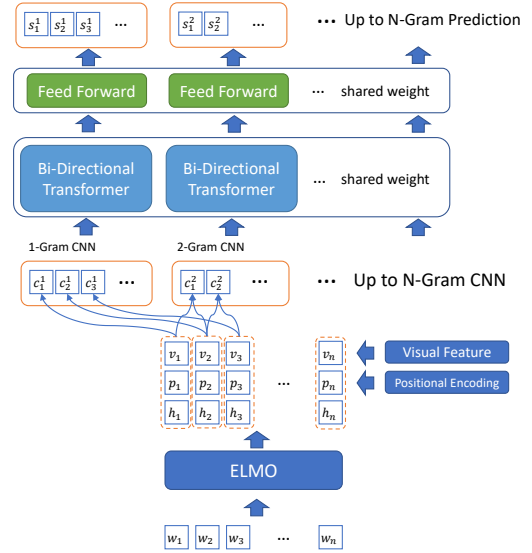This section describes the architecture, visual features, and weak supervision of BLING-KPE.

Figure 2: The BLING-KPE model architecture

4.1 Network Architecture

As shown in Figure 2, BLING-KPE is a keyphrase *extraction* model. It takes the word sequence of the document, $d = \{w_1, ...w_i, ...w_n\}$, and assigns keyphrase scores to its n-gram: $f(w_{i:i+k}, d)$. This process includes two main components: Hybrid Word Embedding and Convolutional Transformer.

Hybrid Word Embedding. BLING-KPE represents each word by its ELMo embedding, position embedding, and visual features.

The ELMo embedding brings the local contextual information:

$$\vec{h}_i = \text{ELMo}(w_i). \qquad (1)$$

The standard pre-trained ELMo is used (Peters et al., 2018).

The position embedding models the location the word in the document content. It uses the standard sinusoidal position embedding (Vaswani et al., 2017):

$$\vec{\text{pos}}_i(2p) = sin(i/10000^{2p/P}), \qquad (2)$$
$$\vec{\text{pos}}_i(2p+1) = cos(i/10000^{2p/P}). \qquad (3)$$

The p-th dimension of the position embedding is a function of its position (i) and dimension (p).

The visual features represent the visual presentation of each word. We denote the visual feature as $\vec{v}_i$ and will describe its details in §4.2.

The hybrid word embedding is the concatenation of the three:

$$\vec{w}_i = \vec{h}_i \frown \vec{\text{pos}}_i \frown \vec{v}_i. \qquad (4)$$

Convolutional Transformer. BLING-KPE uses a convolutional transformer architecture to model n-grams and their interactions.

It first composes the hybrid word embeddings to n-gram embeddings using CNNs. The embedding of i-th k-gram is calculated as

$$\vec{g}_i^k = \text{CNN}^k(\vec{w}_{i:i+k}), \tag{5}$$

where k is the length of the n-gram, $1 \leq k \leq K$. K is the maximum length of allowed candidate n-grams. Each k-gram has its own set of convolution filters CNN^k with window size k and stride 1.

It then models the interactions between k-grams using Transformer (Vaswani et al., 2017).

$$\vec{t}_i^k = \text{Transformer}_i(\vec{G}^k), \tag{6}$$

$$\vec{G}^k = \vec{g}_1^k \frown ... \frown \vec{g}_i^k ... \frown \vec{g}_{n-k+1}^k. \tag{7}$$

The sequence $\vec{G}^k$ is the concatenations of all k-gram embeddings. The Transformer models the self-attentions between k-grams and fuses them to global contextualized embeddings.

The Transformer is convolutional on all length k of n-grams; the same parameters are used model the interactions between n-grams at each length, to reduce the parameter size. The intuition is that the interactions between bi-grams and that between tri-grams are not significantly different.

The final score of an n-gram is calculated by a feedforward layer upon the Transformer. Like the Transformer, the same feedforward layer is applied (convolutional) on all n-grams.

$$f(w_{i:i+k}, d) = \text{softmax}_{i,k}(s_i^k), \tag{8}$$

$$s_i^k = \text{Feedforward}(\vec{t}_i^k). \tag{9}$$

The softmax is taken over all possible n-grams at each position i and each length k. The model decides the span location and length jointly.

Learning. The whole model is trained as a classification problem using cross-entropy loss:

$$l = \text{Cross-Entropy}(y_i^k, f(w_{i:i+k}, d)), \tag{10}$$

where y_i^k is the label of whether the phrase $w_{i:i+k}$ is a keyphrase of the document.

4.2 Visual Features

We extract four groups of visual features for each word in the document.

- **Size** features include the height and width of the text block a word appears in.

Table 3: Visual Features. All features are extracted at per word level and the parent block level (the parent node of the word in the HTML DOM tree).

Name	Dimension
Font Size	1×2
Text Block Size	2×2
Location in Rendered Page	2×2
Is Bold Font	1×2
Appear In Inline	1×2
Appear In Block	1×2
Appear In DOM Tree Leaf	1×2

- **Location** features include the 2-d location of the word in the rendered web page.

- **Font** feature includes the font size and whether the word is in Bold.

- **DOM** features include whether the word appears in "inline" or "block" HTML tags, also whether it is in a leaf node of the DOM tree.

The full feature set is listed in Table 3. We double the features by including the same features from the word's parent block in the DOM tree. The visual features are included in the OpenKP releases.

4.3 Weak Supervisions from Search

An application of keyphrases is information retrieval. The extracted keyphrases are expected to capture the main topic of the document, thus can provide high quality document indexing terms (Gutwin et al., 1999) or new semantic ranking features (Xiong et al., 2018). Reversely, user clicks bring the user's perception of the document during the search and provide a large number of feedback signals for document understanding (Croft et al., 2010).

BLING-KPE leverages the user feedback signals as weak supervision, in the task of Query Prediction. Given the document d, BLING-KPE learns to predict its click queries $Q = \{q_1, ..., q_m\}$.

This pre-training step uses the same cross entropy loss:

$$l_{pre} = \text{Cross-Entropy}(y_i', f(q_i, d)), \tag{11}$$

where y_i' indicates whether the query q_i is a click query and also appears as an n-gram in the document d. The Query Prediction labels exist at scale in commercial search logs and provide a large number of pre-training signals.

Table 4: Statistics of Query Prediction Dataset. The data is from a sample of Bing search log in on week.

Statistics	Mean	STD
Doc Length	1211.8	1872.6
# of Query per Doc	1.32	0.76
Query Length	2.4	1.07
Doc Vocabulary Size	15M	n.a
Query Vocabulary Size	383K	n.a
# of Documents	1.6M	n.a.
# of Unique Queries	1.5M	n.a.

Table 5: Parameters to learn in BLING-KPE.

Component	Dimensions
ELMo	pre-trained and frozen
Position Embedding	256
Visual Feature	18
N-gram CNN	512 filter, 1-5 window size (ngram)
Transformer	8 head, 512 hidden dimension
Feedforward	512-relu-512-relu-1

5 Experimental Methodology

Datasets used in our experiments include OpenKP, as described in §3, Query Prediction, and DUC-2001 (Wan and Xiao, 2008b).

The Query Prediction data is sampled from the Bing search log with navigational and offensive queries filtered out. We keep only the click queries that are included as an n-gram in the document to be consistent with OpenKP's extractive setting. The statistics of the sample is listed in Table 4.

DUC-2001 is the KPE extraction dataset on DUC news articles (Wan and Xiao, 2008b). It includes 309 news articles and on average 8 keyphrase per article.

We use random 80%-20% train-test splits on OpenKP and Query Prediction. On OpenKP, BLING-KPE is first pre-trained on Query Prediction and then further trained on its manual labels. There is no overlap between the documents in Query Prediction and OpenKP.

DUC-2001 uses the zero-shot evaluation setting from prior research (Meng et al., 2017; Chen et al., 2018); no labels in DUC-2001 are used to train nor validate the neural models. It tests neural models' generalization ability from the training domain to a different testing domain.

Evaluation Metrics. OpenKP and Query Prediction use **P**recision and **R**ecall@{1, 3, 5}. DUC-2001 uses F1@10, the same as prior research (Meng et al., 2017; Chen et al., 2018).

Statistically, significant improvements are evaluated by permutation test with p<0.05 on OpenKP and Query Prediction. The baselines on DUC-2001 reuse scores from previous results; the statistical significant test is not applicable as per document results are not shared.

Baselines. OpenKP and Query Prediction experiments compare `BLING-KPE` with: traditional KPE methods, production systems, and a neural baseline.

Traditional KPE baselines include the follows.

- TFIDF is the unsupervised frequency based KPE system. The IDF scores are calculated on the corresponding corpus.

- TextRank is the popular graph-based unsupervised KPE model (Mihalcea and Tarau, 2004). Our in-house implementation is used.

- LeToR is the feature-based KPE model. It use LambdaMart (Burges, 2010) and standard KPE features, i.e. those in KEA (Witten et al., 2005).

The production baselines include two versions.

- PROD is our current feature-based production KPE system. It uses many carefully engineered features and LambdaMart.

- PROD (Body) is the same system but only uses the body text, i.e. the title is not used.

All these unsupervised and feature-based methods use the same keyphrase candidate selection system with PROD.

The neural baseline is CopyRNN (Meng et al., 2017). We use their open-source implementation and focus on the OpenKP dataset which is publicly available.

Implementation Details. Table 5 lists BLING-KPE parameters. The training uses Adam optimizer, learning rate 0.3 with logarithmic decreasing to 0.001, batch size 16, and 0.2 dropout probability in n-gram CNN, Transformer and feedforward layers. Learning takes about 2.5 hours (2 epochs) to converge on `OpenKPE` and about 13 hours (3 epochs) on Query Prediction, based on validation loss. In BLING-KPE, the maximum document length is 256 and documents are zero-padded or truncated to this length. Baselines use the original documents, except CopyRNN which works better with 256. The maximum n-gram length is set to five (K=5).

Table 6: Keyphrase Extraction Accuracy. **Bold** marks statistically significant improvements over all baselines.

Method	OpenKP						Query Prediction					
	P@1	R@1	P@3	R@3	P@5	R@5	P@1	R@1	P@3	R@3	P@5	R@5
TFIDF	0.283	0.150	0.184	0.284	0.137	0.347	0.403	0.332	0.204	0.491	0.133	0.526
TextRank	0.077	0.041	0.062	0.098	0.055	0.142	0.132	0.111	0.089	0.218	0.073	0.295
LeToR	0.301	0.158	0.173	0.268	0.127	0.324	0.328	0.271	0.169	0.406	0.119	0.471
PROD	0.353	0.188	0.195	0.299	0.131	0.331	0.376	0.308	0.197	0.468	0.129	0.505
PROD (Body)	0.214	0.094	0.130	0.196	0.094	0.234	0.353	0.287	0.191	0.454	0.125	0.492
CopyRNN	0.288	0.174	0.185	0.331	0.141	0.413	–	–	–	–	–	–
BLING-KPE	**0.404**	**0.220**	**0.248**	**0.390**	**0.188**	**0.481**	**0.540**	**0.449**	**0.275**	**0.654**	**0.188**	**0.729**

Table 7: Performance of BLING-KPE ablations. *Italic* marks statistically significant worse performances than Full Model.

Method	OpenKP						Query Prediction					
	P@1	R@1	P@3	R@3	P@5	R@5	P@1	R@1	P@3	R@3	P@5	R@5
No ELMo	*0.270*	*0.145*	*0.172*	*0.271*	*0.132*	*0.347*	*0.323*	*0.274*	*0.189*	*0.450*	*0.136*	*0.527*
No Transformer	*0.389*	*0.211*	0.247	*0.385*	0.189	0.481	*0.489*	*0.407*	*0.258*	*0.618*	*0.178*	*0.698*
No Position	*0.394*	*0.213*	0.247	*0.386*	0.187	*0.475*	0.543	0.452	0.281	0.666	0.191	0.742
No Visual	*0.370*	*0.201*	*0.230*	*0.362*	*0.176*	*0.450*	*0.492*	*0.409*	*0.258*	*0.615*	*0.178*	*0.695*
No Pretraining	*0.369*	*0.198*	*0.236*	*0.367*	*0.181*	*0.460*	–	–	–	–	–	–
Full Model	0.404	0.220	0.248	0.390	0.188	0.481	0.540	0.449	0.275	0.654	0.188	0.729

6 Evaluation Results

Three experiments are conducted to evaluate the accuracy of BLING-KPE, the source of its effectiveness, and its generalization ability.

6.1 Overall Accuracy

The overall extraction accuracy on OpenKP and Query Prediction is shown in Table 6.

TFIDF works well on both tasks. Frequency-based methods are often strong baselines in document representation tasks. LeToR performs better than its frequency feature TFIDF in OpenKP but worse on Query Prediction. Supervised methods are not necessarily stronger than unsupervised ones in KPE (Hasan and Ng, 2014). TextRank does not work well in our dataset; its word graph is likely misguided by the noisy contents.

PROD, our feature-based production system, outperforms all other baselines by large margins on OpenKP. It is expected as it is highly optimized with a lot of engineering efforts. Nonetheless, adapting a complex feature-based system to a new task/domain requires extra engineering work; directly applying it to the Query Prediction task does not work well. The feature-based Production system also needs the title information; PROD (Body) performs much worse than PROD.

CopyRNN performs relatively well on OpenKP, especially on later keyphases. The main challenge for CopyRNN is the low-quality and highly vari-

ant contents on the web. Real-world web pages are not cohesive nor well-written articles but include various structures such as lists, media captions, and text fragments. Modeling them as a word sequence is not ideal. The other differences are not as significant: The vocabulary size and training data size on Query Prediction are similar to CopyRNN's KP 40K dataset; CopyRNN performs better on keyphrase extraction than generation in KP20k (Meng et al., 2017).

BLING-KPE outperforms all other methods by large margins. The improvements are robust and significant on both tasks, both metrics, and on all depths. It achieves 0.404 P@1 on OpenKP and recovers 72% of clicked queries at depth 5 on Query Prediction. The sources of this effectiveness is studied in the next experiment.

6.2 Ablation Study

Table 7 shows ablation results on BLING-KPE's variations. Each variation removes a component and keeps all others unchanged.

ELMo Embedding. We first verify the effectiveness of using ELMo embedding by replacing ELMo with the WordPiece token embedding (Wu et al., 2016). The accuracy of this variation is much lower than the accuracy of the full model and others. The result is shown in the first row of Table 7. The context-aware word embedding is a necessary component of BLING-KPE.

Network Architecture. The second part of

Table 8: Performance on DUC-2001. Neural models are evaluated directly on DUC-2001 without fine-tuning on DUC labels. Results better than TFIDF are marked **Bold**.

Method	F1@10	Method	F1@10
TFIDF	0.270	TopicRank	0.154
TextRank	0.097	KeyCluster	0.140
SingleRank	0.256	ExpandRank	0.269
Trained by Scientific Papers			
CopyRNN	0.164	CorrRNN	0.173
BLING-KPE (No Visual, No Pretraining)			0.267
Trained by Open Domain Documents			
BLING-KPE (No Visual., No Pretraining)			**0.282**

Table 7 studies the contribution of Transformer and position embedding. Transformer contributes significantly to Query Prediction; with a lot of training data, the self-attention layers capture the global contexts between n-grams. But on OpenKP, its effectiveness is mostly observed on the first position. The position embedding barely helps, since real-world web pages are often not one text sequence.

Beyond Language Understanding. As shown in the second part of Table 7, both visual features and search pretraining contribute significantly to BLING-KPE's effectiveness. Without either of them, the accuracy drops significantly. Visual features even help on Query Prediction, though users issued the click queries and clicked on the documents *before* seeing its full page.

The crucial role of ELMo embeddings confirm the benefits of bringing background knowledge and general language understanding, in the format of pre-trained contextual embedding, in keyphrase extraction. The importance of visual features and search weak supervisions confirms the benefits of going beyond language understanding in modeling real-world web documents.

6.3 Generalization Ability

This experiment studies the generalization ability of BLING-KPE using the zero-shot evaluation on DUC-2001 (Meng et al., 2017; Chen et al., 2018). For fair comparisons, we use KP20K or OpenKP, the two public datasets, to train the No Visual & No Pretraining version of BLING-KPE, and evaluate on DUC-2001 directly. No labels in DUC are used to fine-tune the neural models.

To adjust to DUC's larger number of keyphrases, we apply the trained BLING-KPE on the 256-length chunks of DUC articles
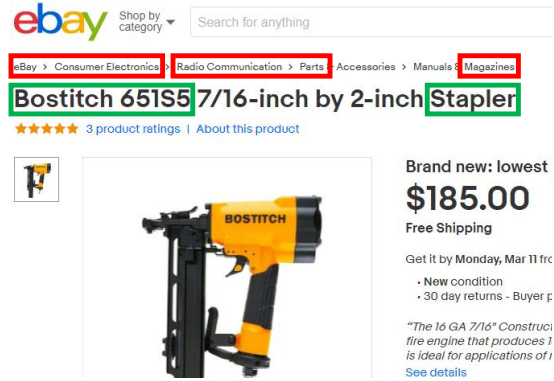

Figure 3: An example of visual feature contributions in BLING-KPE. The big green blocks are the keyphrases extracted with visual features. The small read blocks are those extracted without visual information.

and merge the extracted keyphrases using simple heuristics:

- Weighted Sum: Scores of the same keyphrase from different chunks are summed with weights 0.9^p. P is the index of the chunk.

- Deduplication: A keyphrase is discarded if it is a sub-string of a top 1/4 ranked keyphrase.

The results are shown in Table 8. BLING-KPE, when trained with OpenKP, is the only neural method that outperforms TFIDF in this zero-shot evaluation. It outperforms previous neural methods by more than 60%, and itself when trained on KP20k, confirming the strong generalization ability of BLING-KPE and the training with OpenKP.

6.4 Discussion

Our manual case studies found many interesting examples that illustrate the advantage of modeling documents with visual information.

For example, in Figure 3, the page is annotated with "Bostitch 651S5", the product name, and "Stapler", the product type. Their salience is highlighted by larger and bold fonts, which are picked up by BLING-KPE. However, without the visual information, the product ontology names are extracted as keyphrases: they are meaningful concepts, correlated with the page content, and positioned at the beginning of the document—only that they appear less important in the web page by design.

7 Conclusion

This paper curates OpenKP, the first public large scale open domain keyphrase extraction benchmark to facilitate future research keyphrase extraction research in real-world scenarios. It also develops `BLING-KPE`, which leverages visual representation and search-based weak supervision to model real-world documents with variant contents, appearances, and diverse domains.

Our experiments demonstrate the robust improvements of BLING-KPE compared to previous approaches. Our studies showcase how BLING-KPE's language understanding, visual features and search weak supervision jointly deliver this effective performance, as well as its generalization ability to an unseen domain in zero-shot setting.

In the future, we plan to extend OpenKP with more annotated documents and connect it with downstream applications.

References

Isabelle Augenstein, Mrinal Das, Sebastian Riedel, Lakshmi Vikraman, and Andrew McCallum. 2017. SemEval 2017 task 10: ScienceIE - extracting keyphrases and relations from scientific publications. In *SemEval@ACL*.

Chris J.C. Burges. 2010. From RankNet to LambdaRank to LambdaMART: An overview. MSR-TR-2010-82.

Jun Chen, Xiaoming Zhang, Yu Wu, Zhao Yan, and Zhoujun Li. 2018. Keyphrase generation with correlation constraints. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018*, pages 4057–4066.

Mo Chen, Jian-Tao Sun, Hua-Jun Zeng, and Kwok-Yan Lam. 2005. A practical system of keyphrase extraction for web pages. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 277–278. ACM.

Wang Chen, Yifan Gao, Jiani Zhang, Irwin King, and Michael R. Lyu. 2019. Title-guided encoding for keyphrase generation. In *The Thirty-Third AAAI Conference on Artificial Intelligence 2019*, pages 6268–6275.

W Bruce Croft, Donald Metzler, and Trevor Strohman. 2010. *Search Engines: Information Retrieval in Practice*. Addison-Wesley Reading.

Maria P. Grineva, Maxim N. Grinev, and Dmitry Lizorkin. 2009. Extracting key terms from noisy and multitheme documents. In *Proceedings of the 18th International Conference on World Wide Web, WWW 2009*, pages 661–670.

Carl Gutwin, Gordon W. Paynter, Ian H. Witten, Craig G. Nevill-Manning, and Eibe Frank. 1999. Improving browsing in digital libraries with keyphrase indexes. *Decision Support Systems*, 27:81–104.

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, ACL 2014*, pages 1262–1273.

Feifan Liu, Deana Pennell, Fei Liu, and Yang Liu. 2009a. Unsupervised approaches for automatic keyword extraction using meeting transcripts. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, HLT-NAACL 2009*, pages 620–628.

Zhengzhong Liu, Chenyan Xiong, Teruko Mitamura, and Eduard Hovy. 2018. Automatic event salience identification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018s*, pages 1226–1236.

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, EMNLP 2010*, pages 366–376.

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

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, EMNLP 2009*, pages 1318–1327.

Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. 2017. Deep keyphrase generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017*, pages 582–592.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing , EMNLP 2004*, pages 404–411.

Chau Q. Nguyen and Tuoi T. Phan. 2009. An ontology-based approach for key phrase extraction. In *ACL 2009, Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP, ACL/IJCNLP 2009*, pages 181–184.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018*, pages 2227–2237.

Peter D. Turney. 2001. Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In *Machine Learning: EMCL 2001, 12th European Conference on Machine Learning, 2001, Proceedings*, pages 491–502.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems, NIPS 2017*, pages 5998–6008.

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

Xiaojun Wan and Jianguo Xiao. 2008b. Single document keyphrase extraction using neighborhood knowledge. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008*, pages 855–860.

Ian H Witten, Gordon W Paynter, Eibe Frank, Carl Gutwin, and Craig G Nevill-Manning. 2005. KEA: Practical automated keyphrase extraction. In *Design and Usability of Digital Libraries: Case Studies in the Asia Pacific*, pages 129–152. IGI Global.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Chenyan Xiong, Zhengzhong Liu, Jamie Callan, and Tie-Yan Liu. 2018. Towards better text understanding and retrieval through kernel entity salience modeling. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018*, pages 575–584.

Hai Ye and Lu Wang. 2018. Semi-supervised learning for neural keyphrase generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4142–4153.

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, WWW 2006*, pages 213–222.

Xingdi Yuan, Tong Wang, Rui Meng, Khushboo Thaker, Daqing He, and Adam Trischler. 2018. Generating diverse numbers of diverse keyphrases. *CoRR*, abs/1810.05241.