

# Joint Keyphrase Chunking and Saliency Ranking with BERT

Si Sun<sup>1</sup>    Chenyan Xiong<sup>2</sup>    Zhenghao Liu<sup>3</sup>    Zhiyuan Liu<sup>3</sup>    Jie Bao<sup>1</sup>

<sup>1</sup>Department of Electronic Engineering, Tsinghua University, Beijing, China

<sup>2</sup>Microsoft Research AI, Redmond, USA

<sup>3</sup>Department of Computer Science and Technology, Tsinghua University, Beijing, China

Institute for Artificial Intelligence, Tsinghua University, Beijing, China

State Key Lab on Intelligent Technology and Systems, Tsinghua University, Beijing, China

## Abstract

An effective keyphrase extraction system requires to produce self-contained high quality phrases that are also key to the document topic. This paper presents BERT-JointKPE, a multi-task BERT-based model for keyphrase extraction. JointKPE employs a chunking network to identify high-quality phrases and a ranking network to learn their saliency in the document. The model is trained jointly on the chunking task and the ranking task, balancing the estimation of keyphrase quality and saliency. Experiments on two benchmarks demonstrate JointKPE’s robust effectiveness with different BERT variants. Our analyses show that JointKPE has advantages in predicting long keyphrases and extracting phrases that are not entities but also meaningful. The source code of this paper can be obtained from <https://github.com/thunlp/BERT-KPE>.

## 1 Introduction

**KeyPhrase Extraction** (KPE) systems automatically extract important and topical phrases from documents, either to provide users quick summarizations of, e.g., scientific papers, or to benefit downstream applications such as document indexing, summarization, and recommendation (Turney, 2000; Meng et al., 2017; Xiong et al., 2019; Hasan and Ng, 2014; Pudota et al., 2010; Chen et al., 2018; Ye and Wang, 2018). A unique and interesting challenge of the keyphrase extraction task is the mixture of local sequence level modeling, to obtain high-quality, self-contained, meaningful phrases, and the global saliency modeling, to find the phrases that are important to the document’s information (Liu et al., 2010; Hasan and Ng, 2014).

The recent adoption of pretrained language models in KPE, e.g. ELMo (Peters et al., 2018), mainly leverages the contextualized embeddings to *chunk* high-quality phrases; the saliency ranking relies

more on frequency signals (Xiong et al., 2019). This favors head-ish phrases that appear frequently in the document and may implicitly bias towards shorter phrases, while in many scenarios, tail-ish and longer phrases also convey representative information of the documents (Hasan and Ng, 2014).

This paper presents BERT-JointKPE, which jointly learns to chunk self-contained phrases and to estimate their saliency in a multi-task setting. Starting from BERT representations, JointKPE uses CNN to compose n-gram embeddings, a chunking network to identify high-quality phrases, and a ranking network to pick those most salient in the document. During learning, the chunking network uses phrase level loss for the meaningfulness of n-grams, the ranking network uses learning to rank loss for saliency estimation, and the two tasks are combined to balance the phrase quality and saliency.

Experiments on two keyphrase extraction benchmarks, OpenKP (Xiong et al., 2019) with Bing web pages and KP20K (Meng et al., 2017) with scientific papers, demonstrate BERT-JointKPE’s state-of-the-art (SOTA) and robust effectiveness with three pretrained BERT variants: BERT, SpanBERT, and RoBERTa (Devlin et al., 2019; Joshi et al., 2019; Liu et al., 2019). **Our studies find JointKPE thrives on extracting long and non-entity keyphrases, which were challenging for previous KPE techniques.** Our ablation study confirms JointKPE’s effectiveness mainly attributes to the joint estimation of keyphrase quality and saliency in the multi-task learning.

## 2 Methodology

BERT-JointKPE takes a document  $D = \{w_1, \dots, w_i, \dots, w_n\}$  as input and learns to extract keyphrases  $p$  from its n-grams that are self-contained and salient units in the document.

Model	OpenKP									KP20k	
	F1@1	<b>F1@3</b>	F1@5	P@1	P@3	P@5	R@1	R@3	R@5	F1@5	F1@10
<b>Baselines</b>											
CopyRNN	0.217	0.237	0.210	0.288	0.185	0.141	0.174	0.331	0.413	0.327	0.278
DivGraphPointer	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	<b>0.368</b>	<b>0.292</b>
BLING-KPE	0.267	0.292	0.209	0.397	0.249	0.149	0.215	0.391	0.391	n.a.	n.a.
LLbeBack	<b>0.349</b>	<b>0.341</b>	<b>0.246</b>	<b>0.519</b>	<b>0.297</b>	<b>0.178</b>	<b>0.281</b>	<b>0.438</b>	<b>0.438</b>	n.a.	n.a.
<b>Our BERT (Base)</b>											
BERT-SpanKPE	0.318	0.332	0.289	0.476	0.285	0.209	0.253	0.436	0.521	0.393	0.325
BERT-ChunkKPE	0.340	0.356	0.311	0.511	0.306	0.225	0.271	0.464	0.558	0.412	0.337
BERT-TagKPE	0.321	0.361	0.314	0.484	0.312	0.227	0.255	0.469	0.563	0.407	0.335
BERT-RankKPE	0.342	0.374	<b>0.325</b>	0.513	0.323	<b>0.235</b>	0.273	0.489	0.582	<b>0.413</b>	<b>0.340</b>
<b>BERT-JointKPE</b>	<b>0.349</b>	<b>0.376</b>	<b>0.325</b>	<b>0.521</b>	<b>0.324</b>	<b>0.235</b>	<b>0.280</b>	<b>0.491</b>	<b>0.583</b>	0.411	0.338
<b>BERT Variants (Base)</b>											
SpanBERT-ChunkKPE	0.348	0.372	0.324	0.523	0.321	0.235	0.278	0.486	0.581	0.411	0.338
SpanBERT-RankKPE	0.355	0.380	0.331	0.530	0.327	0.240	0.284	0.497	0.593	0.412	0.338
<b>SpanBERT-JointKPE</b>	<b>0.359</b>	<b>0.385</b>	<b>0.336</b>	<b>0.535</b>	<b>0.331</b>	<b>0.243</b>	<b>0.288</b>	<b>0.504</b>	<b>0.603</b>	<b>0.416</b>	<b>0.340</b>
RoBERTa-ChunkKPE	0.355	0.373	0.324	0.533	0.322	0.235	0.283	0.486	0.581	0.408	0.337
RoBERTa-RankKPE	0.361	0.390	0.337	0.538	<b>0.337</b>	0.244	0.290	0.509	0.604	0.417	0.343
<b>RoBERTa-JointKPE</b>	<b>0.364</b>	<b>0.391</b>	<b>0.338</b>	<b>0.543</b>	<b>0.337</b>	<b>0.245</b>	<b>0.291</b>	<b>0.511</b>	<b>0.605</b>	<b>0.419</b>	<b>0.344</b>

Table 1: Overall keyphrase extraction accuracy. OpenKP scores are from the official leaderboard with a blind test and **F1@3** is the main evaluation metric (marked in bold). The baseline evaluation results on KP20K of the extractive setting are obtained from corresponding papers.

This is achieved by two main components: **a chunking network**, which identifies meaningful n-grams, and **a ranking network**, which assigns salience scores to phrases. Both networks use *n-gram representations* built upon the contextualized *token embeddings* in BERT.

**Token Embedding.** JointKPE uses BERT to encode  $D$  to a sequence of vectors  $\mathbf{H} = \{\mathbf{h}_1, \dots, \mathbf{h}_i, \dots, \mathbf{h}_n\}$ :

$$\mathbf{H} = \text{BERT}\{w_1, \dots, w_i, \dots, w_n\}, \quad (1)$$

where  $\mathbf{h}_i$  is the contextualized embedding of  $w_i$  from the last transformer layer.

**N-gram Representation.** Then JointKPE composes the contextualized embeddings to n-gram representations using CNNs. The representation of the  $i$ -th  $k$ -gram  $c_i^k = w_{i:i+k-1}$  is calculated as:

$$\mathbf{g}_i^k = \text{CNN}^k\{\mathbf{h}_i, \dots, \mathbf{h}_{i+k-1}\}. \quad (2)$$

It uses  $\text{CNN}^k$ , a set of CNNs with window size  $k$  ( $1 \leq k \leq K$ ) to compose the token embeddings to n-gram representations.

**Chunking Network** directly predicts whether the n-gram is a keyphrase based on its representation  $\mathbf{g}_i^k$  (Xiong et al., 2019). The keyphrase probability of n-gram  $c_i^k$  is calculated by a linear layer:

$$P(c_i^k = y_i^k) = \text{softmax}(\text{Linear}(\mathbf{g}_i^k)), \quad (3)$$

where  $y_i^k$  is the binary classification output on the n-gram  $c_i^k$ : is keyphrase ( $y_i^k = 1$ ), or not ( $y_i^k = 0$ ).

**Ranking Network** uses a linear layer on the n-gram representations  $\mathbf{g}_i^k$  to generate the salience

scores for the n-gram  $c_i^k$ :

$$f(c_i^k, D) = \text{Linear}(\mathbf{g}_i^k). \quad (4)$$

The same phrase may appear multiple times and correspond to multiple n-grams in the documents. JointKPE gathers these n-gram scores of the same phrase  $f(c, D)$  to the phrase score  $f^*(p, D)$ .

Specifically, for the phrase  $p^k$  with length of  $k$ , JointKPE first gathers all  $k$ -grams in the document that are exactly the same with the phrase  $\{c_j^k, \dots, c_l^k, \dots, c_m^k\}$ , and then merges their scores using max-pooling to obtain  $p^k$ 's salience score:

$$f^*(p^k, D) = \max\{f(c_j^k, D), \dots, f(c_l^k, D), \dots, f(c_m^k, D)\}. \quad (5)$$

**Joint Training.** BERT-JointKPE is trained jointly on the chunking loss and ranking loss, to better consider both the quality and salience of the phrases when extracting. The multi-task loss is the linear combination of the two loss:

$$L = L_{\text{Chunk}} + L_{\text{Rank}}. \quad (6)$$

Both the chunking loss ( $L_{\text{Chunk}}$ ) and the ranking loss ( $L_{\text{Rank}}$ ) are formulated using the ground truth keyphrase labels:  $\hat{P} = \{\hat{p}_1, \dots, \hat{p}_j, \dots, \hat{p}_m\}$ , the ground truth keyphrases of the document  $D$ .

To obtain *chunking labels*, we consider n-grams that can match any keyphrases in  $\hat{P}$  as the positive set  $C_+$ , and then calculate the chunking label of n-gram  $c_i^k$  as:

$$y_i^{k*} = 1, \text{ if } c_i^k \in C_+; y_i^{k*} = 0, \text{ otherwise.} \quad (7)$$

The chunking loss is the cross-entropy loss:

$$L_{\text{Chunk}} = \text{CrossEntropy}(P(c_i^k = y_i^{k*})). \quad (8)$$

To obtain *ranking labels*, we put all phrases in the document that are labeled as keyphrase, in the positive set  $P_+$ , and the others to the negative set  $P_-$ . The ranking loss is the standard hinge loss in pairwise learning to rank:

$$L_{\text{Rank}} = \sum_{p_+, p_- \in D} \max(0, 1 - f^*(p_+, D) + f^*(p_-, D)). \quad (9)$$

It enforces BERT-JointKPE to rank the keyphrases  $p_+$  ahead of the non-keyphrases  $p_-$  within the same document  $D$ .

### 3 Experimental Methodology

This section presents our experimental methods.

**Dataset.** Two datasets are used in our experiments, OpenKP (Xiong et al., 2019), the web KPE benchmark, and KP20K (Meng et al., 2017), the scientific paper KPE benchmark. OpenKP includes open domain web pages of various topics from Bing, and is our main dataset. In OpenKP, we follow the official split of training (134k documents), development (6.6k) and testing (6.6k). In KP20K, we follow the original work’s partition of training (528K documents), development (20K) and testing (20K) sets (Meng et al., 2017).

**Evaluation Metrics.** Precision (P), Recall (R) and F-measure (F1) of the top  $N$  keyphrase predictions are used. Following the prior research (Xiong et al., 2019; Meng et al., 2017), we use  $N = \{1, 3, 5\}$  on OpenKP and  $N = \{5, 10\}$  on KP20K.

**Baselines.** Eight models are compared: the top two on the OpenKP leaderboard, BLING-KPE (Xiong et al., 2019) and LLbeBack, two state-of-the-arts on KP20K, CopyRNN (Meng et al., 2017) and DivGraphPointer (Sun et al., 2019), as well as our two implementations, including BERT span extraction (BERT-SpanKPE) and BERT Sequence Tagging (BERT-TagKPE), SpanKPE predicts the start and end positions of keyphrases. In addition, two sub-networks of JointKPE are also considered: ChunkKPE, its chunking network, and RankKPE, its ranking network.

**Implementation Details.** The base version models of BERT, SpanBERT and RoBERTa, initialized from their pretrained weights, are used in BERT-JointKPE. All our methods are optimized using Adam with  $5e-5$  learning rate, 10% warm-up proportion, and 64 batch size. We use maximum

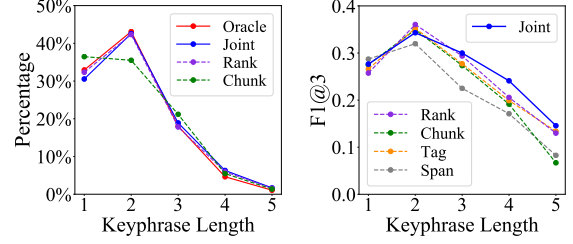


Figure 1: Keyphrase length analyses. Figure 1(a) shows the keyphrase length distribution of different models. Figure 1(b) shows model’s performance with different keyphrase length. F1@3 is used to evaluate the KPE performance. We abbreviate JointKPE, RankKPE, ChunkKPE, TagKPE and SpanKPE as Joint, Rank, Chunk, Tag and Span, respectively.

sequence length 512 and maximum phrase length five ( $K = 5$ ). The training used 2 Tesla T4 GPUs and took about 25 hours. Our implementation is based on PyTorch-Transformers.

## 4 Results And Analysis

This section presents the evaluation results and analyses of BERT-JointKPE in different scenarios.

### 4.1 Overall Accuracy

Table 1 presents the overall evaluation results. BERT-JointKPE outperforms all baselines on all evaluation scenarios of two datasets.

Compared to the best published baselines, BLING-KPE (Xiong et al., 2019) and DivGraphPointer (Sun et al., 2019), JointKPE improves all metrics by more than 6%. Notably, the F1@3 of JointKPE on the open-domain OpenKP dataset is 28% higher than the previous SOTA BLING-KPE.

BERT-JointKPE also outperforms other parallel works using pretrained transformers, including BERT-SpanKPE, BERT-TagKPE and LLbeBack. LLbeBack is based on T5 (Raffel et al., 2019).

Compared with ChunkKPE and RankKPE, the individual chunking network and ranking network, JointKPE performs stably better on all metrics on all datasets, demonstrating the effectiveness of our multi-task learning and the benefits of conducting chunking and salience ranking jointly.

JointKPE’s effectiveness is further improved with initialized from SpanBERT (Joshi et al., 2019) and RoBERTa (Liu et al., 2019), two recent variants with updated pretraining. The gains of switching from BERT to well-trained RoBERTa are consider-

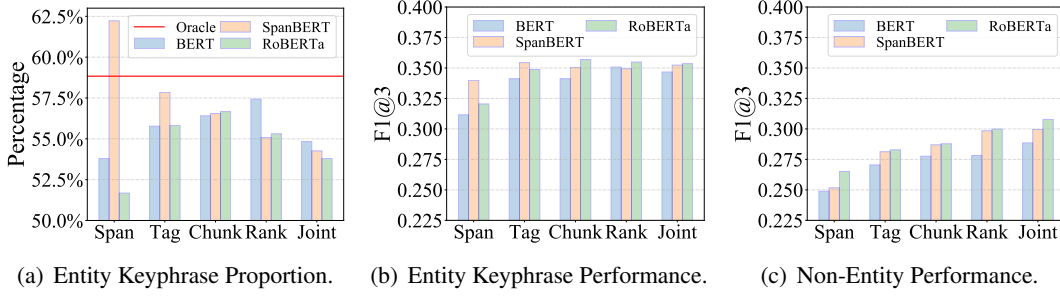


Figure 2: Keyphrase type analyses. Figure 2(a) shows the proportion of entity keyphrases. The red line indicates the oracle entity proportion of the ground truth keyphrases. Figure 2(b) and Figure 2(c) shows the model performance for entity keyphrases and non-entity keyphrases, respectively. We abbreviate JointKPE as Joint, etc.

able, showing the advantages of better pre-training and JointKPE’s ability to utilize them.

#### 4.2 Performance w.r.t. Keyphrase Lengths

One challenge in previous keyphrase extraction systems is the bias towards short keyphrases (Xiong et al., 2019; Meng et al., 2017). This experiment studies the behavior and effectiveness of JointKPE in extracting keyphrases of different lengths.

Figure 1(a) shows the length distribution of keyphrases predicted by our JointKPE and its two sub-networks, ChunkKPE and RankKPE. The length distributions of keyphrases extracted by JointKPE and RankKPE are consistent with the distributions of ground truth, while ChunkKPE inclines to predict more single-word keyphrases. The results indicate that the only considering local information tends to extract shorter phrases which are self-contained sub-phrases in long keyphrases and may require global information from the entire document to decide the boundary.

Figure 1(b) illustrates the F1@3 scores of different methods on different keyphrase lengths. The effectiveness of the models differs mainly on longer keyphrases (length  $\geq 3$ ). Among them, JointKPE performs the best on long keyphrases; its accuracy is 17% better than its sub-networks on keyphrase with length 4. The joint learning helps alleviate the challenge of extracting long keyphrases.

#### 4.3 Performance w.r.t. Keyphrase Types

A main difference between keyphrase and salient entities is that not all keyphrases are knowledge graph entities. The non-entity keyphrases with various forms are also meaningful units but more complicated to identify. This experiment studies the effectiveness of our models on non-entity phrases.

Figure 2(a) shows the proportion of entities in la-

beled keyphrases and in extracted keyphrases from different methods, with entities identified by the CMNS linker (Xiong et al., 2016). About 40% ground truth keyphrases are not entities, a significant portion of the dataset. Other than SpanKPE (SpanBERT) which heavily biases towards entities, the other methods actually extract more non-entity keyphrases than in ground truth.

Figure 2(b) and Figure 2(c) show the accuracy of different methods on entity keyphrases and non-entity ones. All methods perform well in extracting entity keyphrases. The pretrained BERT might already capture the entity information and simple layers upon it effectively identify entity keyphrases. However, their accuracy drops significantly on high-variant non-entity keyphrases; while JointKPE and RankKPE outperform others significantly. Explicitly modeling the salience of phrases using learning to rank helps overcome some complexity of non-entity keyphrase extraction.

### 5 Conclusion

This paper proposes BERT-JointKPE, a multi-task model built upon BERT that jointly learns the chunking of self-contained phrases and the estimation of their salience in the document. Two sub-networks are added upon BERT’s pre-trained transformer representations and the training combines losses from both n-gram chunking and pairwise phrase salience ranking.

In our experiments, BERT-JointKPE provides state-of-the-art keyphrase extraction on both the web domain and the scientific domain, without any domain specific adaptation. The joint learning in JointKPE effectively leverages the advantage of different BERT pretraining variants, and provides a more balanced and robust performance on keyphrases of different lengths and variant types.

## References

- Jun Chen, Xiaoming Zhang, Yu Wu, Zhao Yan, and Zhoujun Li. 2018. Keyphrase generation with correlation constraints. In *Proceedings of EMNLP*, pages 4057–4066.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL*, pages 4171–4186.
- Kazi Saidul Hasan and Vincent Ng. 2014. Automatic keyphrase extraction: A survey of the state of the art. In *Proceedings of ACL*, pages 1262–1273.
- Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2019. Spanbert: Improving pre-training by representing and predicting spans. *arXiv preprint arXiv:1907.10529*.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.
- Zhiyuan Liu, Wenyi Huang, Yabin Zheng, and Maosong Sun. 2010. Automatic keyphrase extraction via topic decomposition. In *Proceedings of EMNLP*, pages 366–376. Association for Computational Linguistics.
- Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. 2017. Deep keyphrase generation. In *Proceedings of ACL*, pages 582–592.
- 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*, pages 2227–2237.
- Nirmala Pudota, Antonina Dattolo, Andrea Baruzzo, Felice Ferrara, and Carlo Tasso. 2010. Automatic keyphrase extraction and ontology mining for content-based tag recommendation. *International Journal of Intelligent Systems*.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.
- Zhiqing Sun, Jian Tang, Pan Du, Zhi-Hong Deng, and Jian-Yun Nie. 2019. Divgraphpointer: A graph pointer network for extracting diverse keyphrases. In *Proceedings of SIGIR*, pages 755–764.
- Peter D Turney. 2000. Learning algorithms for keyphrase extraction. *Information retrieval*, 2(4):303–336.
- Chenyan Xiong, Jamie Callan, and Tie-Yan Liu. 2016. Bag-of-entities representation for ranking. In *Proceedings of the sixth ACM International Conference on the Theory of Information Retrieval (ICTIR 2016)*, pages 181–184. ACM.
- Lee Xiong, Chuan Hu, Chenyan Xiong, Daniel Campos, and Arnold Overwijk. 2019. Open domain web keyphrase extraction beyond language modeling. In *Proceedings of EMNLP*, pages 5178–5187.
- Hai Ye and Lu Wang. 2018. Semi-supervised learning for neural keyphrase generation. In *Proceedings of the EMNLP*, pages 4142–4153.



## A Entity Keyphrase vs Non-Entity Keyphrase

As shown in Table 2, the keyphrases manually labeled in OpenKP include many non-entity phrases. These non-entity keyphrases are also meaningful units but may be more complicated to identify.

Entity Keyphrase
Facebook (BUSINESS.BRAND)
The Happy Wanderer (MUSIC.COMPOSITION)
Health and Human Services (GOVERNMENT.AGENCY)
Non-Entity Keyphrase
Human Development Index (RARE ENTITY)
Join A World (VERB PHRASE)
Beautiful Clothing (ADJECTIVE PHRASE)
Opposite A Monkey (PREPOSITIONAL PHRASE)
Firing Wice With Double Click (COMPLEX PHRASE)
Landmarks And Historic Buildings (COMPOSITION PHRASE)

Table 2: Examples of entity and non-entity keyphrases.

## B Case Study

Figure 3 shows some examples of long keyphrases and non-entity keyphrases.

The first case shows that BERT-JointKPE successfully extracted the long keyphrase “Mosquito Control Board Race”, while its two sub-networks BERT-RankKPE and BERT-ChunkKPE mainly extracted short ones.

In the second case, all three models have extracted the entity keyphrase “Firewire port”, but the non-entity keyphrase “DV to Laptop” was only extracted by BERT-JointKPE.

### Long Keyphrase Example

**Document :** Walter Lagraves on Mosquito Control Board Race I will not vote for by Walter Lagraves Dear Editor Ralph DePalma is a candidate for the Board of the Florida Keys Mosquito Control Authority MC ...

@ **Joint :** Walter Lagraves ; Mosquito Control Board Race ; Mosquito Control  
 @ **Rank :** Mosquito Control ; Walter Lagraves ; Ralph DePalma  
 @ **Chunk :** Walter Lagraves ; Ralph DePalma ; Mosquito Control Board

### Non-Entity Keyphrase Example

**Document :** How to transfer DV to Laptop without having a Firewire port Shake the Future 134K 940161 views 1379 145 Published on Apr 15 2012 Pimp Your USB Speakers and Add Bluetooth ...

@ **Joint :** Firewire port ; DV to Laptop ; DV  
 @ **Rank :** Firewire port ; Laptop ; DV  
 @ **Chunk :** Firewire port ; DV ; Laptop

Figure 3: Examples of long and non-entity keyphrases. Ground truth keyphrases are underlined in the document. Top three predictions of Joint (JointKPE), Rank (RankKPE), and Chunk (ChunkKPE) are listed. The blue keyphrases are extracted by all methods; red keyphrases are only by JointKPE.