EmpLite: A Lightweight Sequence Labeling Model for Emphasis Selection of Short Texts

Vibhav Agarwal, Sourav Ghosh, Kranti Chalamalasetti, Bharath Challa, Sonal Kumari, Harshavardhana, Barath Raj Kandur Raja

{vibhav.a, sourav.ghosh, kranti.ch, bharath.c, sonal.kumari, harsha.vp, barathraj.kr}@samsung.com

Samsung R&D Institute Bangalore, Karnataka, India 560037

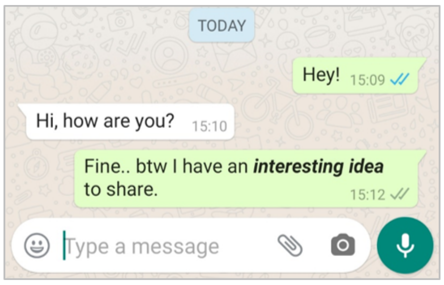
Abstract

Word emphasis in textual content aims at conveying the desired intention by changing the size, color, typeface, style (bold, italic, etc.), and other typographical features. The emphasized words are extremely helpful in drawing the readers' attention to specific information that the authors wish to emphasize. However, performing such emphasis using a soft keyboard for social media interactions is time-consuming and has an associated learning curve. In this paper, we propose a novel approach to automate the emphasis word detection on short written texts. To the best of our knowledge, this work presents the first lightweight deep learning approach for smartphone deployment of emphasis selection. Experimental results show that our approach achieves comparable accuracy at a much lower model size than existing models. Our best lightweight model has a memory footprint of 2.82 MB with a matching score of 0.716 on SemEval-2020 (shallowLearner, 2020) public benchmark dataset.

Index terms: emphasis selection, mobile devices, natural language processing, on-device inferencing, deep learning.

1 Introduction

Emphasizing words or phrases is commonly performed to drive a point strongly and/or to highlight the key terms and phrases. While speaking, speakers can use tone, pitch, pause, repetition, etc. to highlight the core of a speech in the minds of an audience. Similarly, while writing or messaging, authors can emphasize the words by customizing the formatting like typeface, font size, bold, italic, font color, etc. as illustrated in Figure 1. With the


TODAY

Hey! 15:09

Hi, how are you? 15:10

Fine.. btw I have an ***interesting idea*** to share. 15:12

Figure 1: Prominent words in a message are being emphasized (**Bold** + *Italic*) automatically

explosion of social media and messaging platforms, word emphasis has become more critical in engaging readers' attention and conveying the author's message in the shortest possible time.

Emphasis selection of text has recently emerged as a focus of research interest in natural language processing (NLP). The goal of emphasis selection is to automate the identification of words or phrases that bring clarity and convey the desired meaning. Automatic emphasis selection can help in better graphic designing and presentation applications, as well as can enable voice assistants and digital avatars to realize expressive text-to-speech (TTS) synthesis. High-quality emphasis selection models can enable automatic design assistance for creating flyers, posters and accelerate the workflow of design programs such as Adobe Spark (Adobe, 2016), Microsoft PowerPoint, etc. These emphasis selection models can also empower digital avatars like Samsung Neon (NEON, 2020) to achieve human-like TTS systems. Understanding emphasis selection is also crucial for many downstream applications in NLP tasks including text summarization, text categorization, information retrieval, and opinion mining.

In the current work, we propose a novel lightweight neural architecture for automatic emphasis selection in short texts, which can perform inference in a low-resource constrained environment like a smartphone. Our proposed architecture achieves near-SOTA performance, with as low as 0.6% of its model size.

2 Related Work

Table 1: Keyword Extraction (MonkeyLearn, 2020) vs. Emphasis Selection

Input Text	**Keywords/Key phrases Detected**	**Emphasis Selection**
A simple I love you means more than money	A simple I love you means more than **money**	A simple **I love you** means more than money
Traveling – It leaves you speechless then turns you into story teller	Traveling – It leaves you **speechless** then turns you into **story teller**	**Traveling** – It leaves you **speechless** then turns you into **story teller**
Challenges are what make life more interesting and overcoming them is what makes life meaningful	**Challenges** are what make life more **interesting** and overcoming them is what makes life **meaningful**	**Challenges** are what make life more **interesting** and **overcoming** them is what makes life **meaningful**

Prior work in NLP literature towards identifying important words or phrases have focused widely on *keyword or key-phrase extraction*. Considerable progress has been made in keyword or key-phrase extraction systems for long documents such as news articles, scientific publications, etc. (Rose et al., 2010). The core operation procedure of these systems is to extract the nouns and noun phrases. To achieve these, researchers have used statistical co-occurrence (Matsuo and Ishizuka, 2004), SVM (Zhang et al., 2006), CRF (Zhang, 2008), graph-based extraction (Litvak and Last, 2008), etc. Recent efforts have even expanded the idea from a set of documents to social big data (Kim, 2020). However, in the context of short texts like text messages, headlines, or quotes, keyword extraction systems often mislabel most nouns as important without considering the essence of the text, thus performing poorly at the task.

Emphasis selection aims to overcome this by scoring words which properly capture the essence of a text by focusing on subtle cues of emotions, clarifications, and words that capture readers' attention, as seen in Table 1. Recent research interest towards these tasks often uses label distribution learning (Shirani et al., 2019). MIDAS (Anand et al., 2020) uses label distribution as well as contextual embeddings. One drawback of using label distribution learning is the requirement of annotations, which are not readily available in most datasets. Pre-trained language model has also been used to achieve emphasis selection (Huang et al., 2020). Singhal et al. (Singhal et al., 2020) achieves significantly good performance with (a) Bi-LSTM + Attention approach, and (b) Transformers approach. To achieve their modest performances, these architectures produce huge models. For instance, IITK model (Singhal et al., 2020) takes up 469.20 MB in BiLSTM + Attention approach, while requiring almost 1.5 GB in Transformers approach. This is partly due to the use of embeddings like BERT (1.2 GB) (Devlin et al., 2018), XLNET (1.34 GB) (Yang et al., 2019), RoBERTa (1.3 GB) (Liu et al., 2019), etc. General-purpose models that emphasize on model size still consume significant ROM: 200 MB (for DistilBERT (Sanh et al., 2019)) and 119 MB (for MobileBERT (Sun et al., 2020b) quantized int8 saved model and variables; sequence length 384). Thus, in spite of the performance benefits, these emphasis selection systems with high-memory footprints are not suitable for the storage specifications of mobile devices.

Thus, while keyword extraction systems are not suitable for short text content, emphasis selection systems perform much better at such tasks. However, most existing architectures of the latter are not light-weight, and thus, not suitable for on-device inferencing on low-resource devices. This motivates us to propose EmpLite, which (a) outperforms keyword extraction systems by using emphasis selection for use with short texts, and (b) differs from existing emphasis selection architectures by ensuring a very light-weight model for efficient on-device inferencing on mobile devices. Our decisions towards achieving low model size include using a subset of GloVe (Pennington et al., 2014) word embeddings, thus, reducing embedding size from 347.1 MB to 2.5 MB, which we discuss in section 4.1.

Table 2: A short text example from dataset along with its nine annotations

Word	A1	A2	A3	A4	A5	A6	A7	A8	A9	Freq [B|I|O]	Emphasis Prob (B+I)/(B+I+O)
Kindness	B	B	B	O	O	O	B	B	B	6|0|3	0.666
is	O	O	O	O	O	O	I	I	O	0|2|7	0.222
like	O	O	O	O	O	O	I	I	O	0|2|7	0.222
snow	O	O	B	O	O	O	I	I	O	1|2|6	0.333

3 Data Collection

We use the officially released SemEval-2020 dataset (RiTUAL-UH, 2020), which is the combination of Spark dataset (Adobe, 2016) and Wisdom Quotes dataset (Quotes, 2020). The dataset consists of 3,134 samples labeled for token-level emphasis by multiple annotators. There are 7,550 tokens with fewer than 10 words in a sample and they are randomly divided into training (70%), development (10%), and test (20%) sets by the organizers. Table 2 shows a short text example from the training set, annotated with the BIO annotations, where 'B (beginning) / I (inside)' and 'O (outside)' represent emphasis and non-emphasis words, respectively, as decided by an annotator. The last column shows the emphasis probability for a word, computed as (B+I) divided by the total number of annotators, i.e. 9. We generate data labels for model training using emphasis probabilities by assigning 0 to low emphasis words (having probability $<$ threshold$_{prob}$) and 1 to high emphasis words (probability $\geq$ threshold$_{prob}$). We experiment with different values for threshold$_{prob}$ and observe that 0.4 yields the best results.

3.1 Evaluation Metric

The evaluation metric for our problem is defined as follows:

Match$_m$ (shallowLearner, 2020): For each instance x in the test set D_{test}, we select a set $S_m^{(x)}$ of $m \in (1..4)$ words with the top m probabilities with high emphasis according to the ground truth. Analogously, we select a prediction set $\hat{S}_m^{(x)}$ for each m, on the basis of the predicted probabilities. We define matching score, or Match$_m$, as:

$$\text{Match}_m = \frac{\sum_{x \in D_{\text{test}}} \left| S_m^{(x)} \cap \hat{S}_m^{(x)} \right| / m}{|D_{\text{test}}|} \quad (1)$$

Then, we compute the average rank score by averaging all possible Match$_m$ scores:

$$\text{Average Score} = \frac{\sum_{m \in (1..4)} \text{Match}_m}{4} \quad (2)$$

3.2 Data Augmentation

There are only 3,134 annotated samples in the training data, which makes it difficult to improve the accuracy with our neural model. So, to enlarge the amount of training data, we experiment with four data augmentation strategies (Sun et al., 2020a):

1. Randomly removing ≤ 1 word per sentence,
2. Randomly removing ≥ 1 word per sentence,
3. Upper-casing a word randomly, and
4. Reversing the sentence.

The effect of these techniques on our accuracy metric is presented in Section 5.1.

4 System Overview

We begin with a basic model and enhance that model with contextual information (in the form of pre-trained embeddings, char-level embeddings, Parts of Speech Tag concatenation, etc.). We describe the key components (layers) of our final EmpLite neural network architecture, as illustrated in Figure 2.

4.1 Character and Word level features

A combination of word-level and character-level input representations has shown great success for several NLP tasks (Liang et al., 2017). This is because word representation is suitable for relation classification, but it does not perform well on short, informal, conversational texts, whereas char representation handles such informal texts very well. To take the

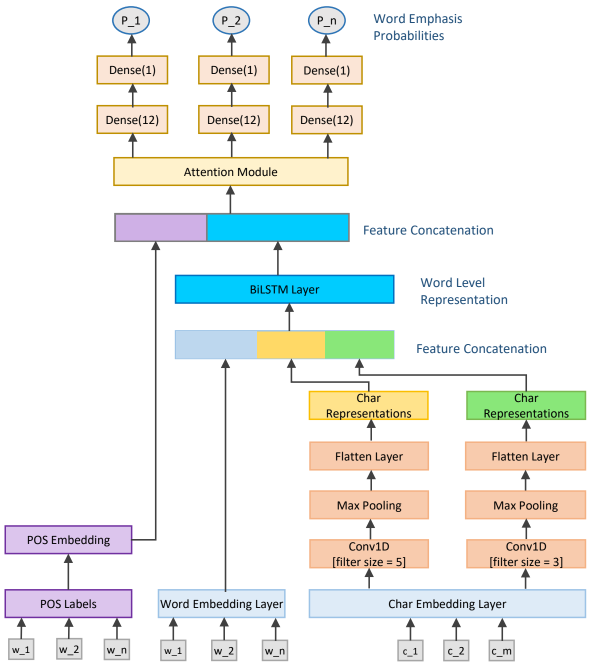


Figure 2: The proposed EmpLite Model Architecture

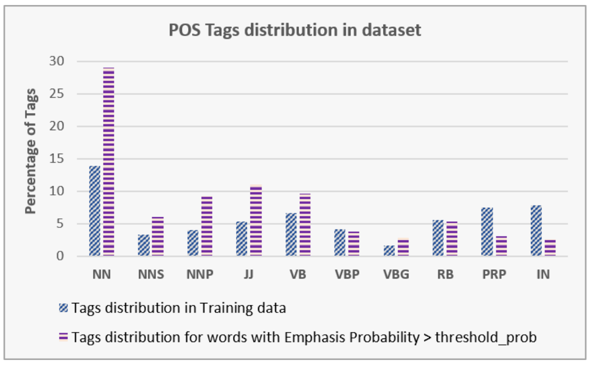


Figure 3: Percentage distribution of top POS tags in training data and for words with emphasis probability greater than threshold

best of both representations, our proposed EmpLite model employs a combination of word and character encodings for a robust understanding of context.

We used two layers of CNN (Ma and Hovy, 2016) with 1D convolutional layers of filter sizes 3 and 5 that extracts multiple character-level representations and handles misspelled words as well as models sub-word structures such as prefixes and suffixes. We use the same number of filters for both convolutional layers: 16, selected on the basis of experimental results for optimal accuracy.

The character level embeddings obtained are then concatenated with pre-trained GloVe (Pennington et al., 2014) 50-dimensional word embeddings. Here, we use a subset of the GloVe embeddings corresponding to training set vocabulary (4331 words), bringing the embedding size down to 2.5 MB. We do not use ELMo (Peters et al., 2018) or other deep contextual embeddings, as it is not feasible to port these heavy pre-trained models for on-device inferencing. In order to handle words not part of training vocabulary, we use a representation, `<UNK>` token. We set the word embedding layer as trainable as that yields the best score due to fine-tuning of layer weights on our task. The i^{th} word encoding, o_{w_i}, is computed as:

$$o_{w_i} = \text{concat}\Big(e_{w_i}, \text{CNN}_1\left(e_{c_1}, e_{c_2}, ..., e_{c_n}\right), \text{CNN}_2\left(e_{c_1}, e_{c_2}, ..., e_{c_n}\right)\Big) \quad (3)$$

where, e_{w_i} is the word embedding for each word, w_i, in the dataset and e_{c_i} is the character embedding for the input character c_i.

4.2 Word level BiLSTM

The concatenated word representations obtained are then passed through a BiLSTM (Hochreiter and Schmidhuber, 1997) layer with 16 units. The BiLSTM layer extracts the features from both forward and backward directions and concatenates the output vectors from each direction. Also, regular and recurrent dropouts with value 0.2 are applied to reduce model overfitting. Let $\overrightarrow{r}$ and $\overleftarrow{r}$ be the forward and backward output states of the BiLSTM. Then, the output vector, r_b, is defined as:

$$r_b = \overrightarrow{r} \oplus \overleftarrow{r} \quad (4)$$

4.3 Part of Speech (POS) Tag feature

Figure 3 illustrates occurrence of top 10 POS Tags (Marcus et al., 1994) in our training data. We can infer that POS acts as an important input modeling feature as words with POS Tag: Noun (NN, NNP, NNS), Adjective (JJ) or Verb (VB, VBP, VBG) usually have high emphasis probability whereas Pronouns (PRP) and Prepositions (IN) are less likely to be emphasized. Therefore, we use 16-dimensional embedding to encode POS tag information,

Table 3: Comparison of different model architectures

Model	**Model Size (MB)**	**Match$_m$**				**Average Score**
		m = 1	**m = 2**	**m = 3**	**m = 4**	
Base: Word_Emb + BiLSTM + Dense Layer	1.10	0.479	0.639	0.731	0.785	0.659
Concat[**LSTM(Char_Emb)** + Word_Emb] + BiLSTM + Dense Layer	1.10	0.473	0.658	0.739	0.786	0.664
Concat[LSTM(Char_Emb) + **Word_GloVe** (Non-trainable)] + BiLSTM + Dense Layer	1.02	0.514	0.660	0.748	0.795	0.679
Concat[**CNN**(Char_Emb) + Word_GloVe (Non-trainable)] + BiLSTM + Dense Layer	1.04	0.523	0.669	0.754	0.801	0.687
Concat[CNN(Char_Emb) + Word_GloVe (**Trainable**)] + BiLSTM + Dense Layer	2.70	0.538	0.680	0.766	0.811	0.699
Concat[CNN$_1$ (Char_Emb) + **CNN$_2$ (Char_Emb)** + Word_GloVe (Trainable)] + BiLSTM + Dense Layer	2.77	0.528	0.690	0.771	0.810	0.701
Above Model + **Attention**	2.80	**0.549**	0.684	0.779	0.817	0.707
EmpLite: Above Model + **POS Feature** Concatenation	2.82	0.541	**0.698**	**0.782**	**0.823**	**0.711**

which is concatenated with the output of the Bi-LSTM layer (obtained from Equation 4):

$$\overrightarrow{h} = \text{concat}\left(r_b, e_{pos}\right) \tag{5}$$

where, e_{pos} is the POS feature embedding for the sequence.

4.4 Attention Layer

We add the attention (Vaswani et al., 2017) layer to effectively capture prominent words in the input text sequence. The attention weight is computed as the weighted sum of the output of the previous layer, as shown below:

$$Z = \text{softmax}\left(w^T\left(\tanh\left(\overrightarrow{h_1}, \overrightarrow{h_2}, ..., \overrightarrow{h_i}, ..., \overrightarrow{h_n}\right)\right)\right) \tag{6}$$

where, $\overrightarrow{h_i}$ represents output vector of the previous layer, and w^T is the transpose of the trained parameter vector.

The attention layer output is then passed through two time-distributed dense layers with 12 and 1 units, respectively, with sigmoid activation function to output emphasis probability with respect to each word.

5 Experimental Settings & Results

We attempt numerous small changes to our model to enhance the performance. We choose the hyperparameters to optimize accuracy while maintaining a small model size. As the proposed solution is for mobile devices, we have also captured a system-specific metric, the model size in MB. We use the Tensorflow framework (Abadi et al., 2016) for building the models. Table 3 shows the comparison of Match$_m$ scores across different variants of lightweight models evaluated on test data.

The total number of trainable parameters vary in the range of 21,574 to 238,620 for all the model results reported in Table 3. We train the models with 32 batch-size and compile the model using Adam optimizer (Kingma and Ba, 2014). We observe that using CNN gives a better score as compared to LSTM because varying the size of kernels (3 and 5) and concatenating their outputs allow the model to detect patterns of multiple sizes.

We can also infer that using 50-dimensional GloVe embeddings and setting it as trainable improves the overall matching score. This is because we are utilizing language semantic knowledge acquired from the pre-trained embeddings and then fine-tuning it for our task. However, there is an increase in model size due to more number of trainable parameters. Furthermore, we observe marginal gains in the matching score by adding POS tag as a feature followed by an attention layer as these help in a better identification of prominent keywords based on the context.

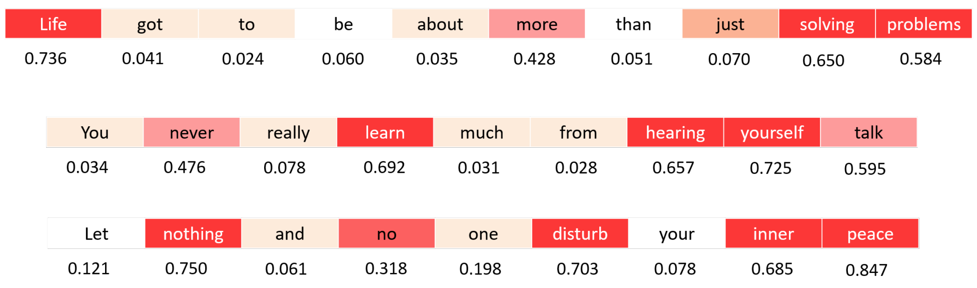


Figure 4: Emphasis Heatmap for test set samples with word probabilities from EmpLite

Table 4: Comparison with SOTA (Singhal et al., 2020)

Model	Match$_m$	Size (MB)
IITK: BiLSTM + Attention Approach	0.747	469.20
IITK: Transformers Approach	**0.804**	1536.00
EmpLite	0.716	**2.82**

Figure 4 presents the Emphasis Heatmap for some examples from the test set using our final model. In Table 4 we benchmark our EmpLite model with the state-of-the-art solution by IITK (Singhal et al., 2020) which utilized huge pre-trained models like ELMo, BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019) and XLNet (Yang et al., 2019). These models require huge RAM/ROM for on-device inferencing making it unsuitable for edge devices where resources are constrained.

Table 5: Data Augmentation Analysis

Augmentation Approach	Dataset modified (%)	Match$_m$ Score
None	0	0.711
Word removal (≤1 per sentence)	20	0.705
	50	0.702
	100	**0.716**
Word removal (≥1 per sentence)	20	0.711
	50	0.704
	60	0.712
	70	0.705
Upper-casing a word	30	0.687
Reversing the sentence	10	0.704
	100	0.707

5.1 Data Augmentation Analysis

Table 5 shows that there is a little score gain by applying data augmentation techniques. For each strategy, we experiment by applying the augmentation approach to different percentages of the total training data and calculated Match$_m$ score. We observe that word uppercasing strategy results in a significant drop in the score due to model overfitting whereas word removal strategy (maximum 1 word per sentence) on entire training data gives highest Match$_m$ score of 0.716.

6 Conclusion

Modeling lightweight neural models that can run on low-resource devices can greatly enhance the end-user experience. In this work, we propose a novel, lightweight EmpLite model for text emphasis selection that can run on edge devices such as smartphones for choosing prominent words from short, informal text. We approach the emphasis selection problem as a sequence labeling task and multiple experiments have shown consistent improvement in the accuracy. Our experimental results show the impact of the attention layer and of using POS as an additional feature in boosting the matching score. Our best performing model achieves an overall matching score of 0.716 with a size of 2.82 MB, proving its effectiveness to run on low-resource edge devices.

Future work includes increasing the vocabulary with commonly used words in English and exploring thin versions of BERT like DistilBERT (Sanh et al., 2019), MobileBERT (Sun et al., 2020b), and TinyBERT (Jiao et al., 2020) for modeling emphasis.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

Adobe. 2016. Adobe Spark. `https://spark.adobe.com`, accessed 2020-11-28.

Sarthak Anand, Pradyumna Gupta, Hemant Yadav, Debanjan Mahata, Rakesh Gosangi, Haimin Zhang, and Rajiv Ratn Shah. 2020. Midas at semeval-2020 task 10: Emphasis selection using label distribution learning and contextual embeddings. *arXiv preprint arXiv:2009.02619*.

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*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Zhengjie Huang, Shikun Feng, Weiyue Su, Xuyi Chen, Shuohuan Wang, Jiaxiang Liu, Xuan Ouyang, and Yu Sun. 2020. Ernie at semeval-2020 task 10: Learning word emphasis selection by pre-trained language model. *arXiv preprint arXiv:2009.03706*.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online. Association for Computational Linguistics.

Hyeon Gyu Kim. 2020. Efficient keyword extraction from social big data based on cohesion scoring. *Journal of the Korea Society of Computer and Information*, 25(10):87–94.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Dongyun Liang, Weiran Xu, and Yinge Zhao. 2017. Combining word-level and character-level representations for relation classification of informal text. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 43–47.

Marina Litvak and Mark Last. 2008. Graph-based keyword extraction for single-document summarization. In *Coling 2008: Proceedings of the workshop Multi-source Multilingual Information Extraction and Summarization*, pages 17–24.

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*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The penn treebank: Annotating predicate argument structure. In *Proceedings of the Workshop on Human Language Technology*, HLT '94, page 114–119, USA. Association for Computational Linguistics.

Yutaka Matsuo and Mitsuru Ishizuka. 2004. Keyword extraction from a single document using word co-occurrence statistical information. *International Journal on Artificial Intelligence Tools*, 13(01):157–169.

MonkeyLearn. 2020. Keyword Extraction: A Guide to Finding Keywords in Text. `https://monkeylearn.com/keyword-extraction/`, accessed 2020-07-03.

NEON. 2020. NEON. `https://www.neon.life/`, accessed 2020-06-24.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Wisdom Quotes. 2020. Wisdom Quotes - Get wiser slowly. `https://wisdomquotes.com`, accessed 2020-11-30.

RiTUAL-UH. 2020. SemEval2020 Task10 Emphasis Selection. `https://github.com/RiTUAL-UH/SemEval2020_Task10_Emphasis_Selection`, accessed 2020-11-22.

Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. Automatic keyword extraction from individual documents. *Text mining: applications and theory*, 1:1–20.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

shallowLearner. 2020. SemEval 2020 - Task 10: Emphasis Selection For Written Text in Visual Media. `https://competitions.codalab.org/competitions/20815`, accessed 2020-08-15.

Amirreza Shirani, Franck Dernoncourt, Paul Asente, Nedim Lipka, Seokhwan Kim, Jose Echevarria, and Thamar Solorio. 2019. Learning emphasis selection for written text in visual media from crowd-sourced label distributions. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1167–1172.

Vipul Singhal, Sahil Dhull, Rishabh Agarwal, and Ashutosh Modi. 2020. Iitk at semeval-2020 task 10: Transformers for emphasis selection. *arXiv preprint arXiv:2007.10820*.

Yu Sun, Shuohuan Wang, Yu-Kun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2020a. Ernie 2.0: A continual pre-training framework for language understanding. In *AAAI*, pages 8968–8975.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020b. Mobilebert: a compact task-agnostic bert for resource-limited devices. *arXiv preprint arXiv:2004.02984*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30:5998–6008.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763.

Chengzhi Zhang. 2008. Automatic keyword extraction from documents using conditional random fields. *Journal of Computational Information Systems*, 4(3):1169–1180.

Kuo Zhang, Hui Xu, Jie Tang, and Juanzi Li. 2006. Keyword extraction using support vector machine. In *international conference on web-age information management*, pages 85–96. Springer.