MY PAPER

August 11, 2018

Contents

List of Figures

Listings

Literature review

1 Natural Language Processing with Deep Learning

It is non-negligible thing to unified fundamental knowledge in a Deep Learning architecture from Collobert et al. (2011) [1]. The work is cover various task on Natural Language Processing (NLP) problem including part-of-speech tagging, chunking, named entity recognition (NER), and semantic role labeling. There are only two models introduced in this work which are sentence approach network and window approach network. The window approach network consider fixed size window of words around one word. Given that the tag of a word depends mainly on its neighboring, the window approach network usually provides a good result on most tasks such as part-of-speech tagging, chunking, name entity detection. However, for semantic role labeling task, a tag of each word usually depends on a verb. Given that a verb of one word could be placed outside the window, the tag of that word would be incorrect. The word approach network consists with a predicted target word and its neighbor. For example, if the window size equal to two, to predict $\mathbf{W_n}$, we would required $\mathbf{W_{n-2}}$, $\mathbf{W_{n-1}}$, $\mathbf{W_n}$, $\mathbf{W_{n+1}}$, and $\mathbf{W_{n+2}}$ as input for this network as 1D tensor of word indexes. This input is then mapped to an embedding layer which equivalence to a lookup table to replace these word indexes to 2D tensor features. The output from embedding layer then become an input for the next linear layer. Finally, the linear layer output is used as the input of a classifier layer to determined the tag of a word located in the center of the window. Figure 1 portrayed a simplest sample of a model to predict a tag of a second word ($\mathbf{W_2}$) in sentence $\mathbf{i}$ ($\mathbf{S_i}$) with window size equal to one. As for sentence approach, the whole $\mathbf{S_i}$ is used to predict each word tags using a convolution layer instead of a linear layer as illustrates in Figure 2. The convolution layer extracts features of each word in the sentence as 2D tensor. This convolution layer is introduced by Waibel et al (1989) [2] called Time Delay Neural Networks (TDNNs). Finally, each word features then used as input for the classifier layer individually to determine a tag of each input word in that sentence. This model is well-known and is proofed to be effective in various NLP tasks. Hence, it has usually been used to forge a baseline for multiple tasks in NLP research field.

2 Neural architectures for named entity recognition

To illustrate a clear picture of each work, one should views the published model into three level separately: Character-Encoder level, Word-Encoder level, and Tag-Decoder level. The process during Character-Encoder level are counted from the character feature look-up table to each word features are extracted from a sequence of characters. The Word-Encoder level is a process from a creation of each word feature to an acquisition of each word hidden feature. Finally, the Tag-Decoder level is the usage of each word hidden feature to clarify tag of each word using Conditional Random Field (CRF) proposed by Lafferty

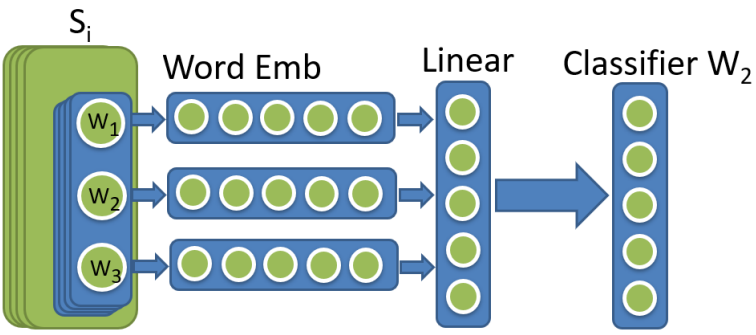

Figure 1: Window approach network.

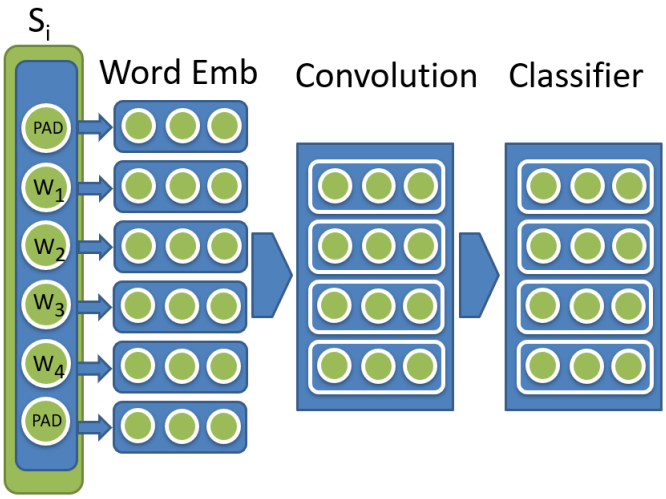

Figure 2: Sentence approach network.

et al. (2001) [3]. Since recently pioneered by Collobert et al. (2011) [1] there are several works regarding NER task especially that based on his work. Huang et al. (2015) [4] proposed a changed from Convolutional Neural Networks (CNN) proposed by Lecun et al. (1995) [5] to a Long-Short Term Memory (LSTM) proposed by Hochreiter et al. (1997) [6] in the Word-Encoder level. Introduce by Lample et al. (2016) [7], their word involve with a variant combination methods such as pretrain word using Skip-n-gram Word2Vec (Ling et al., 2015) [8], dropout (Srivastava et al., 2014) [9], LSTM , and CRF. Their approach is separated into two aspects: LSTM-CRF and Stacked LSTMs. However, the important distinction of their work is a feature extraction from character-level using Bidirectional LSTM (BiLSTM) proposed by Graves et al. (2005) [10] instead of hand-engineering feature creation such as prefix and suffix. The usage of character-level idea have been found useful for morphologically rich languages to handle an out of vocabulary problem that always occur. The character-level output are concatenate with word presentation feature to form

a full feature words which use as an input for LSTM layer of Word-Encoder level to achieve each word hidden feature. Thus, the model can made a tag prediction of one word by using previous tags and the word own hidden feature. The work has shown that the combination of LSTM-LSTM-CRF added with pre-train word embedding and dropout given the best result. Similar work is done by Chiu et al. (2016) [11] which replaced LSTM in the Character-Encoder level to CNN. Yang et al. (2016) [12] also replaced the Character-Encoder and Word-Encoder levels with a Gate Recurrent Unit (GRU) proposed by Cho et al. (2014) [13] result in a GRU-GRU-CRF model. In another aspects, the Tag-Decoder level is a main focus for Nguyen et al. (2016) [14] where the use of a recurrent neural network (RNN) found by Hopfield (1982) [15] is introduced. With this changed, when the number of entity types is enormous, the model could be train faster than CRF. Later on, it has been demonstrated by Shen et al. (2017) [16] that a usage of LSTM in Tag-Decoder level could outperform CRF. His work introduced a changed in the Tag-Decoder that usually used CRF for NER task to LSTM while combining each word with Character-Encoder level approach. Furthermore, with Active learning algorithm, a model proof to be able to manipulate its learning process by choosing contexts in training set to learn by itself, which minimize amount of sampling for training. Compare with other models, his LSTM-LSTM-LSTM achieve the highest F1-score.

3 Weighted Cross-Entropy for imbalance data

The total amount of name entity in our training data contains 190 classes excluding "B-", "I-", and "O" tags. This leads to an imbalance label problem. Furthermore, NER task is usually fill with high ratio of O-Tags. As a result, the training model will subsequently made an adjustment on O-Tags first to minimize the loss. When this problem arise, the training model is bound to stuck in its local minima. Thus, result in a model which could only predicted O-Tags correctly. Therefore, we view that the weighted calculation for each tag in our categorical Cross-Entropy is a non-trivial matter and it could be solve similarly to a rare word detection problem. There are some published which discussed about this in a keyword detection model proposed by Panchapagesan et al. (2016) [17], and Sun et al. (2017) [18].

4 Our work

From the previous studying, even through, while the model which done in a more complicate method usually provide a better result, its also raise a question that could a compact one be able to achieve such a result. Hence, an optimization of the model is the main focus of this work. In term of optimization, our main contribution lays in various aspect such as a training duration, most needed feature, and an amount of epoch for training until model converge. The author absolutely agree that the usage of Character-Encoder level cannot be neglect in the case such as Chinese characters (Kanji) could hold an important feature such as the location or person name. There are various combination of Kanji and the training corpus obviously could not suffice all of its combination. However, instead of the usage of LSTM at Character-Encoder level to a whole sentence

like the other work. The LSTM is applied at Character-Encoder level on a batch of set of word. As Japanese language has no explicit word boundary markers in every sentence, the word segmentation is quite crucial and the Character-Encoder level should not deal with the whole string directly as the relationship between character should laid in their respective word only. The main reason for not using BiLSTM on this layer is that, if the order of characters is changed, it would become a different word accordingly. Therefore, the discernment on character ordering is automatically distinguished in the Word-Encoder level. In additional, considering that usually Noun is a name entity, characters of a word which is not "Noun" could be neglected to improve training speed. The Word-Encoder level is where the model applied a handcrafted feature: part-of-speech, and partial handcrafted feature: Japanese particle augmented to the output of Character-Encoder. Finally, the Tag-Decoder layer, each tag is a final output which is a classifying result from an output of Word-Encoder and a previous tag as its LSTM input. The model is portrayed and explained in detailed in the next section.

Method

5 Corpus

The corpus used in model experiment is called “ene-corbocha-mainiti”. It consists with 8,228 news articles, 53,224 unique words, and 2,226,147 tokens. The sentence is also annotated with word segmentation, POS, and chunk dependencies that annotated by the tool called: CaboCha. Therefore, the corpus is about 7.38 times larger than CoNLL-2003 English corpus which consists of 1,393 articles with 301,418 tokens. Despite the larger corpus, there also has wide variety of name entities in total of 190 types. However, as the corpus of Japanese language is arranged and distinguished in a hierarchy following “Sekine’s extended named entity hierarchy - 7.1.0 -”. Worrying that the label could be very sparse result that model rarely learn something, we decided to group all sub-categories into one top-most of its class accordingly to represent the name entity of each name entity word. Thus, result to the amount of label types to decrease to 26 tags: **God, Percent, Location, Latitude Longtitude, Product, Ordinal Number, Name Other, Numex Other, Multiplication, School Age, Timex, Age, Natural Object, Disease, Person, Organization, Facility, Color, Money, Point, Rank, Countx, Frequency, Measurement, Event, and Period**. Furthermore, the model contains a lot of handcraft features such as Part-Of-Speech and Word-pronunciation. Also, arrange in XML format and group by sentence, chunk, and token level respectively. Data format The corpus is collected and pre-processing into XML format annotated by <sentence >, <chunk>, and <tok>tag as shown in code listing 1.

Listing 1: Corpus XML format

```
<sentence>
  <chunk>
     <tok id="0" feature="..." ne="O" ene="B-Game">WORD1</tok>
     <tok id="0" feature="..." ne="O" ene="I-Game">WORD2</tok>
     <tok id="0" feature="..." ne="O" ene="O">WORD3</tok>
  </chunk>
  <chunk>
    <tok id="0" feature="..." ne="O" ene="O">WORD4</tok>
    <tok id="0" feature="..." ne="O" ene="O">WORD5</tok>
  </chunk>
</sentence>
```

The attributes of each token consist of **id**, **features**, **ne**, and **ene** where each attribute represent by the following:

- **id:** Token ID
- **feature:** POS, base form, pronounce
- **ne:** Named entity tag with BIO format
- **ene:** Fine grained named entity tag with BIO format

However, “ne” is an automatically identified named entity by CaboCha while “ene” is a manually annotated name entity. Therefore, “ene” is the gold named entity tag. Furthermore, for each chunk of the sentence, we can extract another feature given one chunk is affected by its following Japanese particle word. The Japanese particle word could be categorized into 8 groups:

1. **Case markers 「格助詞」(Kaku-joshi) -** Represents a semantic relation in a sentence. Thus, it would provided an information about relationship between each name entity in a sentence. Case markers are included 「が、の、を、に、へ、と、で、から、より」.

2. **Parallel markers 「並立助詞」(Heiritsu-joshi) -** Align two things together. This Japanese particle would improve the ability to detect multiple name entities in a sentence. This markers included 「か、の、や、に、と、やら、なり、だの」.

3. **Sentence ending particles 「終助詞」(Shū-joshi) -** Adds meaning such as doubt, prohibition and impression about the end of sentences and phrases. This particle consist with 「か、の、や、な、わ、とも、かしら」. However, this particle not really helpful on name entity detection but could be used in a task such as sentiment analysis.

4. **Interjectory particles 「間投助詞」(Kantō-joshi) -** This particle consist with 「さ、よ、ね」and is used for give an expression. Same as Sentence ending particles, this particle is not really helpful on name entity detection but could be one emphasizing features used in other task.

5. **Adverbial particles 「副助詞」(Fuku-joshii) -** Adverb as a whole for phrases, adverbs particles, etc. This particles consists of 「ばかり、まで、だけ、ほど、くらい、など、なり、やら」.

6. **Binding particles 「係助詞」(Kakari-joshi) -** This particles is used to emphasize the meaning of attached word. It consists of 「は、も、こそ、でも、しか、さえ、だに」. Likewise, this particle given the information of subject and object in the sentence as well as provide an information of the attached word whether the previous attached word is either topic, add, or exclusive expression in the context.

7. **Conjunctive particles 「接続助詞」(Setsuzoku-joshi) -** Express the semantic relation between sentence and sentence. Same as Sentence ending particles and Interjectory particles, this particle type is not really helpful on name entity detection but could be used in other task such as finding a relationship from sequence-to-sequence. This particles consists of 「ば、や、が、て、のに、ので、から、ところが、けれども」.

8. **Phrasal particles 「準体助詞」(Juntai-joshi) -** Convert various words to a phrase as a whole. This particle usually used for grouping a same typical words together. Thus, it emphasizes the possibility of word using this particles to be the same name entity type or related. This particles consists of 「の、から」.

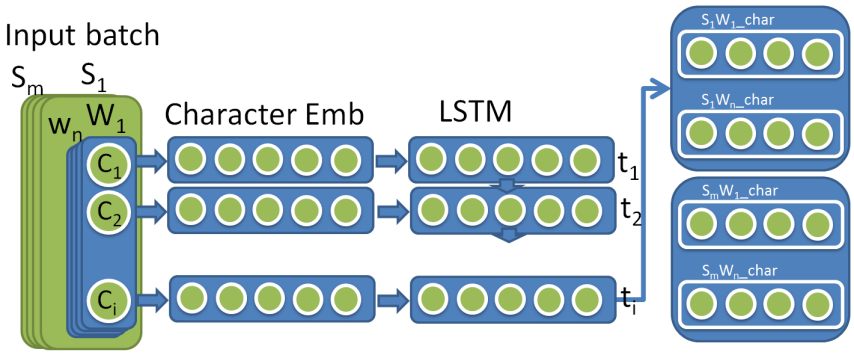

Figure 3: Character-Encoder level.

6 Model

6.1 Character-Encoder level

Input in this level are character indexes of each word from sentences. As illustrates in Figure 3, **S** stands for a sentence from **1** to **m** for one input batch. **M** stands for a word from **1** to **n** for one sentence. Finally, **C** stands for a character from **1** to **i** for one word. This input connected with a Character-Embedding level which will provide vector for each character. As the main goal is to create a small and fast learning model as much as possible, the BiLSTM could be redundant to Word-Encoder level extracted features as switching of a character ordering would automatically count as a new word. Thus, the output of Character-Embedding layer is connected to uni-direction LSTM which will accumulated and provided a vector for each word according to each sentence via it final hidden state output as $\mathbf{S_{1\text{-}m}W_{1\text{-}n}}$ char. In addition, the experiment also applied on the filtering input character. The filtering input character is the idea we proposed to improve accuracy, or at least accelerate a learning process. Given that a name entity usually be **Noun** and the task is to seek only name entities within sentences, any word which is not **Noun** could be neglect. The Character-Encoder level could be useful in such a case where there is a set of characters that could be used to identify a name entity for an organization same as "CO.", "LTD."in English. As for Japanese, 「本」character sometime could represent a location such as 「日本」(Japan) or a person/organization such as 「宮本、山本」(Muramoto, Yamamoto).

6.2 Word Encoder level

This level is applied with various combination features inputs such as Japanese particle types (JP) as mentioned in the previous section, and Part-of-speech (POS). As illustrate in Figure 4, there are several chunk in one sentence. Initially, consider input sentence of sentence **m** is $\mathbf{S_m}$, each word is an input that connected to its embedding layer to generate its vector from word **1** to **n** as $\mathbf{S_mW_{1\text{-}n}}$. Furthermore, each Part-of-speech is corresponded to each word and connected with its own embedding layer which result in $\mathbf{S_mPOS_{1\text{-}n}}$. Finally, each Japanese particle types is considered by the last word of each chunk from

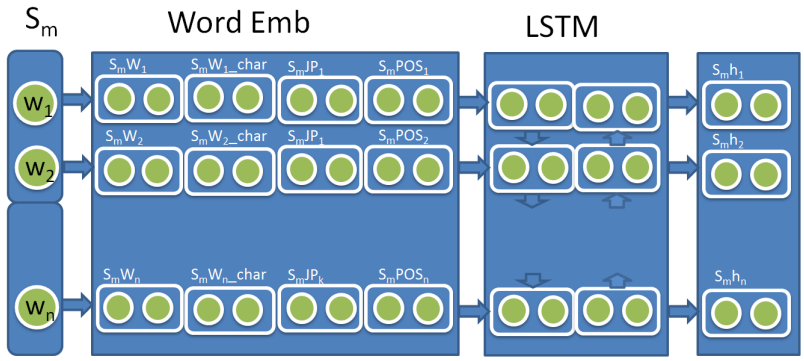

Figure 4: Word-Encoder level.

1 to **k**. The chosen Japanese particle type then past its own embedding layer result in a vector $\mathbf{S_m JP_{1\text{-}k}}$. These features are concatenated with the previous level vector $\mathbf{S_m W_{1\text{-}n}}$_char to provide a new input which will feed to BiLSTM to figure the relationship between each word from both directions. The output from Bi-directional LSTM for each word is collected as a vector $\mathbf{S_m h_{1\text{-}n}}$.

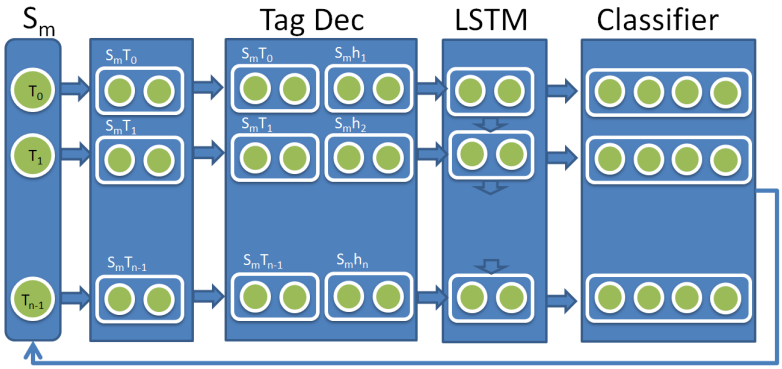

Figure 5: Tag-Decoder level.

6.3 Tag-Decoder level

Tag Decoder level is the last part in the model. Its input is a concatenate vector between the output vector of each word from the previous Word-Encoder level and the output from a tag embedding layer. However, the input of tag embedding layer is quite crafty as it is a previous timestep of its tag index. Therefore, to predict the second word's name entity of a sentence, the first word's name entity is used as a feature and so on. The concatenation of this input then pass through LSTM layer and one last linear layer to finally determine the each word's label. Hence, it means that the output of word **n** of sentence **m** is determined by tag **n-1** of sentence **m** $\mathbf{S_m T_{n\text{-}1}}$ and hidden vector n of sentence **m** ($\mathbf{S_m h_n}$).

7 Weighted categorical cross-entropy

Unlike usual name entity detection task on other work, ene-corbocha-mainiti corpus contains 26 name entity tags. As a result, the model suffers greatly from an imbalance tags problem. Also, a sampling method to adjust each label equally is likely impossible, as NLP task usually treat each input data as a sequence. Thus, chosen Weighted categorical cross-entropy as a loss function seem to be a correct logical decision. An equation used for calculation such weight is described in following:

$$W_l = \frac{\sqrt{\sum_{i=0}^{n} X_i^2}}{X_l} \tag{1}$$

Although, the weighted Cross-Entropy is not something new. The calculation of a proper weight never been defined even from a tutorial of Cross-Entropy (Boer, 2005) [19]. Provided a simplest yet effective way for weighted calculation to reduce the time consumption for training one model is also our goal to deliver a compact model. Equation 1 variable consists of $\mathbf{W_l}$ as a weight of target label and $\mathbf{X}$ as an amount of each label in a training data, $\mathbf{n}$ is the number of classes and $\mathbf{X_l}$ is an amount of that class label. Thus, the equation is an inverse function of Euclidean norm. Upon the task, the model should give high priority to the name entity detection. As name entity tags are much lower than non-name entity tags in number, this function is one of the most well-known and befit to abruptly adjust the model when the erroneous prediction occurs in minor groups.

$$loss(S, tag) = W_l[tag](-S[tag] + log(\sum_j exp(S[j]) \tag{2}$$

After achieve weight of each tag, the loss is calculated by Equation 2. Hence, $\mathbf{W_l[tag]}$ is a weight of each tag, $\mathbf{S[tag]}$ represents a score on the correct tag, and $\mathbf{S[j]}$ represents each score achieved from class $\mathbf{0}$ to class $\mathbf{j}$ where $\mathbf{j}$ is a total class exists. After summation of loss, the overall loss is divided by the number of prediction times to find an average. We believed that by this learning method. Once the model able to distinguish between minority group which obviously a name entity accurately, the other tag should not be hard to adjust later on. Therefore, the model could avoid such a local minima that could occurred when adjust itself to O-Tags.

References

[1] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. Natural language processing (almost) from scratch. *CoRR*, abs/1103.0398, 2011.

[2] Alexander H. Waibel, Toshiyuki Hanazawa, Geoffrey E. Hinton, Kiyohiro Shikano, and Kevin J. Lang. Phoneme recognition using time-delay neural networks. *IEEE Trans. Acoustics, Speech, and Signal Processing*, 37:328–339, 1989.

[3] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[4] Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991, 2015.

[5] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

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

[7] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. *CoRR*, abs/1603.01360, 2016.

[8] Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernández Astudillo, Silvio Amir, Chris Dyer, Alan W. Black, and Isabel Trancoso. Finding function in form: Compositional character models for open vocabulary word representation. *CoRR*, abs/1508.02096, 2015.

[9] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[10] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5-6):602–610, 2005.

[11] Jason P. C. Chiu and Eric Nichols. Named entity recognition with bidirectional lstm-cnns. *CoRR*, abs/1511.08308, 2015.

[12] Zhilin Yang, Ruslan Salakhutdinov, and William W. Cohen. Multi-task cross-lingual sequence tagging from scratch. *CoRR*, abs/1603.06270, 2016.

[13] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.

[14] Thien Huu Nguyen, Avirup Sil, Georgiana Dinu, and Radu Florian. Toward mention detection robustness with recurrent neural networks. *CoRR*, abs/1602.07749, 2016.

[15] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

[16] Yanyao Shen, Hyokun Yun, Zachary C. Lipton, Yakov Kronrod, and Animashree Anandkumar. Deep active learning for named entity recognition. *CoRR*, abs/1707.05928, 2017.

[17] Sankaran Panchapagesan, Ming Sun, Aparna Khare, Spyros Matsoukas, Arindam Mandal, Björn Hoffmeister, and Shiv Vitaladevuni. Multi-task learning and weighted cross-entropy for dnn-based keyword spotting. In *INTERSPEECH*, pages 760–764. ISCA, 2016.

[18] Ming Sun, Anirudh Raju, George Tucker, Sankaran Panchapagesan, Gengshen Fu, Arindam Mandal, Spyros Matsoukas, Nikko Strom, and Shiv Vitaladevuni. Max-pooling loss training of long short-term memory networks for small-footprint keyword spotting. *CoRR*, abs/1705.02411, 2017.

[19] Pieter-Tjerk de Boer, Dirk Kroese, Shie Mannor, and Reuven Y. Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67, 1 2005. Imported from research group DACS (ID number 277).