POS Tagging and Chunking with Subword2Word models

Sébastien Jean
Courant Institute
New York University
sj2233@nyu.edu

Andrew Drozdov
Courant Institute
New York University
apd283@nyu.edu

Arpit Jain
Courant Institute
New York University
aj1511@nyu.edu

Abstract

Neural network models with character-level inputs have recently proven to be well-suited for a variety of NLP tasks. In this project, we measure the effect of using various sub-word units as input instead of characters. Comparing many segmentation schemes on both part-of-speech tagging and chunking, we observe that characters are quite a strong baseline. We reach almost identical performance with some mild segmentation heuristics, but too aggressive approaches lead to worse results. Compared to the state-of-the-art, our chunking models are somewhat inferior, and comparisons to previous work indicate that it may be important to integrate hand-designed features, possibly because of the smaller size of the training data.

1 Introduction

Neural networks have been used for natural language processing tasks for more than a decade [16] and have recently grown in popularity [43]. These models generally represent words as vectors, which may be pre-trained or adapted to the task at hand. Word vectors may then be combined, automatically extracting features that will be helpful to accomplish a given task.

Models that use character-level inputs [39, 42, 41, 47, 25] rather than starting directly from words have shown promising results. One of the advantages of character-level models is their ability to handle rare or unknown words through patterns found in character frequency and context. Obviously, to learn from characters, there must be regularities in orthography or syntax, which appears to be the case for the natural languages (at least the ones that we understand).

Another alternative is to take a middle-ground approach, using subword units instead [28, 44, 22, 31]. This approach may be beneficial in some scenarios for a few reasons. If segmentation is done on the source side, every sentence would generally have less subword units than characters, so training speed may be faster. On some difficult problems, we hypothesize that such units would make the optimization problem easier. Moreover, if the segmentation is not too aggressive, such models may still deal with unknown words, as those based on characters. However, predefining subwords units also has drawbacks. In particular, a given segmentation may prevent the system to learn about useful features, for example if the interactions between two characters could be helpful for a given task [42], but they belong to different subword units.

In this project, we investigate the impact of different word segmentation techniques on two closely related tasks, part-of-speech tagging and chunking [15, 14], using the neural network architecture described by Ling et al. [42]. On these two tasks, restricting ourselves to using no additional data or features other than characters or subword units, the character-level baseline is quite hard to beat. Some segmentation heuristics reach comparable performance, although these were mostly conservative, with many short segments. More aggressive segmentations generally did not perform as well.

2 Tasks

2.1 POS tagging

For POS tagging we use the Penn WSJ treebank (PTB), a collection of 2,499 stories from the WSJ prior to 1993 [5]. This corpus contains 48 POS tags composed of 36 regular POS tags such as JJ, NN, and POS plus 12 other tags designating punctuation and symbols. This dataset is divided into 25 sections, and we've adopted the following splits which have become the norm since Michael

Collins introduced them in 2002 [15]: train (0-18), development (19-21), test (22-24).

We evaluated the models on the development set using tag-level accuracy, although others have used sentence-level and unknown word accuracy in their experiments [18].

2.2 Chunking

In addition to POS tagging, we perform chunking on the same dataset using IOB and BEISO tags. We emphasize IOB tag results as they are the format used in Conference of Computational Natural Language Learning in 2000 (Conll2000) [14].

The splits in this case are different: training (15-18), development (21), test (20). They are the same as in [15]. For the sake of comparing to other models, it is perhaps more useful that the train and test splits match the shared task of chunking from Conll2000 [14], which were originally used in [11]. The development set was not used in this shared task, but we use it for the sake of standardizing our code across both POS tagging and chunking tasks, and it is used in other chunking systems [15, 17].

IOB chunk tags have B indicate the beginning of a chunk, I the outside, and O the outside. BEISO tags are similar with the addition of two tags. E indicates the end of a chunk (previously an I), and S indicates a standalone chunk (previously a B).

The chunklink script from Conll2000 [1] uses tree structure and POS tags to generate IOB chunk tags. The PTB data does not have chunk tags, so we use the chunklink script to generate them. It is easy to go between IOB and BEISO format taking the different tag rules into account.

We could have used the part-of-speech tags provided in the Penn TreeBank, but this would have been unfair for the task of chunking, seeing that the accuracy of the part-of-speech tags will be unrealistically accurate (you could bypass learning and use the chunklink script to generate perfect chunk tags). For the Conll2000 shared task a Brill Tagger was used to generate POS tags [14] for the PTB dataset. We use a different Brill Tagger from the NLTK framework [2], which has a relatively low accuracy of 94.10% on the train set. Comparatively, we measured the Conll2000's POS tags to have 96.44% accuracy on the train set. Even though our own tagger had higher accuracy, it would have been unfair to use since it was being constantly validated against the POS development set, which overlaps with the test set of the chunking tasks.

We evaluate chunking across all tags using F-Score, which is how Conll2000 ranked competing systems [14]. F-Score is a combination of Precision and Recall, all three of which are metrics borrowed from Information Retrieval [1]. The value of Precision is the measure of predicted chunks over all chunks predicted, and Recall is the measure of correctly predicted chunks over all chunks that are correct. This evaluation is done with a script provided for the Conll2000 task. [3]

3 Models

3.1 C2W model

The networks we built for this project are based on the C2W (character-to-word) models initially used by Ling et al. for POS tagging [42]. To deal with unknown words and data sparsity, these models take as input characters instead of using words directly. Each words' characters are represented as vectors in a look-up table and are composed with a bidirectional LSTM [10, 21]. After, the final hidden state of each RNN may be combined in order to obtain a word representation.

By themselves, such word embeddings would not be sufficient to obtain a good tagger or chunker since the word representations do not depend on neighboring words and could not handle tag ambiguity. As such, another bi-LSTM composes the word embeddings to obtain such context-dependent representations. These vectors may then be transformed via a simple linear transformation followed by a softmax layer to obtain probabilities over tags or chunks.

The architecture is similar to the hierarchical RNNs of Sordoni et al. [45], who successively applied RNNs at different scales for the task of query suggestion. In their model, the first RNN processes words within a sentence, while another combines sentence representations.

3.2 Architectural changes

Rather than using the publicly available implementation of the C2W models, [4] we reimplemented it with Theano [23, 26], using the neural

[1] http://ilk.uvt.nl/team/sabine/chunklink/chunklink_2-2-2000_for_conll.pl

[2] https://github.com/japerk/nltk-trainer

[3] http://cnts.ua.ac.be/conll2000/chunking/conlleval.txt

[4] https://github.com/wlin12/JNN

machine translation library *dl4mt-material* [5] as a starting point. When running preliminary experiments on POS tagging, we struggled reaching 97% accuracy on the development set. Although only test set scores were reported in Ling et al. [42], we believed these results to be sub-optimal for C2W models, in large part because the comparable Stanford tagger could reach 97.28% on the development set [24]. We modified the models somewhat to try to bridge the gap between the observed results and our expectations.

Dropout

Arguably the most helpful change was the addition of dropout to the model [27, 34], where some units of the neural networks are randomly dropped during training, or scaled appropriately at the test stage. Dropout may be thought of as an implicit ensemble method, and has been recognized as a strong regularizer [34]. Following a suggestion of Zaremba et al. [33], we dropped the non-recurrent inputs to all RNNs, at both hierarchical levels. Moreover, we also applied the technique to the feed-forward layer used just before predicting labels.

GRU-LSTM hybrid

Based on validation set scores, mostly on the POS tagging task, we also modified the recurrent units of our models. We finally opted out for an hybrid between the GRU [32] and the LSTM [9]. More precisely, we reintroduced the input and forget gates of the LSTM instead of the GRU's update gate. In doing so, we also applied the recommendation of Jozefowicz et al. [38, 12] to this hybrid, initializing the forget gate bias to 1 to reduce the impact of the vanishing gradient problem [6]. Equations for this GRU-LSTM hybrid, with σ and $\odot$ denoting the sigmoid function and element-wise product respectively, are given by

$$
\begin{aligned}
i^{(t)} &= \sigma(W_i x^{(t)} + U_i h^{(t-1)} + b_i) \\
f^{(t)} &= \sigma(W_f x^{(t)} + U_f h^{(t-1)} + b_f) \\
r^{(t)} &= \sigma(W_r x^{(t)} + U_r h^{(t-1)} + b_r) \\
\tilde{h}^{(t)} &= tanh(W_h x^{(t)} + U_h(r^{(t)} \odot h^{(t-1)}) + b_h) \\
h^{(t)} &= f^{(t)} h^{(t-1)} + i^{(t)} \tilde{h}^{(t)}
\end{aligned}
$$

[5] https://github.com/kyunghyuncho/dl4mt-material

Feedback

For a minority of our chunking experiments, [6] we also trained models with a slightly different architecture. In addition to combining the output of the word-level RNNs differently, concatenating the hidden states of the forward and backward RNNs instead of applying linear transformations and adding the resulting vectors, the prediction of a given chunk would now depend on the previous one. The top-level MLP now receives two inputs, the embedding of the previous chunk as well as the output of the recurrent neural networks. During training, we use the gold standard, whereas we must rely on the network's predictions at test time, using beam search. Such feedback has previously been applied in neural machine translation [32, 36].

3.3 Ensembling

A simple and common way to improve performance of neural networks is ensembling [8], where the predictions of multiple models are averaged. Even with nearly identical networks, reshuffling the data and changing the random seeds, as in [35], may lead to improvements as the learning objective is not convex. However, more variance amongst (roughly equally good) models would generally still be beneficial [8].

4 Word segmentation

We used characters as the basic input unit for the model explained above. Apart from characters, we can also use subword units as input. A potential advantage of using subword units is that they share the advantages of using a word level model and are as general as character level models. Sennrich et al. [44] proposed a hypothesis that segmentation of rare words into appropriate subwords is sufficient to allow a neural network to learn and generalize the transparent structure of words. We further experimented with this hypothesis by trying different segmentation techniques including the one proposed by Sennrich et al.

4.1 N-grams

For this model, we used n-grams of a word as the input to the model. We experimented with unigram (character level) and bi-gram models.

[6] We also applied the model to POS tagging, but could not get the results in time

4.2 BPE

Byte Pair Encoding [7] is a data compression technique in which the most common pair of bytes is replaced with a byte that does not occur within that data. This process is repeated iteratively. Sennrich et al. [44] adapted this data compression technique to word segmentation. Instead of replacing the common byte, they merged most frequent character pairs to create new symbols. For example:
aaabdaaabac $\rightarrow$ ZabdZabac $\rightarrow$ ZYdZYac $\rightarrow$ XdXac

4.3 Vowel-based

Sometimes, it is not necessary that every word needs to be segmented in subword units. We used the heuristic-based word segmentation method described in [28]. They kept the most frequent W words and segmented the rest into subword using vowels as the split points. Next they kept the most frequent S subwords and segmented the rest into individual characters. At test time, we use the counts from the training data to segment the words. Hyper-parameters for this segmentation technique are W and S. For example:
company interactive magazines $\rightarrow$ company in+te+ra+cti+ve ma+ga+zi+ne+s

4.4 Hyphenation

The hyphenation algorithm is a set of rules that decide at which point a word can be split over two lines with a hyphen. It is widely used in TeX typesetting systems and was first proposed by [2]. We used this algorithm to find all possible split points and use them as subwords. For example:
sophistication $\rightarrow$ so + phis + ti + ca + tion

4.5 PMI-based

Finally, we tried a heuristic approach that we thought could help alleviate perceived weaknesses of BPE. In particular, when merging two consecutive frequent n-grams into a new one, the original components may remain in the text, with low frequency. As such, we disallowed merges that would leave one of its original unit with a count of less than 100, unless they would both become 0.

Moreover, rather than merging frequent consecutive n-grams, we opted to use a variant of pointwise mutual information (PMI) [3] in order to combine characters (or potentially n-grams) that occur together more often than they should according to their individual frequencies. To counter the tendency of PMI to favor low-frequency tokens, the counts of the right tokens were raised to the power of 0.75, as in [40]. We also multiplied the PMI scores of a pair by the logarithm of its count.

After computing all modified PMI scores, we ranked the allowed pairs and then greedily merged consecutive characters for a given number of iterations. We initially tried to recompute PMI before each merge, similarly to BPE, but finally chose not to do so as it was quite time-consuming. In consequence, we only merge characters into bigrams, but the technique could be easily extended, alternating between PMI computations and multiple merges.

5 Results

Based on validation set performance, the dimensionality of all vectors (subword/word embeddings, RNN hidden states, etc.) was set to 200 for POS tagging experiments, with dropout values between 0.5 and 0.7. [7] On the chunking task, we reduced the capacity of the models, with vectors of dimension 100. During training, we used mini-batches of size 48, limiting target sentences to length 60. Models were optimized with Adadelta [29], clipping to gradients to 1 [30], and we computed validation accuracy every 100 mini-batches.

For characters, BPE and PMI-based models, we used all available subword units. For bigrams and other more aggressive segmentation approaches, we used most of the vocabulary, but replaced some of the rarest tokens by an unknown symbol.

5.1 POS tagging

We report results on POS tagging in table 1.

With our modified C2W models, we have obtained a test accuracy of 97.18%, which is somewhat close to the result reported in [42], but still somewhat inferior. We unfortunately cannot figure out the source of the discrepancy, as we believe our changes were beneficial, at least based on validation set performance. We tried using the JNN library [8] released by the authors of [42], but could not reproduce the results, although this may have been due to technical difficulties. For unknown reasons, the code often crashed, although

[7] We initially tried the hyper-parameters mentioned in [42]
[8] https://github.com/wlin12/JNN

Model	Accuracy (%)
Characters	97.18 (97.34)
BPE (100 merges)	97.23 (97.23)
PMI-based (50 merges)	97.16 (97.34)
Bigrams	97.11 (97.19)
Vowel-based	97.14 (97.19)
Hyphenation	96.48 (96.59)
Ensemble (13 models)	97.50 (97.62)
Brill [4]	94.59 (94.38)
Stanford [19]	97.32 (97.28)
Characters [42]	97.36
Characters + features [42]	97.57
LSTM-CRF (features) [37]	97.45

Table 1: POS tagging performance on the Penn Treebank. Development set performance is reported in parentheses, if known.

Wang Ling told us that he did not experience such behaviour [48]. When it did so, it would reload the model that had the lowest validation perplexity (not the latest model, or the one with highest validation accuracy), and resume training from the beginning of the data.

With either BPE or the PMI-based approach, we obtained very similar results as with characters. However, it is important to note that we did few merges, so the segmentations were quite similar to using characters. When we tried to increase the number of merges, performance of the development set worsened. For example, using BPE with 500 merges and 1000 merges, development set accuracy was 97.17% and 97.09% respectively. Moreover, although we developed the PMI-based heuristic to address what we thought were weaknesses of BPE, it did not help. As for bigrams, performance appeared to drop somewhat, although by a fairly small margin.

For the vowel-based method, the hyperparameters W and S changed the performance a lot. With values of W and S set to 1000 and 2000 respectively, the subword segmentation performed well, with a development set accuracy of 97.19 % and a test set accuracy of 97.14%. As we increased W the performance dropped since the model started to become a word level model and lost the advantages of character level models, like modelling unknown words better.

5.2 Chunking

Table 2 presents results on the chunking task, with our models being trained on IOB tags. No segmentation technique amongst those we tried surpassed characters. In fact, performance seems to deteriorate more quickly than for POS tagging when using aggressive segmentation techniques.

Model	F-Score (%)
Characters	92.86 (92.37)
BPE (100 merges)	92.38 (91.94)
PMI-based (100 merges)	92.65 (92.42)
Bigrams	91.76 (91.41)
Vowel-based	90.10 (90.03)
Hyphenation	88.61 (88.03)
Characters + feedback	93.32 (92.86)
MaxEnt (NLTK)	92.20 (92.70)
MaxEnt (from Conll2000) [13]	91.97
Perceptron [20]	93.74
Bi-LSTM-CRF (features) [37]	94.46

Table 2: Chunking performance on the Penn Treebank (all chunks). For fair comparison, all our models were trained with IOB tags. Development set performance is reported in parentheses, if known.

Using feedback from the previously predicted chunk appears useful, as we obtained our highest performance with it. We also trained some models with BEISO tags, converting them back to IOB before evaluation, but could not do so for all segmentation techniques. In general, it appeared to have a positive impact. For example, the character-based model reached 92.63% and 93.01% on the development and test set respectively.

We should note that contrarily to POS tagging, the results we obtained on this task are pretty far from the state of the art. In particular, even models based on LSTMs (and CRFs) may reach upwards of 94% [37]. Given the small size of the training data, we believe adding pretrained POS tags may be useful. We didn't use those we obtained in the other task because the chunking test section overlaps with the POS development set. We also could not get a Brill tagger to exactly replicate the POS tags from the CoNLL 2000 training and test set, and hence could not generate appropriate tags for the validation data. We believe hand-crafted features may also be useful here.

6 Conclusion

In this project, we reimplemented the POS tagging model of Ling et al [42], as well as a few variants. We also applied these models to chunking, a closely related task.

We had set to investigate the impact of using different types of subword units as input to the model and observed that characters are quite a strong baseline, at least when restricted to using no additional features. We could obtain fairly similar results with many segmentation heuristics, but none proved clearly superior.

Acknowledgements

We communicated with Wang Ling by email to get some clarifications about his code (expected training time, crashes we experienced). We also asked him what the development set accuracy was for POS tagging, but he did not recall.

The use of SEIBO tags for chunking was suggested by Slav Petrov.

We used the chunking data, preprocessing and evaluation scripts from the CoNLL-2000 shared task. [9] The Penn Treebank dataset was downloaded by Kyunghyun Cho, with NYU's LDC account.

The code we wrote for this project is built on top of *dl4mt-material* [10]. We also looked at *arctic-captions* [11] [46] and the LSTM *deeplearning.net* tutorial[12] for the LSTM implementation. Moreover, we consulted the *hred-qs* library[13] [45].

For BPE and bigram segmentation, we used Rico Sennrich's *subword-nmt* code. [14]

We relied on NLTK [15] for a variety of preprocessing and training steps. The Brill tagger [16] and Maximum Entropy tagger [17] were based on built-in NLTK taggers.

I, Sébastien Jean, talked about the project with Kyunghyun Cho and Junyoung Chung, telling them what I was doing in the NLP class.

I, Andrew Drozdov, discussed text compression techniques related to subword segmentation with Serban Porumbescu.

We would like to acknowledge the CIMS helpdesk staff for increasing the quota for our CIMS computing accounts to 14 Gb and providing additional 20 Gb for data set. Also, we would like to thank CIMS computing for providing the GPU nodes.

[9] www.cnts.ua.ac.be/conll2000/chunking

[10] https://github.com/kyunghyuncho/dl4mt-material

[11] https://github.com/kelvinxu/arctic-captions

[12] http://deeplearning.net/tutorial/lstm.html#lstm

[13] https://github.com/sordonia/hred-qs

[14] https://github.com/rsennrich/subword-nmt

[15] https://github.com/nltk/nltk

[16] https://github.com/japerk/nltk-trainer

[17] http://www.nltk.org/book/ch07.html

Honor pledge

We pledge our honor that all the work described in this report is solely ours and that we have given credit to all third party resources that we have used.

References

[1] CJ van Rijsbergen. *Information Retrieval. 1979*. Butterworth, 1979.

[2] Franklin Mark Liang. "Word Hy-phen-a-tion by Comput-er (Hyphenation, Computer)". PhD thesis. Stanford, CA, USA, 1983.

[3] Kenneth Ward Church and Patrick Hanks. "Word association norms, mutual information, and lexicography". In: *Computational linguistics* 16.1 (1990), pp. 22–29.

[4] Eric Brill. "A Simple Rule-based Part of Speech Tagger". In: ANLC. 1992, pp. 152–155.

[5] Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. "Building a large annotated corpus of English: the penn treebank". In: *Comput. Linguist.* 19.2 (June 1993), pp. 313–330.

[6] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. "Learning long-term dependencies with gradient descent is difficult". In: *Neural Networks, IEEE Transactions on* 5.2 (1994), pp. 157–166.

[7] Philip Gage. "A New Algorithm for Data Compression". In: *C Users J.* 12.2 (Feb. 1994), pp. 23–38.

[8] A. Krogh and J. Vedelsby. "Neural network ensembles, cross validation and active learning". In: Cambridge MA: MIT Press, 1995, pp. 231–238.

[9] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[10] Mike Schuster and Kuldip K Paliwal. "Bidirectional recurrent neural networks". In: *Signal Processing, IEEE Transactions on* 45.11 (1997), pp. 2673–2681.

[11] Lance A Ramshaw and Mitchell P Marcus. "Text chunking using transformation-based learning". In: *Natural language processing using very large corpora*. Springer, 1999, pp. 157–176.

[12] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. "Learning to forget: Continual prediction with LSTM". In: *Neural computation* 12.10 (2000), pp. 2451–2471.

[13] Rob Koeling. *Chunking with Maximum Entropy Models*. 2000.

[14] Erik F. Tjong Kim Sang and Sabine Buchholz. "Introduction to the CoNLL-2000 Shared Task: Chunking". In: *Proceedings of CoNLL-2000 and LLL-2000*. Ed. by Claire Cardie et al. Lisbon, Portugal, 2000, pp. 127–132. URL: https://aclweb.org/anthology/W/W00/W00-0726.pdf.

[15] Michael Collins. “Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms”. In: *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*. EMNLP ’02. Stroudsburg, PA, USA: Association for Computational Linguistics, 2002, pp. 1–8.

[16] Yoshua Bengio et al. “A Neural Probabilistic Language Model”. In: 3 (2003), pp. 1137–1155.

[17] Fei Sha and Fernando Pereira. “Shallow parsing with conditional random fields”. In: *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*. Association for Computational Linguistics. 2003, pp. 134–141.

[18] Kristina Toutanova et al. “Feature-rich Part-of-speech Tagging with a Cyclic Dependency Network”. In: *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*. NAACL ’03. Association for Computational Linguistics, 2003, pp. 173–180.

[19] Kristina Toutanova et al. “Feature-rich part-of-speech tagging with a cyclic dependency network”. In: *Proceedings of HLT-NAACL*. 2003, pp. 173–180.

[20] Xavier Carreras and Lluis Marquez. “Phrase recognition by filtering and ranking with perceptrons”. In: *Recent advances in natural language processing III: selected papers from RANLP 2003* 260 (2004), p. 205.

[21] Alex Graves and Jürgen Schmidhuber. “Framewise phoneme classification with bidirectional LSTM and other neural network architectures”. In: *Neural Networks* 18.5 (2005), pp. 602–610.

[22] Mathias Creutz and Krista Lagus. “Unsupervised models for morpheme segmentation and morphology learning”. In: *ACM Transactions on Speech and Language Processing (TSLP)* 4.1 (2007), p. 3.

[23] James Bergstra et al. “Theano: a CPU and GPU Math Expression Compiler”. In: *Proceedings of the Python for Scientific Computing Conference (SciPy)*. Austin, TX, June 2010.

[24] Christopher D Manning. “Part-of-speech tagging from 97% to 100%: is it time for some linguistics?” In: *Computational Linguistics and Intelligent Text Processing*. Springer, 2011, pp. 171–189.

[25] Ilya Sutskever, James Martens, and Geoffrey E Hinton. “Generating text with recurrent neural networks”. In: *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. 2011, pp. 1017–1024.

[26] Frédéric Bastien et al. *Theano: new features and speed improvements*. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop. 2012.

[27] Geoffrey E Hinton et al. “Improving neural networks by preventing co-adaptation of feature detectors”. In: *arXiv preprint arXiv:1207.0580* (2012).

[28] Tomáš Mikolov et al. “Subword language modeling with neural networks”. In: *Not published rejected from ICASSP 2012* (2012).

[29] Matthew D Zeiler. “ADADELTA: An adaptive learning rate method”. In: *arXiv preprint arXiv:1212.5701* (2012).

[30] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. “On the difficulty of training recurrent neural networks”. In: *Proceedings of The 30th International Conference on Machine Learning*. 2013, pp. 1310–1318.

[31] Jan A. Botha and Phil Blunsom. “Compositional Morphology for Word Representations and Language Modelling”. In: *Proceedings of the 31st International Conference on Machine Learning (ICML)*. Beijing, China, 2014.

[32] Kyunghyun Cho et al. “Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, 2014, pp. 1724–1734.

[33] Vu Pham et al. “Dropout improves recurrent neural networks for handwriting recognition”. In: *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*. IEEE. 2014, pp. 285–290.

[34] Nitish Srivastava et al. “Dropout: A simple way to prevent neural networks from overfitting”. In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

[35] Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. “Sequence to sequence learning with neural networks”. In: *Advances in neural information processing systems*. 2014, pp. 3104–3112.

[36] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. “Neural machine translation by jointly learning to align and translate”. In: *Proceedings of The 3rd International Conference on Learning Representations*. 2015.

[37] Zhiheng Huang, Wei Xu, and Kai Yu. “Bidirectional LSTM-CRF Models for Sequence Tagging”. In: *CoRR* abs/1508.01991 (2015).

[38] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. “An Empirical Exploration of Recurrent Network Architectures”. In: *Proceedings of The 32nd International Conference on Machine Learning (ICML)*. 2015, pp. 2342–2350.

[39] Yoon Kim et al. “Character-aware neural language models”. In: *AAAI 2016* (2015).

[40] Omer Levy, Yoav Goldberg, and Ido Dagan. “Improving distributional similarity with lessons learned from word embeddings”. In: *Transactions of the Association for Computational Linguistics* 3 (2015), pp. 211–225.

[41] Wang Ling et al. “Character-based Neural Machine Translation”. In: *arXiv preprint arXiv:1511.04586* (2015).

[42] Wang Ling et al. “Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation”. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, 2015, pp. 1520–1530.

[43] Christopher D. Manning. “Computational Linguistics and Deep Learning”. In: *Computational Linguistics* (2015), pp. 699–705.

[44] Rico Sennrich, Barry Haddow, and Alexandra Birch. “Neural Machine Translation of Rare Words with Subword Units”. In: *CoRR* abs/1508.07909 (2015).

[45] Alessandro Sordoni et al. “A hierarchical recurrent encoder-decoder for generative context-aware query suggestion”. In: *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM. 2015, pp. 553–562.

[46] Kelvin Xu et al. “Show, attend and tell: Neural image caption generation with visual attention”. In: *Proceedings of the 32st International Conference on Machine Learning (ICML)*. Lille, France, 2015.

[47] Xiang Zhang, Junbo Zhao, and Yann LeCun. “Character-level Convolutional Networks for Text Classification”. In: *Advances in Neural Information Processing Systems 28*. Ed. by C. Cortes et al. Curran Associates, Inc., 2015, pp. 649–657.

[48] Wang Ling. *Personal communication*.