Neural Semi-Markov Conditional Random Fields for Robust Character-Based Part-of-Speech Tagging

Apostolos Kemos\textsuperscript{2}, Heike Adel\textsuperscript{1,3}, and Hinrich Schütze\textsuperscript{1}

\textsuperscript{1}Center for Information and Language Processing (CIS), LMU Munich, Germany
\textsuperscript{2}Department of Computer Engineering and Informatics, University of Patras, Greece
\textsuperscript{3}Institute for Natural Language Processing (IMS), University of Stuttgart, Germany

Abstract

Character-level models of tokens have been shown to be effective at dealing with within-token noise and out-of-vocabulary words. But these models still rely on correct token boundaries. In this paper, we propose a novel end-to-end character-level model and demonstrate its effectiveness in multilingual settings and when token boundaries are noisy. Our model is a semi-Markov conditional random field with neural networks for character and segment representation. It requires no tokenizer. The model matches state-of-the-art baselines for various languages and significantly outperforms them on a noisy English version of a part-of-speech tagging benchmark dataset.

1 Introduction

Recently, character-based neural networks (NNs) gained popularity for NLP tasks, ranging from text classification (Zhang et al., 2015) and language modeling (Kim et al., 2016) to machine translation (Luong and Manning, 2016). Character-level models are attractive since they can effectively model morphological variants of words and build representations even for unknown words. Thus, they suffer less from out-of-vocabulary problems.

However, most character-level models still rely on tokenization and use characters only for creating more robust token representations (Santos and Zadrozny, 2014; Lample et al., 2016; Ma and Hovy, 2016; Plank et al., 2016). This leads to high performance on well-formatted text or text with misspellings (Yu et al., 2017) but ties the performance to the quality of the tokenizer. While humans are very robust to noise caused by insertion of spaces (e.g., “car nival”) or deletion of spaces (“deeplearning”), this can cause severe underperformance of machine learning models. Similar challenges arise for languages with difficult tokenization such as Chinese or Vietnamese. For text with difficult or noisy tokenization, more robust models are needed.

In contrast to prior work, our model does not require any tokenization. As a result, it does not suffer from the problems mentioned above. It is based on semi-Markov conditional random fields (semi-CRFs) (Sarawagi and Cohen, 2005) which jointly learn to segment and label the input (e.g., characters). To represent the character segments in vector space, we compare different NN approaches.

In our experiments, we address part-of-speech (POS) tagging. However, our model is generally applicable to other sequence-tagging tasks as well since it does not require any task-specific hand-crafted features. Our model achieves state-of-the-art results on the Universal Dependencies dataset (Nivre et al., 2015). To demonstrate its effectiveness, we evaluate it not only on English but also on languages with inherently difficult tokenization, namely Chinese, Japanese and Vietnamese. We further analyze the robustness of our model against difficult tokenization by randomly corrupting the tokenization of the English dataset. Our model significantly outperforms state-of-the-art token-based models in this analysis.

Our contributions are: 1) We present a truly end-to-end character-level sequence tagger that does not rely on any tokenization and achieves state-of-the-art results across languages. 2) We show its robustness against noise caused by corrupted tokenization, further establishing the importance of character-level models as a promising research direction.

2 Model

This section describes our model which is also depicted in Figure 1.
2.1 Character-based Input Representation

The input is the raw character sequence. We convert each character to a one-hot representation. Out-of-vocabulary characters are represented with a zero vector. Our vocabulary does not include the space character since there is no part-of-speech label for it. Instead, our model represents space as two “space features” (lowest level in Figure 1): two binary dimensions indicate whether the previous or next character is a space. Thus, a linear transformation is applied to the extended one-hot encoding to produce a character embedding. The character embeddings are fed into a bidirectional LSTM (biLSTM) (Hochreiter and Schmidhuber, 1997) that computes context-aware representations. These representations form the input to the segment-level feature extractor.

2.2 Semi-Markov CRF

Our model partitions a sequence of characters $x = \{x_1, \ldots, x_T\}$ of length $T$, into (token-like) segments $s = \{s_1, \ldots, s_{|s|}\}$ with $s_j = (a_j, d_j, y_j)$ where $a_j$ is the starting position of the $j^{th}$ segment, $d_j$ is its length and $y_j$ is its label. Thus, it assigns the same label $y_j$ to the whole segment $s_j$. The sum of the lengths of the segments equals the number of non-space characters: $\sum_{j=1}^{|s|} d_j = T$.

The semi-CRF defines the conditional distribution of the input segmentations as:

$$p(s|x) = \frac{1}{Z(x)} \exp \left( \sum_{j=1}^{|s|} F(s_j, x) + A(y_{j-1}, y_j) \right)$$

$$Z(x) = \sum_{s \in S} \exp \left( \sum_{j=1}^{|s|} F(s_j', x) + A(y_{j-1}', y_j') \right)$$

where $F(s_j, x)$ is the score for segment $s_j$ (including its label $y_j$), and $A(y_{j-1}, y_j)$ is the transition score of the labels of two adjacent segments. Thus, $p(s|x)$ jointly models the segmentation and label assignment. For the normalization term $Z(x)$, we sum over the set of all possible segmentations $S$.

The score $F(s_j, x)$ is computed as:

$$F(s_j, x) = w_{y_j}^T f(s_j, x) + b_{y_j},$$

$$W = (w_1, \ldots, w_{|Y|})^T \in \mathbb{R}^{|Y| \times D} \text{ and } b = (b_1, \ldots, b_{|Y|})^T \in \mathbb{R}^{|Y|}$$

are trained parameters, $f(s_j, x) \in \mathbb{R}^D$ is the feature of the labeled segment $s_j$, $|Y|$ is the number of output classes and $D$ is the length of the segment representation.

For training and decoding, we use the semi-Markov analogies of the forward and Viterbi algorithm, respectively (Sarawagi and Cohen, 2005). All computations are performed in log-space.

2.2.1 Segment-level Features

Sarawagi and Cohen (2005) and Yang and Cardie (2012) compute segment-level features by handcrafted rules. Recent work learns the features automatically with NNs (Kong et al., 2015; Zhuo et al., 2016). This avoids the manual design of new features for new languages/tasks. We adopt Gated Recursive Convolutional Neural Networks (grConv) (Cho et al., 2014; Zhuo et al., 2016).

GrConv constructs features by recursively combining adjacent segment representations in a pyramid shape way (see Figure 1). The $d^{th}$ level of the pyramid consists of all representations for segments of length $d$. The first level holds the character representations from our biLSTM. The representation $z_k^{(d)} \in \mathbb{R}^D$, stored in the $k^{th}$ node of layer $d$, is computed as follows:

$$z_k^{(d)} = \theta_L \circ z_k^{(d-1)} + \theta_R \circ z_k^{(d-1)} + \theta_M \circ \hat{z}_k^{(d)}$$

$$\text{with } \hat{z}_k^{(d)} = g(W_L z_k^{(d-1)} + W_R z_k^{(d-1)} + b_w)$$

where $W_L, W_R \in \mathbb{R}^{D \times D}$ and $b_w \in \mathbb{R}^D$ are globally shared parameters, $\theta_L$, $\theta_M$ and $\theta_R$ are gates, $g$ is a non-linearity and $\circ$ denotes element-wise multiplication. The gates are illustrated in the blue box.
of Figure 1 and described in (Zhuo et al., 2016).

3 Experiments and Analysis

Our implementation is in PyTorch (Paszke et al., 2017). See supplementary for hyperparameters.

Data and Evaluation. To compare our model to state-of-the-art character-based POS taggers, we evaluate its accuracy on the English part of the Universal Dependencies (UD) v1.2 dataset (Nivre et al., 2015). For multilingual experiments, we use the English (EN), Chinese (ZH), Japanese (JA) and Vietnamese (VI) part of UD v2.0 (Nivre et al., 2017), using the splits, training and evaluation rules from the CoNNL 2017 shared task (Zeeman et al., 2017). In particular, we calculate joint tokenization and UPOS (universal POS) $F_1$.

3.1 Experiments on English Data (UD v1.2)

Baselines. We compare our model to two character-based models that are state of the art on UD v1.2: bilstm-aux (Plank et al., 2016) and CNN Tagger (Yu et al., 2017). We also compare to a state-of-the-art word-based CRF model MarMot$^3$ (Müller and Schütze, 2015).

Results. Table 1 provides our results on UD v1.2, categorizing the models into token-level ($\vec{w}$) and character-only models ($\vec{c}$). While most pure character-level models cannot ensure consistent labels for each character of a token, our semi-CRF outputs correct segments in most cases (tokenization $F_1$ is 98.69%, see Table 3), and ensures a single label for all characters of a segment. Our model achieves the best results among all character models and comparable results to MarMot.

To evaluate the effectiveness of grConv, we replace it with a Segmental Recurrent Neural Network (SRNN) (Kong et al., 2015). SRNN uses dynamic programming and bi-LSTMs to create segment representations. Its performance is slightly worse compared to grConv (last row of Table 1). While grConv hierarchically combines context-enhanced n-grams, SRNN constructs segments in a sequential order. The latter may be less suited for compositional segments like “airport”.

3.2 Multilingual Experiments (UD v2.0)

Baselines. We compare to the top performing model for EN, JA, VI, ZH from the CoNLL 2017 shared task: UDPipe 1.2 (Straka and Straková, 2017), Stanford (Dozat et al., 2017), FBAML (Qian and Liu, 2017), TRL (Kanayama et al., 2017), and IMS (Björkellund et al., 2017).

Results. Table 2 provides our results. While for each language another shared task system performs best, our system performs consistently well across languages (best or second-best except for EN), leading to the best average scores for both tokenization and POS tagging. Moreover, it matches the state of the art for VI and ZH, two languages with very different characteristics in tokenization.

3.3 Analysis on Noisy Data

Analyzing the robustness of our model on data with corrupted tokenization can give us insight into why it performs well on languages with difficult tokenization (e.g., Chinese, Vietnamese).

Data. We are not aware of a POS tagging dataset with corrupted tokenization. Thus, we create one based on UD v1.2 (EN). For each token, we either delete the space after it with probability $P = p_d$ or insert a space between two characters with $P = p_i$: ”The fox chased the rabbit” $\rightarrow$ ”The f ox cha sed therabbit”. We vary $p_d$ and $p_i$ to construct 3 datasets with different noise levels (LOW, MID, HIGH). See supplementary for statistics and details on labeling the corrupted tokens.

Baseline. We compare our joint model to a traditional pipeline of tokenizer (UDpipe 1.0$^4$) and token-level POS tagger (MarMot).$^5$ We re-train MarMot on the corrupted datasets.

Evaluation. The output of our model is directly evaluated against the gold labels of the original corpus (CLEAN). For MarMot, the evaluation is more tricky since it outputs a label for each of the (possibly wrong) tokens from the tokenizer. To account for this, we choose a relaxed evaluation for the pipeline: We count the POS tag of a gold token as correct if MarMot predicts the tag for any subpart of it. See supplementary for examples.

Results. The performance of our model decreases only slightly when increasing the noise

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|}
\hline
 & $\vec{w}$ & $\vec{c}$ \\
\hline
MarMot & 94.36 & - \\
bilstm-aux & 92.10 & 91.62 \\
CNN Tagger & 89.69 & 93.76 \\
Our & - & 94.27 \\
Our with SRNN & - & 93.86 \\
\hline
\end{tabular}
\caption{POS tag accuracy on UD v1.2 (EN).}
\end{table}

\footnote{UD v1.2 does not provide data for JA, VI, ZH.}

\footnote{http://cistern.cis.lmu.de/marmot/}

\footnote{In contrast to Table 1 with gold tokens for MarMot.}
Table 2: Tokenization and POS $F_1$ on UD v2.0. Best scores are in bold, second-best scores are underlined.

| Noise level | UDpipe+MarMot Tokens | UDpipe+MarMot POS | Our Tokens | Our POS |
|-------------|---------------------|------------------|------------|---------|
| CLEAN       | 98.48               | 93.48            | 98.69      | 94.27   |
| LOW         | 70.90               | 83.73            | 96.08      | 92.80   |
| MID         | 20.62               | 58.53            | 95.28      | 92.54   |
| HIGH        | 20.47               | 56.96            | 95.45      | 92.14   |

Table 3: Tokenization $F_1$ and POS tag accuracies on noisy version of UD v1.2.

Discussion. The results in Table 3 show that our model can reliably recover token boundaries, even in noisy scenarios. This also explains its strong performance across languages: It can handle different languages, independent of whether the language merges tokens without whitespaces (e.g., Chinese) or separates tokens with whitespaces into syllables (e.g., Vietnamese).

4 Related Work

Character-based POS Tagging. Most work uses characters only to build more robust token representations but still relies on external tokenizers (Santos and Zadrozny, 2014; Lample et al., 2016; Plank et al., 2016; Dozat et al., 2017; Liu et al., 2017). In contrast, our model jointly learns segmentation and POS tagging. Gillick et al. (2016) do not rely on tokenization either but in contrast to their greedy decoder, our model optimizes the whole output sequence and is able to revise local decisions (Lafferty et al., 2001). For processing characters, LSTMs (Lample et al., 2016; Dozat et al., 2017) or CNNs (Ma and Hovy, 2016) are used. Our model combines bi-LSTMs and grConv to model both the context of characters (LSTM) and the compositionality of language (grConv).

Joint Segmentation and POS Tagging. The top performing models of EN, JA, VI and ZH use a pipeline of tokenizer and word-based POS tagger but do not treat both tasks jointly (Björkelund et al., 2017; Dozat et al., 2017; Kanayama et al., 2017; Qian and Liu, 2017). Our analysis shows the disadvantage of this. Chen et al. (2017) and Shao et al. (2017), inter alia, jointly word-segment and sequence-tag Chinese with a bi-LSTM-CRF model that predicts one POS tag per Chinese character. This approach is hard to transfer to languages like English and Vietnamese where single characters are less informative and tokens are much longer, resulting in a larger combinatorial label space for the CRF. Thus, we choose a semi-Markov formalization to directly model segments.

Semi-Markov CRFs for Sequence Tagging. Zhuo et al. (2016) and Ye and Ling (2018) apply semi-CRFs to word-level inputs for named entity recognition. In contrast, we use semi-CRFs to model character-based POS tagging. Our model is different in several respects. (i) Its input contains no information about tokens. (ii) The expected length of character segments is considerably larger than the expected length of word-based segments for NER. (iii) We apply an LSTM to automatically learn relevant context information over segment boundaries. Kong et al. (2015) build SRNNs that we use as a baseline. In contrast to their 0-order model, we train a 1-order semi-CRF to model dependencies between segment labels.

5 Conclusion

We presented an end-to-end model for character-based part-of-speech tagging that uses semi-Markov conditional random fields to jointly segment and label a sequence of characters. Input representations and segment representations are trained parameters learned in end-to-end training by the neural network part of the model. The model achieves state-of-the-art results on two benchmark datasets across several typologically diverse languages. By corrupting the tokenization of the dataset, we show the robustness of our model, explaining its good performance on languages with difficult tokenization.
References

Anders Björkelund, Agnieszka Falenska, Xiang Yu, and Jonas Kuhn. 2017. IMS at the CoNLL 2017 UD shared task: CRFs and perceptrons meet neural networks. In Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, pages 40–51, Vancouver, Canada. Association for Computational Linguistics.

Xinchi Chen, Xipeng Qiu, and Xuanjing Huang. 2017. A feature-enriched neural model for joint chinese word segmentation and part-of-speech tagging. In Proceedings of the 26th International Joint Conference on Artificial Intelligence, pages 3960–3966, Melbourne, Australia. AAAI Press.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, pages 103–111, Doha, Qatar. Association for Computational Linguistics.

Timothy Dozat, Peng Qi, and Christopher D Manning. 2017. Stanford’s graph-based neural dependency parser at the CoNLL 2017 shared task. In Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, pages 20–30, Vancouver, Canada. Association for Computational Linguistics.

Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. 2016. Multilingual language processing from bytes. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 1296–1306, San Diego, California. Association for Computational Linguistics.

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

Hiroshi Kanayama, Masayasu Muraoka, and Katsumasa Yoshikawa. 2017. A semi-universal pipelined approach to the conll 2017 ud shared task. In Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, pages 265–273, Vancouver, Canada. Association for Computational Linguistics.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In AAAI Conference on Artificial Intelligence, pages 2741–2749, AAAI Press.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In International Conference on Learning Representations.

Lingpeng Kong, Chris Dyer, and Noah A Smith. 2015. Segmental recurrent neural networks. arXiv preprint arXiv:1511.06018.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In International Conference on Machine Learning, pages 282–289. Morgan Kaufmann Publishers Inc.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 260–270, San Diego, California. Association for Computational Linguistics.

Liyuan Liu, Jingbo Shang, Frank Xu, Xiang Ren, Huan Gui, Jian Peng, and Jiawei Han. 2017. Empower sequence labeling with task-aware neural language model. arXiv preprint arXiv:1709.04109.

Minh-Thang Luong and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1054–1063, Berlin, Germany. Association for Computational Linguistics.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Thomas Müller and Hinrich Schütze. 2015. Robust morphological tagging with word representations. In Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 526–536, Denver, Colorado. Association for Computational Linguistics.

Joakim Nivre et al. 2015. Universal dependencies 1.2. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (`UFAL), Faculty of Mathematics and Physics, Charles University.

Joakim Nivre, Lars Ahrenberg Zeljko Agic, et al. 2017. Universal dependencies 2.0 CoNLL 2017 shared task development and test data. lindat/clarin digital library at the institute of formal and applied linguistics, charles university.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In The future of gradient-based machine learning software and techniques, NIPS 2017.
Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 412–418, Berlin, Germany. Association for Computational Linguistics.

Xian Qian and Yang Liu. 2017. A non-DNN feature engineering approach to dependency parsing – FBAML at CoNLL 2017 shared task. In Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, pages 143–151, Vancouver, Canada. Association for Computational Linguistics.

Cícero N. dos Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In International Conference on Machine Learning, pages 1818–1826.

Sunita Sarawagi and William W Cohen. 2005. Semimarkov conditional random fields for information extraction. In Advances in Neural Information Processing Systems, pages 1185–1192.

Yan Shao, Christian Hardmeier, Jörg Tiedemann, and Joakim Nivre. 2017. Character-based joint segmentation and pos tagging for chinese using bidirectional rnn-crf. In Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 173–183, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Milan Straka and Jana Straková. 2017. Tokenizing, pos tagging, lemmatizing and parsing UD 2.0 with UDPipe. In Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, pages 88–99, Vancouver, Canada. Association for Computational Linguistics.

Bishan Yang and Claire Cardie. 2012. Extracting opinion expressions with semi-markov conditional random fields. In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 1335–1345, Jeju Island, Korea. Association for Computational Linguistics.

Zhi-Xiu Ye and Zhen-Hua Ling. 2018. Hybrid semi-markov crf for neural sequence labeling. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Melbourne, Australia. Association for Computational Linguistics.

Xiang Yu, Agnieszka Falenska, and Ngoc Thang Vu. 2017. A general-purpose tagger with convolutional neural networks. In Proceedings of the First Workshop on Subword and Character Level Models in NLP, pages 124–129, Copenhagen, Denmark. Association for Computational Linguistics.

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajic, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, et al. 2017. Conll 2017 shared task: multilingual parsing from raw text to universal dependencies. Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, pages 1–19.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In Advances in Neural Information Processing Systems, pages 649–657.

Jingwei Zhuo, Yong Cao, Jun Zhu, Bo Zhang, and Zaiqing Nie. 2016. Segment-level sequence modeling using gated recursive semi-markov conditional random fields. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1413–1423, Berlin, Germany. Association for Computational Linguistics.

A Hyperparameters

We train our model end-to-end using backpropagation with Adam as the optimizer (Kingma and Ba, 2014) and a mini-batch size of 20. The parameters of Adam are set to the default values of PyTorch (Paszke et al., 2017): The learning rate is 1e-3. The coefficient for computing running averages of gradient and its square are 0.9 and 0.999, respectively. A term of 1e-8 is added to the denominator for numerical stability.

All remaining hyperparameters are tuned on the development set with the following results. For the input feature layer, we use a character embedding size of 60 and three stacked Bi-LSTM layers with hidden size of 100 for each direction. For the semi-CRF we set the maximum segment length $L = 23$ as tokens of bigger length are rarely seen in the training sets. To avoid overfitting, we apply dropout with probability of 0.25 on each layer including the input (i.e replacing the one hot representations of random characters with a vector of all zeros) similar to the DROP symbol in (Gillick et al., 2016). The last one increases generalization by forcing the model to account better for long-range context and not overfit on local character patterns. We do not apply dropout on the input layer for the noisy experiments, since the corrupt tokenization already acts as a regularizer. In addition, we decrease the learning rate by a factor of 10 if there is no improvement for 10 consecutive epochs on the validation set and employ early stopping after 20 epochs without an accuracy increase on the same set.
B Noisy Dataset

B.1 Label Assignment

For each token, we either delete the space after it with probability $p_d$ or insert a space between two of its characters with probability $p_i$. We assign the label from the original token to every sub-token created by space insertion. For space deletions, we randomly choose one of the two original labels for training and evaluate against the union of them. Figure 2 shows an example.

![Figure 2: Example of label assignment.]

| level | $p_d$ | # deletions | $p_i$ | # insertions |
|-------|------|-------------|------|--------------|
| LOW   | 0.1  | 15198       | 0.05 | 26497        |
| MID   | 0.3  | 39361       | 0.11 | 40474        |
| HIGH  | 0.6  | 65387       | 0.33 | 68209        |

Table 4: Noisy dataset statistics (three different noise levels).

B.3 Pipeline Evaluation: Examples

While we can directly evaluate the output of our model against the gold labels of the original corpus (CLEAN), we use a relaxed evaluation strategy for the pipeline model.

Assume we have the following sentence with gold labels in parantheses in our dataset: “The (DET) fox (NOUN) chased (VERB) the (DET) rabbit (NOUN)”. After the output of the tokenizer on the corrupted dataset, three special cases need to be considered:

1. A token got split into two tokens: “cha” “sed”

   A token-based method will, thus, result in two predictions, one for “cha” and one for “sed”. We evaluate the prediction as correct if one of the two predictions is correct, i.e. “cha (NOUN) sed (VERB)” would be evaluated as correct but “cha (NOUN) sed (NOUN)” would be wrong.

2. Two gold tokens got merged: “chasedthe”

   A token-based method will, thus, result in only one prediction. We check the predicted label against the golden labels, and count one correct for each match, while assuming any mismatch as incorrect. For example “chasedthe (DET)” or “chasedthe (VERB)” would be counted as one correct prediction and one wrong but “chasedthe (NOUN)” would be two incorrect predictions. In the case of tokens that got merged and had the same label, the correct prediction of that label would be counted as two correct predictions.

3. Two tokens got merged but split at another position: “cha” “sedthe”

   In this case, we use a relaxed combination of the previous approaches: We count the prediction as correct if any of its partial predictions match any of the gold labels of the first, splitted token, while evaluating the second, as merged case as described previously. For example: “cha (DET) sedthe (VERB)”, would be counted as one correct (from the split case) and one incorrect (from the merged case).