Elephant: Sequence Labeling for Word and Sentence Segmentation

Kilian Evang[*], Valerio Basile[*], Grzegorz Chrupała[†] and Johan Bos[*]
[*]University of Groningen, Oude Kijk in 't Jatstraat 26, 9712 EK Groningen, The Netherlands
[†]Tilburg University, PO Box 90153, 5000 LE Tilburg, The Netherlands
[*]{k.evang, v.basile, johan.bos}@rug.nl [†]g.chrupala@uvt.nl

Abstract

Tokenization is widely regarded as a solved problem due to the high accuracy that rule-based tokenizers achieve. But rule-based tokenizers are hard to maintain and their rules language specific. We show that high-accuracy word and sentence segmentation can be achieved by using supervised sequence labeling on the character level combined with unsupervised feature learning. We evaluated our method on three languages and obtained error rates of 0.27 ‰ (English), 0.35 ‰ (Dutch) and 0.76 ‰ (Italian) for our best models.

1 An Elephant in the Room

Tokenization, the task of segmenting a text into words and sentences, is often regarded as a solved problem in natural language processing (Dridan and Oepen, 2012), probably because many corpora are already in tokenized format. But like an elephant in the living room, it is a problem that is impossible to overlook whenever new raw datasets need to be processed or when tokenization conventions are reconsidered. It is moreover an important problem, because any errors occurring early in the NLP pipeline affect further analysis negatively. And even though current tokenizers reach high performance, there are three issues that we feel haven't been addressed satisfactorily so far:

- Most tokenizers are rule-based and therefore hard to maintain and hard to adapt to new domains and new languages (Silla Jr. and Kaestner, 2004);
- Word and sentence segmentation are often seen as separate tasks, but they obviously inform each other and it could be advantageous to view them as a combined task;
- Most tokenization methods provide no alignment between raw and tokenized text, which makes mapping the tokenized version back onto the actual source hard or impossible.

In short, we believe that regarding tokenization, there is still room for improvement, in particular on the methodological side of the task. We are particularly interested in the following questions: Can we use supervised learning to avoid hand-crafting rules? Can we use unsupervised feature learning to reduce feature engineering effort and boost performance? Can we use the same method across languages? Can we combine word and sentence boundary detection into one task?

2 Related Work

Usually the text segmentation task is split into word tokenization and sentence boundary detection. Rule-based systems for finding word and sentence boundaries often are variations on matching hand-coded regular expressions (Grefenstette, 1999; Silla Jr. and Kaestner, 2004; Jurafsky and Martin, 2008; Dridan and Oepen, 2012).

Several unsupervised systems have been proposed for sentence boundary detection. Kiss and Strunk (2006) present a language-independent, unsupervised approach and note that abbreviations form a major source of ambiguity in sentence boundary detection and use collocation detection to build a high-accuracy abbreviation detector. The resulting system reaches high accuracy, rivalling handcrafted rule-based and supervised systems. A similar system was proposed earlier by Mikheev (2002).

Existing supervised learning approaches for sentence boundary detection use as features tokens preceding and following potential sentence boundary, part of speech, capitalization information and lists of abbreviations. Learning methods employed in

these approaches include maximum entropy models (Reynar and Ratnaparkhi, 1997) decision trees (Riley, 1989), and neural networks (Palmer and Hearst, 1997).

Closest to our work are approaches that present token and sentence splitters using conditional random fields (Tomanek et al., 2007; Fares et al., 2013). However, these previous approaches consider tokens (i.e. character sequences) as basic units for labeling, whereas we consider single characters. As a consequence, labeling is more resource-intensive, but it also gives us more expressive power. In fact, our approach kills two birds with one stone, as it allows us to integrate token and sentence boundaries detection into one task.

3 Method

3.1 IOB Tokenization

IOB tagging is widely used in tasks identifying chunks of tokens. We use it to identify chunks of characters. Characters outside of tokens are labeled O, inside of tokens I. For characters at the beginning of tokens, we use S at sentence boundaries, otherwise T (for token). This scheme offers some nice features, like allowing for discontinuous tokens (e.g. hyphenated words at line breaks) and starting a new token in the middle of a typographic word if the tokenization scheme requires it, as e.g. in *did|n't*. An example is given in Figure 1.

```
It didn't matter if the faces were male,
SIOTIITIIOTIIIIIOTIOTIIOTIIIIOTIIIOTIIITO
female or those of children. Eighty-
TIIIIIOTIOTIIIIOTIOTIIIIIIITOSIIIIIIO
three percent of people in the 30-to-34
IIIIIOTIIIIIIOTIOTIIIIIOTIOTIIOTIIIIIIIO
year old age range gave correct responses.
TIIIOTIIOTIIOTIIIIOTIIIOTIIIIIIOTIIIIIIIIT
```

Figure 1: Example of IOB-labeled characters

3.2 Datasets

In our experiments we use three datasets to compare our method for different languages and for different domains: manually checked English newswire texts taken from the Groningen Meaning Bank, GMB (Basile et al., 2012), Dutch newswire texts, comprising two days from January 2000 extracted from the Twente News Corpus, TwNC (Ordelman et al., 2007), and a random sample of Italian texts from the PAISÀ corpus (Borghetti et al., 2011).

Table 1: Datasets characteristics.

Name	Language	Domain	Sentences	Tokens
GMB	English	Newswire	2,886	64,443
TNC	Dutch	Newswire	49,537	860,637
PAI	Italian	Web/various	42,674	869,095

The data was converted into IOB format by inferring an alignment between the raw text and the segmented text.

3.3 Sequence labeling

We apply the Wapiti implementation (Lavergne et al., 2010) of Conditional Random Fields (Lafferty et al., 2001), using as features the output label of each character, combined with 1) the character itself, 2) the output label on the previous character, 3) characters and/or their Unicode categories from context windows of varying sizes. For example, with a context size of 3, in Figure 1, features for the E in *Eighty-three* with the output label S would be E/S, O/S, ␣/S, i/S, Space/S, Lowercase/S. The intuition is that the 31 existing Unicode categories can generalize across similar characters whereas character features can identify specific contexts such as abbreviations or contractions (e.g. *didn't*). The context window sizes we use are 0, 1, 3, 5, 7, 9, 11 and 13, centered around the focus character.

3.4 Deep learning of features

Automatically learned word embeddings have been successfully used in NLP to reduce reliance on manual feature engineering and boost performance. We adapt this approach to the character level, and thus, in addition to hand-crafted features we use text representations induced in an unsupervised fashion from character strings. A complete discussion of our approach to learning text embeddings can be found in (Chrupała, 2013). Here we provide a brief overview.

Our representations correspond to the activation of the hidden layer in a simple recurrent neural (SRN) network (Elman, 1990; Elman, 1991), implemented in a customized version of Mikolov (2010)'s RNNLM toolkit. The network is sequentially presented with a large amount of raw text and learns to

predict the next character in the sequence. It uses the units in the hidden layer to store a generalized representation of the recent history. After training the network on large amounts on unlabeled text, we run it on the training and test data, and record the activation of the hidden layer at each position in the string as it tries to predict the next character. The vector of activations of the hidden layer provides additional features used to train and run the CRF. For each of the $K = 10$ most active units out of total $J = 400$ hidden units, we create features $(f(1) \dots f(K))$ defined as $f(k) = 1$ if $s_{j(k)} > 0.5$ and $f(k) = 0$ otherwise, where $s_j(k)$ returns the activation of the k^{th} most active unit. For training the SRN only raw text is necessary. We trained on the entire GMB 2.0.0 (2.5M characters), the portion of TwNC corresponding to January 2000 (43M characters) and a sample of the PAISÀ corpus (39M characters).

4 Results and Evaluation

In order to evaluate the quality of the tokenization produced by our models we conducted several experiments with different combinations of features and context sizes. For these tests, the models are trained on an 80% portion of the data sets and tested on a 10% development set. Final results are obtained on a 10% test set. We report both absolute number of errors and error rates per thousand (‰).

4.1 Feature sets

We experiment with two kinds of features at the character level, namely Unicode categories (31 different ones), Unicode character codes, and a combination of them. Unicode categories are less sparse than the character codes (there are 88, 134, and 502 unique characters for English, Dutch and Italian, respectively), so the combination provide some generalization over just character codes.

Table 2: Error rates obtained with different feature sets. Cat stands for Unicode category, Code for Unicode character code, and Cat-Code for a union of these features.

Feature set	Error rates per thousand (‰) English	Dutch	Italian
Cat-9	45 (1.40)	1,403 (2.87)	1,548 (2.67)
Code-9	**6 (0.19)**	782 (1.60)	692 (1.20)
Cat-Code-9	8 (0.25)	**774 (1.58)**	**657 (1.14)**

From these results we see that categories alone perform worse than only codes. For English there is no gain from the combination over using only character codes. For Dutch and Italian there is an improvement, although it is only significant for Italian ($p = 0.480$ and $p = 0.005$ respectively, binomial exact test). We use this feature combination in the experiments that follow. Note that these models are trained using a symmetrical context of 9 characters (four left and four right of the current character). In the next section we show performance of models with different window sizes.

4.2 Context window

We run an experiment to evaluate how the size of the context in the training phase impacts the classification. In Table 4.2 we show the results for symmetrical windows ranging in size from 1 to 13.

Table 3: Using different context window sizes.

Feature set	Error rates per thousand (‰) English	Dutch	Italian
Cat-Code-1	273 (8.51)	4,924 (10.06)	9,108 (15.86)
Cat-Code-3	118 (3.68)	3,525 (7.20)	2,013 (3.51)
Cat-Code-5	20 (0.62)	930 (1.90)	788 (1.37)
Cat-Code-7	10 (0.31)	778 (1.60)	667 (1.16)
Cat-Code-9	**8 (0.25)**	774 (1.58)	**657 (1.14)**
Cat-Code-11	9 (0.28)	761 (1.56)	692 (1.21)
Cat-Code-13	**8 (0.25)**	**751 (1.54)**	670 (1.17)

4.3 SRN features

We also tested the automatically learned features derived from the activation of the hidden layer of an SRN language model, as explained in Section 3. We combined these features with character code and Unicode category features in windows of different sizes. The results of this test are shown in Table 4. The first row shows the performance of SRN features on their own. The following rows show the combination of SRN features with the basic feature sets of varying window size. It can be seen that augmenting the feature sets with SRN features results in large reductions of error rates. The Cat-Code-1-SRN setting has error rates comparable to Cat-Code-9.

The addition of SRN features to the two best previous models, Cat-Code-9 and Cat-Code-13, reduces the error rate by 83% resp. 81% for Dutch,

and by 24% resp. 26% for Italian. All these differences are statistically significant according to the binomial test ($p < 0.001$). For English, there are too few errors to detect a statistically significant effect for Cat-Code-9 ($p = 0.07$), but for Cat-Code-13 we find $p = 0.016$.

Table 4: Results obtained using different context window sizes and addition of SRN features.

Feature set	Error rates per thousand (‰) English	Dutch	Italian
SRN	24 (0.75)	276 (0.56)	738 (1.28)
Cat-Code-1-SRN	7 (0.21)	212 (0.43)	549 (0.96)
Cat-Code-3-SRN	4 (0.13)	165 (0.34)	507 (0.88)
Cat-Code-5-SRN	3 (0.10)	136 (0.28)	476 (0.83)
Cat-Code-7-SRN	**1 (0.03)**	**111 (0.23)**	497 (0.86)
Cat-Code-9-SRN	2 (0.06)	135 (0.28)	497 (0.86)
Cat-Code-11-SRN	2 (0.06)	132 (0.27)	**468 (0.81)**
Cat-Code-13-SRN	**1 (0.03)**	142 (0.29)	496 (0.86)

In a final step, we selected the best models based on the development sets (Cat-Code-7-SRN for English and Dutch, Cat-Code-11-SRN for Italian), and checked their performance on the final test set. This resulted in 10 errors (0.27 ‰) for English (GMB corpus), 199 errors (0.35 ‰) for Dutch (TwNC corpus), and 454 errors (0.76 ‰) for Italian (PAISÀ corpus).

5 Discussion

It is interesting to examine what kind of errors the SRN features help avoid. In the English and Dutch datasets many errors are caused by failure to recognize personal titles and initials or misparsing of numbers. In the Italian data, a large fraction of errors is due to verbs with clitics, which are written as a single word, but treated as separate tokens. Table 5 shows examples of errors made by a simpler model that are fixed by adding SRN features. Table 6 shows the confusion matrices for the Cat-Code-7 and Cat-Code-7-SRN sets on the Dutch data. The mistake most improved by SRN features is T/I with 89% error reduction (see also Table 5). The is also the most common remaining mistake.

A comparison with other approaches is hard because of the difference in datasets and task definition (combined word/sentence segmentation). Here we just compare our results for sentence segmentation (sentence F_1 score) with Punkt, a state-of-the-art sentence boundary detection system (Kiss and Strunk, 2006). With its standard distributed models, Punkt achieves 98.51% on our English test set, 98.87% on Dutch and 98.34% on Italian, compared with 100%, 99.54% and 99.51% for our system. Our system benefits here from its ability to adapt to a new domain with relatively little (but annotated) training data.

Table 5: Positive impact of SRN features.

	`Ms. Hughes will joi`
Cat-Code-7	`SIIO`**`S`**`IIIIIOTIIIOTII`
Cat-Code-7-SRN	`SIIO`**`T`**`IIIIIOTIIIOTII`
	`$ 3.9 trillion by t`
Cat-Code-7	`TOT`**`T`**`IOTIIIIIIIOTIOT`
Cat-Code-7-SRN	`TOT`**`I`**`IOTIIIIIIIOTIOT`
	`bleek 0,4 procent`
Cat-Code-11	`OTIIIIOT`**`T`**`IOTIIIIIIO`
Cat-Code-11-SRN	`OTIIIIOT`**`I`**`IOTIIIIIIO`
	`toebedeeld: 6,2. In`
Cat-Code-11	`TIIIIIIIIITOT`**`T`**`ITOSI`
Cat-Code-11-SRN	`TIIIIIIIIITOT`**`I`**`ITOSI`
	`prof. Teulings het`
Cat-Code-11	`TIII`**`T`**`O`**`S`**`IIIIIIIOTIIO`
Cat-Code-11-SRN	`TIII`**`I`**`O`**`T`**`IIIIIIIOTIIO`
	`per costringerlo al`
Cat-Code-11	`TIIOTIIIIIIIII`**`I`**`IOTI`
Cat-Code-11-SRN	`TIIOTIIIIIIIII`**`T`**`IOTI`

Table 6: Confusion matrix for Dutch development set.

Gold	Predicted, Cat-Code-7 I	O	S	T	Predicted, Cat-Code-7-SRN I	O	S	T
I	328128	0	2	469	328546	0	0	53
O	0	75234	0	0	0	75234	0	0
S	4	0	4323	18	1	0	4332	12
T	252	0	33	80828	35	0	10	81068

6 What Elephant?

Word and sentence segmentation can be recast as a combined tagging task. This way, tokenization is cast as a supervised learning task, causing a shift of labor from writing rules to manually correcting labels. Learning this task with CRF achieves high accuracy.[1] Furthermore, our tagging method does not lose the connection between original text and tokens.

In future work, we plan to broaden the scope of this work to other steps in document preparation,

[1] All software needed to replicate our experiments is available at `http://gmb.let.rug.nl/elephant/experiments.php`

such as normalization of punctuation, and their interaction with segmentation. We further plan to test our method on a wider range of datasets, allowing a more direct comparison with other approaches. Finally, we plan to explore the possibility of a statistical universal segmentation model for mutliple languages and domains.

In a famous scene with a live elephant on stage, the comedian Jimmy Durante was asked about it by a policeman and surprisedly answered: "What elephant?" We feel we can say the same now as far as tokenization is concerned.

References

Valerio Basile, Johan Bos, Kilian Evang, and Noortje Venhuizen. 2012. Developing a large semantically annotated corpus. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC 2012)*, pages 3196–3200, Istanbul, Turkey.

Claudia Borghetti, Sara Castagnoli, and Marco Brunello. 2011. I testi del web: una proposta di classificazione sulla base del corpus PAISÀ. In M. Cerruti, E. Corino, and C. Onesti, editors, *Formale e informale. La variazione di registro nella comunicazione elettronica*, pages 147–170. Carocci, Roma.

Grzegorz Chrupała. 2013. Text segmentation with character-level text embeddings. In *ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, Atlanta, USA.

Rebecca Dridan and Stephan Oepen. 2012. Tokenization: Returning to a long solved problem – a survey, contrastive experiment, recommendations, and toolkit –. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 378–382, Jeju Island, Korea. Association for Computational Linguistics.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

Jeffrey L. Elman. 1991. Distributed representations, simple recurrent networks, and grammatical structure. *Machine learning*, 7(2):195–225.

Murhaf Fares, Stephan Oepen, and Zhang Yi. 2013. Machine learning for high-quality tokenization - replicating variable tokenization schemes. In A. Gelbukh, editor, *CICLING 2013*, volume 7816 of *Lecture Notes in Computer Science*, pages 231–244, Berlin Heidelberg. Springer-Verlag.

Gregory Grefenstette. 1999. Tokenization. In Hans van Halteren, editor, *Syntactic Wordclass Tagging*, pages 117–133. Kluwer Academic Publishers, Dordrecht.

Daniel Jurafsky and James H. Martin. 2008. *Speech and Language Processing. An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall, 2nd edition.

Tibor Kiss and Jan Strunk. 2006. Unsupervised multilingual sentence boundary detection. *Computational Linguistics*, 32(4):485–525.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML-01*, pages 282–289.

Thomas Lavergne, Olivier Cappé, and François Yvon. 2010. Practical very large scale CRFs. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 504–513, Uppsala, Sweden, July. Association for Computational Linguistics.

Andrei Mikheev. 2002. Periods, capitalized words, etc. *Computational Linguistics*, 28(3):289–318.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Interspeech*.

Roeland Ordelman, Franciska de Jong, Arjan van Hessen, and Hendri Hondorp. 2007. TwNC: a multifaceted Dutch news corpus. *ELRA Newsleter*, 12(3/4):4–7.

David D. Palmer and Marti A. Hearst. 1997. Adaptive multilingual sentence boundary disambiguation. *Computational Linguistics*, 23(2):241–267.

Jeffrey C. Reynar and Adwait Ratnaparkhi. 1997. A maximum entropy approach to identifying sentence boundaries. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 16–19, Washington, DC, USA. Association for Computational Linguistics.

Michael D. Riley. 1989. Some applications of tree-based modelling to speech and language. In *Proceedings of the workshop on Speech and Natural Language*, HLT '89, pages 339–352, Stroudsburg, PA, USA. Association for Computational Linguistics.

Carlos N. Silla Jr. and Celso A. A. Kaestner. 2004. An analysis of sentence boundary detection systems for English and Portuguese documents. In *Fifth International Conference on Intelligent Text Processing and Computational Linguistics*, volume 2945 of *Lecture Notes in Computer Science*, pages 135–141. Springer.

Katrin Tomanek, Joachim Wermter, and Udo Hahn. 2007. Sentence and token splitting based on conditional random fields. In *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics*, pages 49–57, Melbourne, Australia.