Trainable, Multiword-aware Linguistic Tokenization Using Modern Neural Networks

Clara Madeline Elise Boesenberg
Institute of Linguistics
Heinrich Heine University Düsseldorf
clara.boesenberg@hhu.de

Kilian Evang
Institute of Linguistics
Heinrich Heine University Düsseldorf
kilian.evang@hhu.de

Abstract

We revisit MWE-aware linguistic tokenization as a character-level and token-level sequence labeling problem and present a systematic evaluation on English, German, Italian, and Dutch data. We compare a standard linguistic tokenizer trained without MWE-awareness as a baseline (UDPipe), a character-level SRN+CRF model (Elephant), and transformer-based MaChAmp models trained either directly on gold character labels or as token-level postprocessors on top of UDPipe. Our results show that the two-stage pipeline – UDPipe pretokenization followed by MaChAmp postprocessing – consistently yields the best accuracy. Our analysis of error patterns highlights how different architectures trade off over- and under-segmentation. These findings provide practical guidance for building MWE-aware tokenizers and suggest that postprocessing pipelines with transformers are a strong and general strategy for non-standard tokenization.

1 Introduction

Tokenization is a fundamental task in natural language processing that forms the first step of many pipelines. It segments a text (a sequence of *characters*) into a sequence of slightly larger units called *tokens*. How tokens are defined exactly depends on the application. Recent deep neural models use *subwords* that are induced statistically, trading off vocabulary size and sequence lengths. By contrast, in traditional natural language processing (NLP), tokens closely correspond to the linguistic notion of *word*. How a word is defined can also differ: for writing systems with whitespace, word boundaries are most commonly taken to coincide with whitespace and punctuation. But some applications, such as compositional semantic interpretation, employ more complex definitions of tokens, aiming to make them semantically coherent units. For example, multiword expressions like *in spite of* may be treated as a single token because their meaning is atomic and not derivable from the component parts. Conversely, one may want to split some orthographic words like the Italian *vederlo* ("see him"), which consists of a verb and a clitic pronoun.

```
I wasn't in New York.
SOTIITIIOTIOTIIIIIIIT
```

(a) Character-level STIO labeling. S = beginning of sentence, T = beginning of token, I = inside token, O = outside token.

```
I wasn't in New York.
BOBIIBIIOBIOBIIIIIIIB
```

(b) Character-level BIO labeling. Like character-level STIO labeling, but there is no S/T distinction.

```
I was n't in New York .
O O   O   O  B   I    O
```

(c) Pretoken-level BIO labeling. The text is pretokenized by a rule-based system. Contiguous MWEs are recognized and tagged with B, I tags; other pretokens receive O tags.

Figure 1: Three different sequence-labeling schemas for MWE-aware tokenization. In all three cases, the tokens are: *I*, *was*, *n't*, *in*, *New York*, .

Since datasets differ in what they treat as a token, tokenizers for different applications must be adapted; and ideally they should be automatically trainable from data. While prior work motivates adapting tokenization to the downstream objective (Hiraoka et al., 2021), methods for MWE-aware approaches are so far underexplored. In particular, the potential of modern neural networks has not yet been sufficiently explored for this task. There is also the question of what works better for languages with whitespace writing systems: sequence tagging at the character level, or rule-based tokenization followed by MWE recognition, as shown in Figure 1.

Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics
Volume 4: Student Research Workshop, pages 266–276

4	New York
4	Napoleon Bonaparte
3	Vladislav Listyev
3	um die (“circa”)
35	kind of
35	at all
34	in love
23	fed up
16	by no means
4	Charles de Gaulle
3	Jan De Bont
3	hot dog
2	Vladislav Listyev
2	Thomas Edison
21	aan het (progressive marker)
5	meer dan (“more than”)
4	half negen (“half past eight”)
3	minder dan (“less than”)

Table 1: The five most frequent German, English, Italian, and Dutch contiguous MWEs in the PMB 4.0.0 gold data

1.1 Multiword Expressions

Multiword expressions (MWEs) pose long-standing challenges for NLP because their syntactic variability and limited compositionality make them difficult to segment and interpret using standard tokenization strategies (Sag et al., 2002; Baldwin and Kim, 2010).

MWEs range from idioms (*kick the bucket*) to less fixed but lexically cohesive units (*take advantage of*), are frequent and central to fluent usage (Erman and Warren, 2000; Alves et al., 2024), and exhibit diverse realizations across languages (Savary et al., 2018). In this work, MWEs are operationalized as *contiguous* sequences of orthographic words forming a single unit at one or more linguistic levels. Contiguous MWEs are attested across all evaluated languages, spanning several common linguistic categories. Proper noun MWEs occur across all languages in country names (e.g., *die Vereinigten Staaten* “the United States”), person names (e.g., *Rosa Parks*) and organisation names (e.g., *Unione Europea* “European Union”). Idiomatic adverbials are likewise widespread, including German *auf einmal* (“suddenly”) and Dutch *in ieder geval* (“in any case”). Compound nouns are common in English: *traffic jam*, *apple tree*. Further examples are shown in Table 1.

MWEs are challenging for compositional semantic interpretation because their meaning does not follow compositionally from the meaning of their parts. The Parallel Meaning Bank (PMB; Abzianidze et al., 2017) addresses this issue by using a non-standard tokenization approach in which multiple consecutive standard orthographic tokens are merged into a single unit before subsequent processing steps such as semantic tagging, syntactic parsing, and compositional interpretation. This approach only applies to contiguous MWEs and not to discontiguous ones, which must be handled via other mechanisms (Evang et al., 2025).

While this tokenization schema improves the representation of contiguous MWEs for downstream semantics, it complicates the use of standard tokenizers for compositional semantic parsing on the PMB such as Shen and Evang (2022), where mismatches between off-the-shelf tokenizers and MWE-aware gold tokenization introduce avoidable errors. Despite the centrality of tokenization in NLP pipelines, there is surprisingly little systematic evidence on how rule-based, neural, and transformer-based tokenizers perform on MWE-aware segmentation, especially in a multilingual setting.

1.2 Tokenization

Traditional tokenizers, including rule-based systems (Marcus et al., 1993; Schmid, 1994) and statistical models (Church, 1988), struggle to keep MWEs intact, and probabilistic methods biased toward frequent patterns often fragment rarer or domain-specific MWEs (Collobert et al., 2011; Alvarez and Smith, 2025), propagating errors into parsing, NER, and semantic tagging (Constant et al., 2017). Related studies have explored BPE-based subword tokenization by allowing space-bridging tokens (Kumar and Thawani, 2022) and found that purely frequency-based MWE modeling degrades model performance. Modern tools illustrate this spectrum: UDPipe (Straka et al., 2016) assumes whitespace-based tokenization and cannot merge multiple tokens into a single multiword unit (Straka and Straková, 2017); Elephant (Evang et al., 2013) operates at the character level with STIO tags but relies on outdated neural models; MaChAmp (van der Goot et al., 2021) offers flexible neural sequence labeling with transformer encoders and has been used for BIO tagging of subwords for tokenization (van der Goot, 2024) but not for MWE-aware tokenization.

MWE recognition itself is typically cast as sequence labeling over *pre-tokenized* input, supported by resources such as PARSEME and DiM-

SUM (Savary et al., 2017; Schneider et al., 2016). State-of-the-art systems use BIO-style tagging with BiLSTMs, CRFs, or transformers (Klyueva et al., 2017; Moreau et al., 2018; Rohanian et al., 2019; Avram et al., 2023), sometimes enriched with syntactic or semantic features (Taslimipoor et al., 2019; Fakharian and Cook, 2021). However, they generally take token boundaries for granted and therefore offer limited guidance on how to build tokenizers that are themselves MWE-aware.

1.3 Contributions

Several gaps remain. There is little systematic evaluation of how different tokenization strategies – rule-based, statistical, neural, and transformer-based – perform specifically on *contiguous MWE segmentation* in a multilingual setting. The interaction between tagging scheme (STIO vs. character-level BIO), and architecture (classical vs. transformer-based) has not been explored in a controlled setting.

We address these gaps by treating MWE-aware tokenization as a sequence labeling problem under varying tagging schemes, supervision sources, and model architectures within a shared evaluation framework on the task of identifying contiguous MWEs in the PMB for English, German, Italian, and Dutch. The resulting analysis is intended both to benchmark existing systems and to inform the design of future tokenizers that align more closely with linguistically motivated resources.

2 Method

2.1 Task and Data

We use PMB version 4.0.0 (Abzianidze et al., 2017), the most recent release that provides explicit tokenization information at the character-level in the form of STIO tags as shown in Figure 1a. The corpus contains 10 715 English, 2 844 German, 1 686 Italian, and 1 467 Dutch gold documents. Only the *gold* tier is used. In the English gold data, for example, 1 913 documents contain at least one whitespace character tagged `I`, indicating merged multiword units and confirming the relevance of MWEs for this resource. All four datasets are split into training, development, and test sets using an 80:10:10 ratio.

2.2 Models

We compare four different configurations for transforming raw text documents into token sequences. They are summarized in Table 2 and described in the following.

Model	Input	Tags	Output
UDPipe	characters	–	tokens
Elephant	characters	STIO	tokens
MaChAmp_standalone	characters	BIO	tokens
MaChAmp_post	UDPipe tokens	BIO	tokens

Table 2: Overview of evaluated configurations

The **`UDPipe`** configuration serves as a non-trainable baseline. We use the UDPipe software (Straka et al., 2016, v1.3.1) with pretrained standard tokenization models. It produces "pretokens" as shown in the first line of Figure 1c. It cannot merge multiple whitespace-separated pretokens into a single multiword token (Straka and Straková, 2017) and thus always oversegments cases such as *New York* or *in spite of*.

The following language-specific models were employed: english-ewt-ud-2.5-191206, german-hdt-ud-2.5-191206, italian-isdt-ud-2.5-191206, and dutch-lassysmall-ud-2.5-191206 (Straka and Straková, 2019).

The **`Elephant`** configuration uses the Elephant tokenizer (Evang et al., 2013). It uses a character-level neural model that combines a Simple Recurrent Network with a CRF tagger. It predicts STIO tags over raw text as shown in Figure 1a. In our experiments, we use the original model configuration and feature templates provided by the authors.

The **`MaChAmp_standalone`** configuration uses the MaChAmp software (van der Goot et al., 2021, v0.4.2). MaChAmp is a neural multi-task framework built on transformer encoders. Although its BIO-tagging functionality is designed to operate at the token level, in this configuration we use it at the character-level, thus as a character-level sequence labeling tokenizer like Elephant, the only essential differences being the more modern neural architecture (Transformers instead of SRN+CRF) and the lack of an S/T distinction, thus working with the character-level BIO scheme shown in Figure 1b. Since our focus is on tokenization and not on sentence segmentation, this is not a problem.

Finally, the **`MaChAmp_post`** configuration also uses MaChAmp's BIO-tagging functionality, but this time at the pretoken level and as a *postprocessing* step for UDPipe output. Here, MaChAmp corrects UDPipe's oversegmentation by recognizing contiguous MWEs and marking them with `B`, `I` tags as shown in Figure 1c.

We test the MaChAmp models with two different underlying encoders: the default multilingual cased BERT model (Devlin et al., 2019) and XLM-RoBERTA (Conneau et al., 2020).

2.3 Hyperparameters

To keep the comparison focused on model design rather than hyperparameter tuning, we largely rely on default settings. MaChAmp models are trained with their default hyperparameters for all languages. For Elephant, we tried the predefined configurations from Evang et al. (2013) and selected the best-performing feature set for each language on the development data.

3 Results and Discussion

We compare the different configurations in terms of span level F1 score (Section 3.1), pretoken-level labeling accuracy for the postprocessing models (Section 3.2), character-level labeling accuracy (Section 3.3), and document-level error rate (Section 3.4). We furthermore analyze error patterns through tag confusion matrices (Section 3.5) and perform a qualitative analysis of a sample to gauge the frequency of different error types (Section 3.6).

3.1 Span-level Evaluation

A tokenizer should correctly recognize a large proportion of tokens in the gold test data (recall) and predict few or no false tokens (precision). F1 score is the harmonic mean of recall and precision and serves as an overall performance indicator. For this evaluation, we convert the output of all configurations to the pretoken-level BIO format. A predicted span (a pretoken labeled `B` followed by zero or more pretokens labeled `I`) is counted as correct only if it exactly matches the corresponding span in the gold data. We show the F1 scores of the different configurations in Table 3.

The two postprocessing models consistently obtain the highest span-level F1-scores, with `MaChAmp_post_xlmr` ranking first in English, German, and (jointly) Dutch, and `MaChAmp_post` ranking first in Italian and (jointly) Dutch. The standalone MaChAmp variants (`MaChAmp_standalone`, `MaChAmp_standalone_xlmr`) are competitive but systematically outperformed by the UDPipe-postprocessing models. `Elephant` performs moderately well across languages, whereas `UDPipe` yields the lowest F1-scores overall.

Model	EN	DE	IT	NL
UDPipe	0.9470	0.9632	0.9678	0.9710
Elephant	0.9524	0.9594	0.9819	0.9741
MaChAmp_standalone	0.9809	0.9789	0.9883	0.9827
MaChAmp_standalone_xlmr	0.9818	0.9775	0.9846	0.9843
MaChAmp_post	0.9906	0.9843	**0.9899**	**0.9913**
MaChAmp_post_xlmr	**0.9909**	**0.9852**	0.9883	**0.9913**

Table 3: Span-level F1-scores across languages

Across languages, the MaChAmp postprocessing models improve span-level F1 by approximately 2–4.5 points over UDPipe and 1–3 points over Elephant. Gains are largest in English and Dutch, the most MWE-dense languages, suggesting that contextual postprocessing is particularly beneficial in structurally complex settings. These results support the central hypothesis of this work: a two-stage UDPipe–MaChAmp pipeline yields the strongest overall performance for MWE recognition.

3.2 Pretoken-level Evaluation for Postprocessing Models

For `MaChAmp_post`, we compute pretoken-level precision, recall, and F1 over BIO labels assigned to UDPipe pretokens. These pretoken-level scores serve as a diagnostic, indicating to what extent the postprocessing step directly corrects MWE-specific errors introduced by UDPipe. The scores are shown in Tables 4 and 5.

Language	Precision	Recall	F1-Score
English	0.9156	**0.8918**	**0.9035**
German	0.8846	0.7931	0.8364
Italian	0.8667	0.7647	0.8125
Dutch	**0.9286**	0.8125	0.8667

Table 4: Token-level accuracy (MaChAmp_post)

Language	Precision	Recall	F1-Score
English	0.9043	**0.9004**	**0.9024**
German	0.9200	0.7931	0.8519
Italian	0.7895	0.8824	0.8333
Dutch	**0.9286**	0.8125	0.8667

Table 5: Pretoken-level accuracy (MaChAmp_post_xlmr)

For English, pretoken-level F1 of around 0.90 for both pipeline models aligns with the substantial increase in span-level F1 (from 0.947 for UDPipe

to roughly 0.991), indicating that many UDPipe MWE-specific segmentation errors are directly corrected in the postprocessing stage. German and Dutch exhibit a similar pattern, with token-level F1 in the mid-0.8 range and span-level F1 gains of approximately 2–3 points over UDPipe. Italian, despite having the lowest token-level F1 among the four languages, still shows sizable improvements at the span level, suggesting that each corrected MWE in this less MWE-dense language has a comparatively larger impact on aggregate span-based metrics.

From an architectural perspective, the span-level and pretoken-level results illustrate the advantage of transformer-based sequence labeling systems. UDPipe exemplifies the limitations of standard tokenizers without MWE treatment: it provides a robust baseline for generic tokenization, but its inability to recover MWEs leads to systematically lower span-level F1. Elephant, while stronger than UDPipe, does not reach the accuracy of MaChAmp. In contrast, MaChAmp makes use of multilingual contextual embeddings to detect and reconstruct MWE spans, and its postprocessing stage effectively turns UDPipe's under-recognition of MWEs into competitive, linguistically informed tokenization. The addition of XLM-RoBERTa occasionally yields small improvements over the default mBERT transformer, but these gains are generally marginal; this suggests that the main advantage comes from the pipeline architecture rather than from a specific underlying transformer.

3.3 Character-level Evaluation

In this subsection we report character-level accuracy, which captures how reliably models assign STIO labels to individual characters and is useful for gauging overall label consistency. However, character-level metrics are dominated by frequent tags, in particular `I`, and can therefore give an overly optimistic picture. A model that mislabels only a single character as `T` in the middle of an MWE (e.g., for `New York`: `TIIITIII` instead of `TIIIIIII`) may achieve very high character accuracy while completely fragmenting the expression (tokens *New* , *York* instead of *New York*). For this evaluation, we convert the output of all configurations to the character-level STIO resp. BIO format. We also report an *adjusted* accuracy that ignores confusion of the labels S and T, which the MaChAmp models do not distinguish.

English, the most MWE-dense language in the evaluation, illustrates the relationship between character-level accuracy and span-level F1 (Table 6). Character-level accuracy is uniformly high across models, whereas span-level F1 differentiates performance more clearly. After adjusting for $S \leftrightarrow T$ confusion, the gold-trained MaChAmp models reach accuracies comparable to those of the postprocessing models but still lag behind in span-level F1, indicating that the pipeline recovers more complete MWE spans.

Model	Acc.	Adj. Acc.[1]	Span F1
UDPipe	0.9821	0.9821	0.9470
Elephant	0.9843	0.9843	0.9524
MaChAmp_standalone	0.9594	0.9940	0.9809
MaChAmp_standalone_xlmr	0.9596	0.9942	0.9818
MaChAmp_post	0.9970	0.9970	0.9906
MaChAmp_post_xlmr	**0.9972**	**0.9972**	**0.9909**

Table 6: English character accuracy and span-level F1

The same qualitative pattern holds for the remaining languages.

In all four languages, raw character-level accuracies of UDPipe and Elephant exceed 0.98, which would suggest similar performance if only character-level metrics were considered. However, their span-level F1-scores clearly lag behind those of all MaChAmp configurations, revealing frequent errors in MWE boundary detection. The UDPipe+MaChAmp pipeline remains superior in span-level F1, underscoring the importance of span-based evaluation for this task.

3.4 Document-Level Evaluation

In this subsection, we report document-level error rate (percentage of documents with at least one tagging error) as a coarse indicator of practical reliability in downstream pipelines. This metric is particularly informative in combination with the proportion of sentences containing MWEs: in our test sets, English has the highest proportion of MWE-containing sentences (19.78%), followed by Dutch (10.14%) and German (8.77%), while Italian has the lowest proportion (7.65%). Higher MWE density generally correlates with higher tagging difficulty, and thus with higher document-level error rates for weaker models.

Table 7 reports the percentage of sentences containing at least one tagging error. The MaChAmp-post models achieve the best results, reducing document-level error rates by a substantial mar-

[1]Adjusted accuracy excludes $S \leftrightarrow T$ confusion.

Model	EN	DE	IT	NL
UDPipe	20.80	11.23	9.41	10.81
Elephant	18.84	12.98	5.88	9.46
MaChAmp_standalone	7.56	7.72	4.12	6.76
MaChAmp_standalone_xlmr	7.28	7.72	4.71	6.08
MaChAmp_post	**3.73**	4.91	3.53	3.38
MaChAmp_post_xlmr	**3.73**	**4.56**	**4.12**	**3.38**

Table 7: Document-level error percentages

gin across all languages. In English, for instance, the error rate decreases from 20.8% for UDPipe to 3.73% for the best MaChAmp model, a reduction by roughly a factor of five. English exhibits the highest overall error rates, consistent with its higher MWE density and larger proportion of rare MWE types.

Combined with the MWE percentages, these results suggest that MWEs are a major source of difficulty for weaker models: the language with the highest proportion of MWE-containing sentences (English) also shows the highest document-level error rates for UDPipe and Elephant, whereas Italian – the language with the lowest MWE proportion – exhibits the lowest error rates for the same systems. The UDPipe+MaChAmp pipeline substantially narrows these gaps, achieving document-level error rates between 3.38% and 4.91% even in MWE-dense English, indicating that most sentences are correctly segmented.

From an applied perspective, document-level reliability is crucial for downstream tasks such as compositional semantic parsing, named entity recognition, or machine translation, which typically operate on full sentences. While high character-level accuracy might suggest that all systems are adequate, the proportion of sentences containing any tokenization error provides a more realistic estimate of how often downstream models will encounter structurally flawed input. The fact that the UDPipe-MaChAmp pipeline reduces document-level error rates to below 5% across languages means that segmentation errors become relatively rare, whereas rule-based UDPipe alone missegments more than one in five English sentences.

3.5 Error Patterns from Confusion Matrices

In this subsection, we return to the character-level evaluation of Subsection 3.3, where we converted the output of all systems to the STIO format. We now present confusion matrices of the character labels `S`, `T`, `I`, and `O` to better understand where systems go wrong.

	I	O	S	T
I	19143	**206**	0	**213**
O	27	4600	0	2
S	0	0	1081	**2**
T	40	0	0	5929

Table 8: Representative confusion matrix (UDPipe, English)

The confusion matrices shown in Tables 8–12 reveal significant differences in how systems handle token boundaries. UDPipe (Table 8) exhibits a strong bias toward oversegmentation: characters belonging inside MWEs (gold label `I`) are frequently labeled `O`, `S`, or `T` instead. In English, 97.8% of all UDPipe misclassifications fall into `I`→`O` or `I`→`T`, and similar rates (above 90%) hold for German and Dutch. Elephant shares the same qualitative bias but with a somewhat lower proportion of oversegmentation errors.

	I	O	S	T
I	2836	**3**	0	**3**
O	**3**	582	0	0
S	0	0	170	0
T	**5**	0	0	763

Table 9: Confusion matrix (MaChAmp-post XLM-R, Italian)

The UDPipe+MaChAmp postprocessing models (Table 9) substantially reduce this oversegmentation. Overall error counts drop sharply, and the remaining misclassifications are more evenly distributed across `I`/`O`/`T`. In English, German, and Dutch the proportion of `O`→`I` versus `I`→`O`/`I`→`T` errors is relatively balanced, indicating that the pipeline does not introduce a strong systematic bias in either direction. The main change compared to UDPipe is therefore not a shift from over- to undersegmentation, but a strong reduction in boundary errors in general, with only a mild tendency toward merging boundaries in some languages. The clearest case of undersegmentation occurs in Italian, where `O`→`I` constitutes roughly 57% of all misclassifications for the MaChAmp-post XLM-R model, possibly due to the predominance of preposition–article contractions (e.g. *a* + *il* → *al*, *di* + *lo* → *dello*) in Italian.

	I	O	S	T
I	19469	**43**	0	**50**
O	**40**	4587	0	2
S	0	0	0	**1083**
T	**50**	1	0	5918

Table 10: Confusion matrix (MaChAmp-standalone, English)

	I	O	S	T
I	19456	**51**	0	**55**
O	**32**	4595	0	2
S	0	0	0	**1083**
T	**40**	0	0	5929

Table 11: Confusion matrix (MaChAmp-standalone XLM-R, English)

MaChAmp-standalone models (Tables 10, 11) display comparatively balanced segmentation behaviour once S→T confusion artifacts are excluded. In English, the over-/undersegmentation split is approximately even (51/49 for the base model; 60/40 for XLM-R), as reflected in their confusion matrices.

	I	O	S	T
I	2828	**6**	0	**8**
O	0	585	0	0
S	0	0	0	**170**
T	0	0	0	768

Table 12: Confusion matrix (MaChAmp-standalone, Italian)

For Italian, Dutch, and German, the same models show a somewhat clearer tendency toward oversegmentation. In Italian, for instance, nearly all misclassifications correspond to I→O or I→T errors (Table 12).

Overall, the systems form a continuum. UDPipe strongly favours splitting and never recovers full MWE spans. Elephant reduces this bias but still oversegments many MWEs, especially compound nouns. The UDPipe+MaChAmp pipeline drastically reduces segmentation errors and, apart from a mild undersegmentation tendency in Italian, behaves fairly symmetrically with respect to over- and undersegmentation. The MaChAmp-standalone models show a balanced behavior but without the additional corrections afforded by the UDPipe-based postprocessing configuration.

3.6 Error Analysis

English exhibits by far the highest MWE density and therefore provides the clearest picture of system behavior. Across all configurations, the English data yielded 664 sentences with at least one tagging error and a total of 1 595 tagging errors. Given this volume, a full manual review was not feasible; instead, a randomized subset of 30 documents per model was inspected, focusing on recurring phenomena and grouping errors into linguistic categories. In contrast, the Dutch, German, and Italian test sets contain comparatively few MWEs, limiting the extent to which language-specific generalizations can be drawn for those languages.

Counts in Table 13 reflect the number of individual tagging errors observed in the sampled sentences, grouped into linguistically motivated categories reflecting different types of segmentation errors.

UDPipe produces the highest rate of oversegmentation errors across all languages. Compound nouns like *credit card*, *apple juice*, or *stomach ache* are regularly split, as are named entities such as *Charles de Gaulle* and frequent expressions such as *fed up*, *kind of*, *split up*, and *in love*. The same holds for complex adverbials such as *all of a sudden*, *out of the blue*, Italian *su per giù* (“more or less”), and Dutch *keer op keer* (“again and again”).

MaChAmp-post, operating as a postprocessing module over UDPipe’s output, demonstrably reduces much of this fragmentation. In order to assess how effective this pipeline is at recovering MWEs, all UDPipe errors were compared side by side, showing which ones were successfully recovered as intended. The model frequently re-tags MWEs like *fed up*, *kind of*, and *so-so* as cohesive units and restores compound nouns such as *traffic jam*, *bank account*, and *mother tongue*. It also improves the treatment of multiword named entities: examples such as *Mt. Fuji*, *Charles de Gaulle*, or *European Union* are correctly merged where UDPipe had split them. Nonetheless, improvements are not uniform. Rare or syntactically atypical expressions, nested quotes, and unusual punctuation patterns often remain problematic, suggesting that MaChAmp inherits some limitations from UDPipe’s segmentation or from biases in its own training data. Differences between the base and XLM-RoBERTa variants are subtle in this qual-

Model		UDPipe	Elephant	MaChAmp-standalone		MaChAmp-post	
				mBERT	XLM-R	mBERT	XLM-R
Oversegmentation type	Example						
Compound noun	*cherry tree*	8	11	10	9	8	7
Proper Noun	*Leo Tolstoy*	13	8	0	2	3	2
Idiomatic MWE	*good for nothing*	6	6	2	2	4	4
Verbal / Predicate MWE	*fed up*	1	2	1	4	1	1
Numeric expression	*8:30 a.m.*	2	4	1	1	1	2
Undersegmentation type	Example						
Overmerge	*home as*	0	3	13	15	9	9
Non-whitespace token boundary	*Brian's*	1	2	3	3	4	7

Table 13: Error types for English in a sample of 30 error documents per model

itative analysis; no systematic advantage for one or the other emerges beyond the small quantitative gains reported earlier.

Elephant exhibits error patterns that are broadly similar to those of UDPipe, particularly in its failure to treat many compound nouns as single lexical units. At the same time, it is more robust for MWEs that include punctuation: expressions like *Mt. Fuji* and *8:30 a.m.* are often tagged as complete units. Elephant is, however, more prone than UDPipe to undersegmentation or overmerging. It occasionally incorporates surrounding modifiers, determiners, or even adjacent verbs that are not part of the intended expression, reflecting a tendency to rely on local orthographic and contextual cues in the absence of deeper semantic modeling.

The MaChAmp-standalone models perform markedly better than both UDPipe and Elephant in MWE segmentation. Their most consistent strength lies in the treatment of multi-token named entities, especially person names: sequences such as *Guus Hiddink* and *Eda Charlton* are correctly recognized and kept intact. They also outperform the baselines on common compound nouns, reliably labeling expressions such as *laptop computer* and *word processor* as single spans. A recurring peculiarity in these models is a tendency to undersegment common syntactic structures that are not lexicalized MWEs, for example *There [are millions] of stars in the universe* or *[Can I] use one of yours?*. Similar issues arise in sentence-final collocations like *Let's go to the [theater together]* or embedded coordinations such as *He put milk into his [tea and] stirred it*, where syntactic cohesion appears to be misinterpreted as evidence for multiword status.

Overall, the error analysis highlights systematic over-segmentation in standard tokenizers, which is improved by multiword-aware tokenizers. The UDPipe-MaChAmp postprocessing pipeline is especially effective at recovering MWEs that are frequent and orthographically regular, while rare, irregular, or borderline cases remain challenging across architectures. At a higher level, these findings clarify the role of architecture and supervision in MWE-aware tokenization. UDPipe represents the limitations of standard tokenizers: it achieves high accuracy overall, but does not recognize MWEs. Elephant improves upon this but cannot match the precision of transformer-based models. MaChAmp, by using transformer-based contextual representations, and particularly when combined with UDPipe in a postprocessing pipeline, yields robust MWE segmentation with low error rates. This demonstrates that combining standard tokenization with transformer-based postprocessing offers a powerful and generalizable strategy for accurate multilingual MWE-aware tokenization, and sets a clear direction for future work on joint models that integrate segmentation with higher-level linguistic analysis.

4 Conclusion

The results confirm that transformer-based neural models, especially when used in a postprocessing pipeline on top of rule-based tokenization, offer the most effective solution for MWE-aware tokenization. The most successful configuration combines UDPipe's initial segmentation with MaChAmp's context-sensitive sequence labeling.

At the same time, the analysis highlights that the span-level F1 is more informative than the character-level accuracy in this setting. Models such as `UDPipe` achieve very high character-level scores while still failing to segment MWEs cor-

rectly. In contrast, the postprocessing pipeline yields more balanced behavior, recovering many MWEs that rule-based systems systematically fragment and reducing document-level error rates.

Overall, the findings suggest that accurate tokenization in multilingual NLP requires architectures that integrate local and global context and can adapt to segmentation irregularities. The framework developed here by combining modular tokenizers and sequence labelers in a plug-and-play fashion offers a replicable basis for future work. Promising extensions include handling discontinuous MWEs, adapting the approach to under-resourced languages, and exploring joint models that integrate segmentation with higher-level linguistic analysis.

Limitations

A central limitation of this work is its restriction to four closely related Indo-European languages, English, German, Dutch, and Italian, which, despite some systematic differences, share typological characteristics such as relatively fixed word order, moderate morphological complexity, and whitespace-based tokenization. This raises questions about how well the findings would generalize to typologically distant languages, including agglutinative, polysynthetic, or logographic systems.

A further limitation concerns the exclusive focus on contiguous MWEs as defined in the PMB. Discontinuous, syntactically flexible or clause-spanning MWEs, such as phrasal verbs with intervening material or idioms distributed across syntactic boundaries, remain outside the scope of the present study, leaving open whether current architectures would handle such structures effectively.

Acknowledgments

We wish to thank the anonymous reviewers for their valuable feedback. Kilian Evang's work on this paper was supported by grants no. 467699802 (MWE-SemPrE) and 560341082 (Superframes) of the German Research Foundation (DFG).

References

Lasha Abzianidze, Johannes Bjerva, Kilian Evang, Hessel Haagsma, Rik van Noord, Pierre Ludmann, Duc-Duy Nguyen, and Johan Bos. 2017. The Parallel Meaning Bank: Towards a multilingual corpus of translations annotated with compositional meaning representations. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 242–247, Valencia, Spain. Association for Computational Linguistics.

Juan Alvarez and Jane Smith. 2025. Limitations of tokenizers for building a neuro-symbolic lexicon. In *Proceedings of the International Conference on Computational Linguistics*. SciTePress.

Diego Alves, Stefan Fischer, Stefania Degaetano-Ortlieb, and Elke Teich. 2024. Multi-word expressions in English scientific writing. In *Proceedings of the 8th Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature (LaTeCH-CLfL 2024)*, pages 67–76, St. Julians, Malta. Association for Computational Linguistics.

Andrei-Marius Avram, Verginica Barbu Mititelu, Vasile Păiș, Dumitru-Clementin Cercel, and Ștefan Trăușan-Matu. 2023. Multilingual multiword expression identification using lateral inhibition and domain adaptation. *Mathematics*, 11(11).

Timothy Baldwin and Su Nam Kim. 2010. Multiword expressions. In Nitin Indurkhya and Fred J. Damerau, editors, *Handbook of Natural Language Processing*, pages 267–292. Chapman and Hall/CRC.

Kenneth W. Church. 1988. A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the Second Conference on Applied Natural Language Processing*, pages 136–143.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Mathieu Constant, Gülşen Eryiğit, Johanna Monti, Lonneke van der Plas, Carlos Ramisch, Michael Rosner, and Amalia Todirascu. 2017. Survey: Multiword expression processing: A Survey. *Computational Linguistics*, 43(4):837–892.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Britt Erman and Beatrice Warren. 2000. The idiom principle and the open choice principle. *Text & Talk*, 20.

Kilian Evang, Valerio Basile, Grzegorz Chrupała, and Johan Bos. 2013. Elephant: Sequence labeling for word and sentence segmentation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1422–1426, Seattle, Washington, USA. Association for Computational Linguistics.

Kilian Evang, Rafael Ehren, and Laura Kallmeyer. 2025. The proper treatment of verbal idioms in German discourse representation structure parsing. In *Proceedings of the 16th International Conference on Computational Semantics*, pages 156–165, Düsseldorf, Germany. Association for Computational Linguistics.

Samin Fakharian and Paul Cook. 2021. Contextualized embeddings encode monolingual and cross-lingual knowledge of idiomaticity. In *Proceedings of the 17th Workshop on Multiword Expressions (MWE 2021)*, pages 23–32, Online. Association for Computational Linguistics.

Tatsuya Hiraoka, Sho Takase, Kei Uchiumi, Atsushi Keyaki, and Naoaki Okazaki. 2021. Joint optimization of tokenization and downstream model. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 244–255, Online. Association for Computational Linguistics.

Natalia Klyueva, Antoine Doucet, and Milan Straka. 2017. Neural networks for multi-word expression detection. In *Proceedings of the 13th Workshop on Multiword Expressions (MWE 2017)*, pages 60–65. Association for Computational Linguistics.

Dipesh Kumar and Avijit Thawani. 2022. BPE beyond word boundary: How NOT to use multi word expressions in neural machine translation. In *Proceedings of the Third Workshop on Insights from Negative Results in NLP*, pages 172–179, Dublin, Ireland. Association for Computational Linguistics.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

Erwan Moreau, Ashjan Alsulaimani, Alfredo Maldonado, Lifeng Han, Carl Vogel, and Koel Dutta Chowdhury. 2018. Semantic reranking of CRF label sequences for verbal multiword expression identification. In Stella Markantonatou, Carlos Ramisch, Agata Savary, and Veronika Vincze, editors, *Multiword Expressions at Length and in Depth: Extended Papers from the MWE 2017 Workshop*, pages 177–207. Language Science Press.

Omid Rohanian, Shiva Taslimipoor, Samaneh Kouchaki, Le An Ha, and Ruslan Mitkov. 2019. Bridging the gap: Attending to discontinuity in identification of multiword expressions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2692–2698. Association for Computational Linguistics.

Ivan A Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. 2002. Multiword expressions: A pain in the neck for nlp. *Lecture Notes in Computer Science*, 2276:1–15.

Agata Savary, Marie Candito, Verginica Barbu Mititelu, Eduard Bejček, Fabienne Cap, Slavomír Čéplö, Silvio Ricardo Cordeiro, Gülşen Eryiğit, Voula Giouli, Maarten van Gompel, Yaakov HaCohen-Kerner, Jolanta Kovalevskaitė, Simon Krek, Chaya Liebeskind, Johanna Monti, Carla Parra Escartín, Lonneke van der Plas, Behrang QasemiZadeh, Carlos Ramisch, and 3 others. 2018. PARSEME multilingual corpus of verbal multiword expressions. In Stella Markantonatou, Carlos Ramisch, Agata Savary, and Veronika Vincze, editors, *Multiword Expressions at Length and in Depth: Extended Papers from the MWE 2017 Workshop*, pages 87–147. Language Science Press, Berlin.

Agata Savary, Carlos Ramisch, Veronika Vincze, Silvio Ricardo Cordeiro, Johanna Monti, and 1 others. 2017. The PARSEME shared task on automatic identification of verbal multiword expressions. In *Proceedings of the 13th Workshop on Multiword Expressions (MWE 2017)*, pages 31–47. Association for Computational Linguistics.

Helmut Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing*, pages 44–49. University of Manchester.

Nathan Schneider, Emily Danchik, Chris Dyer, and Noah A. Smith. 2016. SemEval-2016 task 10: Detecting minimal semantic units and their meanings (DiMSUM). In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 546–559. Association for Computational Linguistics.

Minxing Shen and Kilian Evang. 2022. DRS parsing as sequence labeling. In *Proceedings of the 11th Joint Conference on Lexical and Computational Semantics*, pages 213–225, Seattle, Washington. Association for Computational Linguistics.

Milan Straka, Jan Hajič, and Jana Straková. 2016. UDPipe: Trainable pipeline for processing CoNLL-U files performing tokenization, morphological analysis, POS tagging and parsing. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 4290–4297, Portorož, Slovenia. European Language Resources Association (ELRA).

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.

Milan Straka and Jana Straková. 2019. Universal dependencies 2.5 models for UDPipe (2019-12-06). LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL).

Shiva Taslimipoor, Omid Rohanian, and Le An Ha. 2019. Cross-lingual transfer learning and multitask learning for capturing multiword expressions. In *Proceedings of the Joint Workshop on Multiword Expressions and WordNet (MWE-WN 2019)*, pages 155–161, Florence, Italy. Association for Computational Linguistics.

Rob van der Goot. 2024. Where are we still split on tokenization? In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 118–137, St. Julian's, Malta. Association for Computational Linguistics.

Rob van der Goot, Ahmet Üstün, Alan Ramponi, Ibrahim Sharaf, and Barbara Plank. 2021. Massive choice, ample tasks (MaChAmp): A toolkit for multitask learning in NLP. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 176–197, Online. Association for Computational Linguistics.