PaReNT (Parent Retrieval Neural Tool): A Deep Dive into Word Formation Across Languages

Emil Svoboda, Magda Ševčíková
Charles University, Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics
Malostranské náměstí 25, Prague, Czech Republic
{svoboda, sevcikova}@ufal.mff.cuni.cz

Abstract

We present *PaReNT* (Parent Retrieval Neural Tool), a deep-learning-based multilingual tool performing *parent retrieval* and *word-formation classification* in English, German, Dutch, Spanish, French, Russian, and Czech. *Parent retrieval* refers to determining the lexeme or lexemes the input lexeme was based on (e.g. 'darkness' is traced back to 'dark'; 'waterfall' decomposes into 'water' and 'fall'). Additionally, *PaReNT* performs *word-formation classification*, which determines the input lexeme as a *compound* (e.g. 'proofread'), a derivative (e.g. 'deescalate') or as an unmotivated word (e.g. 'dog'). These seven languages are selected from three major branches of the Indo-European language family (Germanic, Romance, Slavic). Data is aggregated from a range of word-formation resources, as well as Wiktionary, to train and test the tool. The tool is based on a custom-architecture hybrid transformer block-enriched sequence-to-sequence neural network utilizing both a character-based and semantic representation of the input lexemes, with two output modules – one decoder-based dedicated to parent retrieval, and one classifier-based for word-formation classification. *PaReNT* achieves a mean accuracy of 0.62 in parent retrieval and a mean balanced accuracy of 0.74 in word-formation classification.

Keywords: word formation, deep learning, morphology

1. Introduction

A significant portion of words in a language share one or more roots with other existing lexemes as a result of various word-formation processes. However, it is difficult to automatically determine what the motivating lexeme(s) looked like. As an example, consider the English word *backache* (example 1). In a computational setting, it is not easy to even determine that it is a compound, as opposed to being a derivative of a single parent like example 2 or an unmotivated word with no parents like example 3.

(1) English

backache → ***back*** ***ache***
N N N

(2) English

backness → ***back***
N N

(3) English

baklava → λ
N $None$

To handle this problem, we present *PaReNT*, a deep-learning tool that accepts words in their dictionary forms (lemmas) from a given language and returns the lemmas of their parent words. It a) handles both attested and newly-coined lexemes, b) its inputs are not limited to any particular word-formation category and c) it returns a valid lexeme in its lemma form in the given language as output. To the best of our knowledge, no other specialized computational tool available today satisfies all of the three conditions. *PaReNT* handles English, German, Dutch, French, Spanish, Czech, and Russian, and does not depend on a dictionary. *PaReNT* performs two related tasks:

1. *Parent retrieval*. Predict which word or words the input lemma is motivated by.
2. *Word-formation classification*. Predict whether the input lemma is a *compound*, a *derivative*, or *unmotivated*.

Even though tools performing partial or related tasks exist, no currently available computational tool specialized in end-to-end parent retrieval, which we will expand upon in Section 2. In Section 3, we will describe which data sources we used to build our data set, how we built *PaReNT*, what metrics we used to evaluate it and what baselines we used to compare it to in Section 4. Finally, we place our findings into of a wider context in Section 5, and analyze the errors of the tool in the same section. We conclude our paper in Section 6.

2. Related work

Our work continues existing research in stemming, morphological segmentation, derivational analysis, compound splitting, and in linguistic data re-

LREC-COLING 2024, pages 12611–12621
20-25 May, 2024. © 2024 ELRA Language Resource Association: CC BY-NC 4.0

Authors	Language(s)	Approach	POS scope	Parent no.
Khaitan et al. (2009)	en	Split-point	Any	Any
Fritzinger and Fraser (2010)	de	Split-point	Any	2
Henrich and Hinrichs (2011)	de	Valid-output	Nominal	2
Clouet and Daille (2014)	en, ru	Valid-output	Any	2
Riedl and Biemann (2016)	de, nl, en	Split-point	Any	Any
Krotova et al. (2020)	de	Split-point	Nominal	Any
Svoboda and Ševčíková (2021)	cs	Valid-output	Any	Any
Vodolazsky and Petrov (2021)	ru	Valid-output	Any	2

Table 1: Comparison of various compound splitters described in the literature, sorted by year of publication.

source building. Parent retrieval can actually be thought of as a generalization over both compound splitting and derivational analysis.

2.1. Static data resources

The most straightforward way of addressing the issue of parent retrieval is to hand-build a dictionary-like data resource and simply find the given word there. The number and quality of such word-formation resources is considerable. Derinet, for example, contains $1,034,354$ Czech lemmas (Vidra et al., 2021), and MorphoLex-FR covers $38,840$ French lemmas (Mailhot et al., 2020). A lookup into such a data source however carries a fundamental limitation – speakers occasionally coin new words, meaning that there will always be words not contained in any such resource. As a result, morphological and lexical databases cannot in principle have enough coverage to be considered a solution for this problem.

2.2. Stemming and morphological segmentation

Stemmers like the Porter stemmer (1980) and morphological segmentators like Morfessor (Smit et al., 2014) can both in principle handle any input lemma, but neither are designed to return valid lemmas in the language in question.[1] Stemmers are typically rule-based or statistical tools that only strip affixes, and do not handle orthographical and morphophonological changes. This often results in non-lemma outputs, such as returning **happi* when presented with *happiness*. In parent retrieval, however, we would only consider *happy* to be a correct output. Early stemmers, such as the aforementioned Porter stemmer, only stripped suffixes, with some later implementations being able to strip prefixes as well, e.g. the Arabic stemmer by Alshalabi et al. (2022) based on a programming language designed specifically for stemming called Snowball (Porter, 2001).

Morphological segmentation tools typically return a list of morphs or morphemes (e.g. [*happ*, *i*, *ness*] or [*happ*, *y*, *ness*]). For some languages, such morphs or morphemes may not be valid words. A well-known segmenter is Morfessor, introduced as unsupervised in 2002 by Creutz and Lagus, extended into semi-supervision in 2010 by Kohonen et al., and generalized beyond morphological segmentation in 2013 by Virpioja et al. It has been shown that its application on two languages can improve machine translation (Grönroos et al., 2018). Other morphological segmenters, however, are tailored to a particular language. For example, Cotterell et al. (2016) built one for English using weighted context-free grammars, and the SIGMORPHON 2022 Shared Task on Morpheme Segmentation (Batsuren et al., 2022) challenged researchers to segment words in eight different languages using training data from the Unimorph database (Batsuren et al., 2022). While this resource does cover segmentation, extracting the information is rather difficult. To address this problem, a multilingual annotation scheme for morphological segmentation has been proposed by Žabokrtský et al. (2022), potentially streamlining the development of multilingual segmenters in the future.

2.3. Compound splitters

A compound splitter (a.k.a decompounder) is any tool that takes a compound word as input and decomposes it into two or more linguistic subelements in some way. In contrast with static data resources, compound splitters are procedural, and as a result should not output an out-of-vocabulary error when presented with a novel coinage. We present a short non-exhaustive overview and classification of compound splitters that have been presented in the literature for the languages in scope; see Table 1.

Split-point compound splitters simply return the index of the place wherein the given word should

[1] In Morfessor's documentation, the term "compound" is used for any morphologically complex word – including derivatives. In contrast, this paper uses terminology congruent with generally accepted linguistic categories, and the term compound is used solely for words containing two or more roots.

Dataset	Language	Compounds	Derivatives	Unmotivated	Authors
Derinet 2.1	Czech	2,240	264,748	13,748	Vidra et al. (2021)
CELEX	Dutch	66,428	19,703	7,569	Baayen et al. (2014)
CELEX	English	6,267	15,435	14,661	Baayen et al. (2014)
Wiktionary	English	20,253	0	0	—
Unimorph	French	161	72,789	2	Batsuren et al. (2022)
MorphoLex	French	313	0	6,655	Mailhot et al. (2020)
Wiktionary	French	173	0	0	—
CELEX	German	19,304	18,372	9,140	Baayen et al. (2014)
GermaNet	German	99,080	0	0	Henrich and Hinrichs (2010)
Golden Compounds	Russian	1,699	0	0	Vodolazsky and Petrov (2021)
DerivBase.ru	Russian	0	133,645	20,612	Zeller et al. (2014)
Unimorph	Spanish	130	30,646	1	Batsuren et al. (2022)
DeriNet.ES	Spanish	0	42,825	16,141	Kyjánek et al. (2021)
Wiktionary	Spanish	329	15	0	—
All sources	All	216,377	598,178	88,529	—

Table 2: The data sources used in the training of *PaReNT*, grouped by language.

be split, which is what Khaitan et al. (2009) applied to English using normalized frequency and character n-grams. While the split-point approach allows the task to be handled as a regression or classification problem (as opposed to a sequence-to-sequence approach), the drawback is that in many languages a point-split systematically fails to yield valid lemmas. This is of little concern in English, where examples of this situation are rare, but such an approach started being a problem once the attention in NLP shifted to more morphologically complex languages.

For example, in the Dutch example 4, we see the *-e-* interfix. Inserting a split-point at *bruide.gom* would result in **bruide*, which is not a Dutch word; conversely, a split-point at *bruid.egom* results in the similarly nonsensical **egom*. This problem can be solved by building split-point splitters so that they drop interfixes, for instance by using a list of them like Henrich and Hinrichs (2011) did. However, in some languages, even interfix dropping falls short, and the split-point approach starts being impractical. For example, splitting the Czech noun *přímotop* (example 5) as *přímo.top* results in **top*, which is not a valid lemma. Instead, the appropriate output would be the verb *topit* 'to heat'. Similarly, Russian *вод.о.провод* (cf. example 6) cannot be point-split correctly, as it would yield the non-existent **вод*. Sporadically, the problem appears even in English, where *women.folk* (example 7) would yield *women*, which is a plural, and therefore not a lemma.

(4) Dutch

bruidegom → ***bruid*** ***gom***
bridegroom.N bride.N groom.N

(5) Czech

přímotop → ***přímo*** ***topit***
heater.N directly.ADV heat.V

(6) Russian

водопровод → ***вода*** ***провод***
water_piping.N water.N conduit.N

(7) English

womenfolk → ***woman*** ***folk***
N N N

Valid-output compound splitters attempt to deal with the previously described problems. This may be achieved by equipping a split-point splitter with a table of rules and/or a vocabulary or corpus, like Clouet and Daille (2014), or Vodolazsky and Petrov (2021) did, or by treating the task as a sequence-to-sequence procedure outright, which is an approach that Svoboda and Ševčíková (2021) took.

Some compound splitters restrict themselves to nominal compounds, such as the splitters by Henrich and Hinrichs (2011) of Krotova et al. (2020) for German; general compound splitters handle any compound regardless of the POS of either the parents or the compound. Finally, splitters differ in how many parents they return. Some splitters either return exactly two parents, like the splitter of Fritzinger and Fraser (2010); others return any number of parents, like Svoboda and Ševčíková's 2021 splitter.

Out of the languages in scope, compound splitters have been built for five of the seven languages. To the best of our knowledge, no splitters have been built for French or Spanish at the time of submission of this paper, with the exception of *DériF* (Namer, 2003), which only handles neoclassical compounds like those in example 8.

(8) French

psychologie → ***-psych-*** ***-log-***
psychology.N soul.NEOC. word.NEOC.

To conclude, all computational tools applicable to parent retrieval known to us carry some fundamental limitation. Static data resources do not

cover newly-coined lexemes; morphological segmentators and stemmers do not return valid lexemes in their lemma form; and compound splitters do not handle derivatives and unmotivated words. We built *PaReNT* to be a computational model that ties into the development of all of these tools, continuing the tradition of stemming and running parallel to the active field of morphological segmentation.

3. Method

The high-level overview of *PaReNT*'s functioning is simple – all available data sources are utilized to compile a flat list of lexemes in their lemma form and a corresponding list of parent sequences:

```
sea             [sea]
seasick         [sea sick]
seasickness     [seasick]
```

The list is then split into training, development, and validation sets. A deep neural model is trained on the training dataset to take a triple consisting of a special language token, the input lexeme as a sequence of character sequences, and the input lexeme as a semantic embedding, and to retrieve the sequence of its parents separated by spaces and classify the lexeme into one of: *compound*, *derivative*, *unmotivated*. Its hyperparameters are tuned using the development set, and its performance is calculated on the validation set. This process can be viewed in the third part of Figure 1.

For the purposes of this paper, we adjust two established linguistic concepts to suit our needs. First, we understand *compounds* as lexical units that regularly appear as graphical words, as opposed to compounds written with a space, which we consider syntactic constructs and therefore out-of-scope. This criterion (as Haspelmath 2017 argues) is an artifact of a combination of the Western linguistic tradition and often arbitrary orthographical conventions, but since the parent retrieval of such compounds is identical to tokenization, it was deemed uninteresting.

Second, the lexemes we deal with are categorized into one and only one of the three categories, despite the facts that a) more types of word formation exist (univerbization, blending, abbreviation, conversion, borrowing, clipping, etc...) and b) their boundaries are fuzzy, especially between derivation and compounding (e.g. is *underscore* a derivative of *score* with the prefix *under-*, or a compound of the preposition *under* and the verb *score*?). We understand that these adjustments are a simplification of actual natural language reality, but consider them a compromise.

3.1. Data

The theoretical basis of *PaReNT*'s functioning is inspired by the structure of Universal Derivations (Kyjánek et al., 2021). This is a collection of word-formation resources representing word families as directed acyclic graphs (DAGs) over a given language's lexicon, with words as nodes and relations between the words as directed labeled edges. Parenthood is indicated by the direction of the edges, meaning that the entire resource can be converted into a set of pairs, with the first element being the lexeme in question (child) and the second element its motivating lexeme(s) (parent(s); which is the given word itself for unmotivated words). Even though the labels may describe more relations between words (e.g. conversion and orthographical variation), we consider each node with a single parent to be a derivative, each node with two or more parents to be a compound, and each node with no parents to be unmotivated. This decision allowed us to utilize resources not included in Universal Derivations. Not all resources we use are structured as DAGs, but all that we use are reducible to a list of (lexeme, [parent sequence]) pairs.

The data sources used in this study were selected so that in summary, they contained at the very least several hundred examples of each of the given categories for each of the languages in scope. A concise description of the sizes of the data sources used in this study can be viewed in Table 2, alongisde their respective citations. Additionally, we crawled Wiktionary to enrich our dataset with French and Spanish compounds, because MorphoLex (2020) offered only several hundred compounds for each language. The data is split into three subsets – training, development, and validation, at a $60/20/20$ ratio. The split was done according to what we call *lexicographical blocks*. This means that lexemes belonging to the same DAG had to end up in the same subset. As a specific example, each of the German words *Arbeit*, *arbeitslos* and *Fabrikarbeit* were placed in the development subset. In total, the training set contained $543,066$ words, the development set $180,293$ words and the validation set $179,725$ words. Splitting along lexicographical blocks ensures that the model's ability to retrieve and classify lexemes bearing unseen roots is evaluated. If information about DAGs is missing from a particular dataset, as is the case with Wiktionary or GermaNet, we consider lexemes sharing the rightmost parent to belong to the same lexicographical block. Lastly, on input, every Russian word is losslessly transcribed into the Latin script, and transcribed back again on output. Evaluation is calculated in the original Cyrilic.

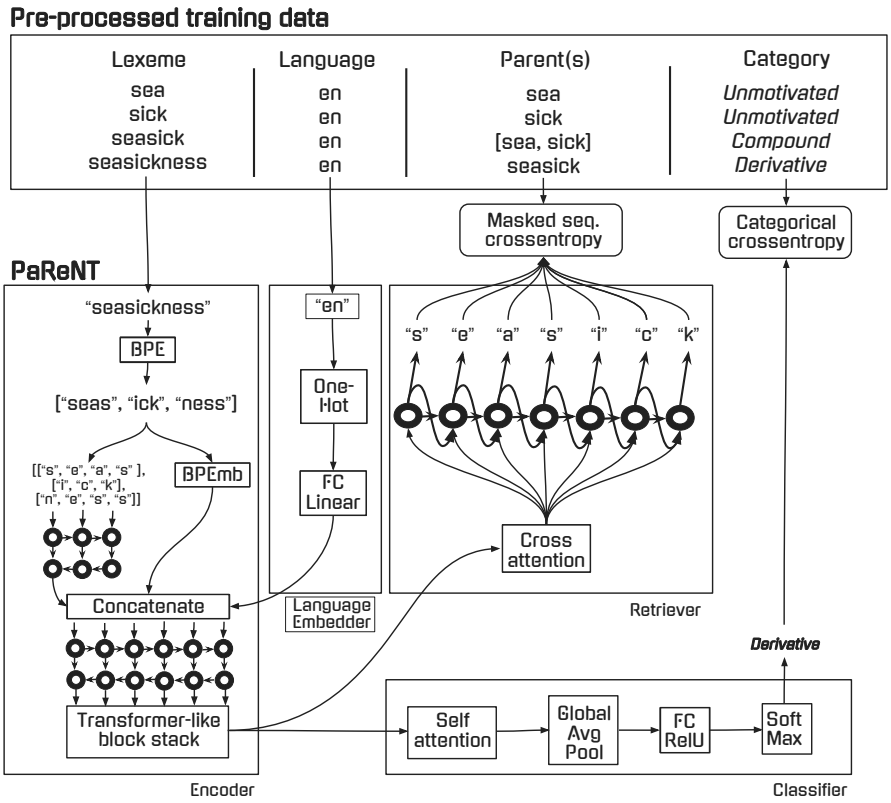

Figure 1: Visual schema of the process of fitting *PaReNT* onto DAG-structured data, along with its architecture and training loop.

3.2. Model architecture

We use TensorFlow to build a custom architecture encoder-decoder recurrent neural model. It is equipped with a sequential decoder module (that we call the Retriever module) with cross-attention for parent retrieval, and a classification head with self-attention for word-formation classification. We use a multi-lingual subword embedding model provided by *BPEmb* (Heinzerling and Strube, 2017) to feed semantic information about the input lexemes into the model. The model architecture can be viewed in Figure 1.

The input lexeme is byte-pair-encoded into a sequence of subwords, and its respective language token is embedded into a two-dimensional space. Then, the subword sequence is fed into the Encoder module, where each subword is in parallel:

1. embedded into a 300-dimensional semantic space provided by *BPEmb*;
2. split into individual characters and fed into a time-distributed bi-directional LSTM layer with a dimensionality of 300.

Thus, a 300-dimensional dual representation, one semantic and one character-based, is obtained for each word. These representations are then concatenated and fed into a bi-directional LSTM layer, the result of which is the output of the Encoder module. A stack of so-called Transformer-like blocks follows. This construct is similar to the familiar Transformer Multi-Head Attention Block, except it runs in a time-distributed manner, as opposed to its parallel-running Transformer counterpart. The number of stacked Transformer-like blocks was one of the hyperparameters that was tuned during the training process. The model then branches off into the Classifier module and the Retriever module.

In the Classifier module, self-attention is calculated on the Encoder output, and the sequence dimension is globally averaged over. Then, the result is passed through a fully-connected layer and passed through a three-unit Softmax layer.

In the Retriever module, the Encoder output is used to recursively generate the parent of the given lexeme grapheme by grapheme in the way that is described by Vaswani et al. (2017). First, the input is fed into a self-attention block, which calculates attention between each pair of items from the input sequence. The attention is then added to the original input sequence. Next, alongside a skip connection, it is passed through a time-distributed fully-connected layer and added back. Finally, it is passed through a layer normalization.

	PaReNT			Most-Frequent			ChatGPT		
Lang	**Retr acc**	**Class acc**	**Cl bal acc**	**Retr acc**	**Class acc**	**Cl bal acc**	**Retr acc**	**Class acc**	**Cl bal acc**
Czech	0.64 (0.75)	0.96	0.66	0.05 (N/A)	0.94	0.33	0.39 (0.66)	0.87	0.36
German	0.60	0.95	0.86	0.06	0.81	0.33	0.28	0.57	0.71
English	0.69	0.86	0.84	0.25	0.49	0.33	0.33	0.39	0.38
Spanish	0.75	0.98	0.74	0.18	0.82	0.33	0.3	0.64	0.64
French	0.50	0.94	0.54	0.08	0.91	0.33	0.4	0.53	0.31
Dutch	0.55	0.89	0.80	0.09	0.70	0.33	0.11	0.60	0.69
Russian	0.64	0.97	0.72	0.13	0.86	0.33	0.25	0.74	0.63
Mean	0.62	0.94	0.74	0.12	0.79	0.33	0.29	0.62	0.53

Table 3: The performance of *PaReNT* and baselines for each language. The dummy balanced accuracy in classification is 0.33 for each language, because there are three word-formation categories. Most-Frequent accuracy for retrieval is the same as the proportion of unmotivated words in the given language's test set. The figure in (*parentheses*) on the second line indicates Family accuracy, which describes how many times the system correctly identified the Czech word-formation family of the correct parent(s). It is not listed for the Most-Frequent model, because it always returns the word unchanged, and a word is trivially part of its own word family in 100% of cases.

4. Results

For the evaluation of *PaReNT*'s performance in parent retrieval, we use Accuracy, which we define as the number of times *PaReNT* returned parents exactly string-equivalent (including capitalization) to the label parents in the correct ordering divided by the number of items in the test set. To evaluate word-formation classification, we used Accuracy, defined as the percentage of class hits divided by the number of items in the test set. Because the datasets we used are generally imbalanced in terms of word-formation class, we additionally used Balanced Accuracy from *scikit-learn*, which is defined as $(Specificity + Sensitivity)/2$. As an auxiliary metric, we use Family accuracy, which describes the proportion of cases where *PaReNT*'s retrieval output shares its word-formation family with the label. We used this metric only on Czech, since DeriNet 2.1 is the only resource at our disposal with the required structure and completeness.

The best model uses two bidirectional layers of $2,048$ units in the encoder, and a single Transformer-like block with two attention heads of dimensionality 512. In the retriever, Luong attention is used for Cross-Attention and one unidirectional layer of $2,048$ units to decode the output. In the classification head, Luong self-attention is used, with 512 units and a dropout of 0.3 in the final fully-connected layer. It was trained for 13 epochs, with a recurrent dropout of 0.2 in all recurrent layers and a regular dropout 0.5 of in all fully connected layers. The optimizer we used was ADAM, and we used a cyclical learning rate schedule (Smith, 2017) with a starting value of 10^{-4} and a final value of 10^{-5}. Its performance, broken down by language, can be found in Table 3.

PaReNT is directly compared against two baselines. The Most-Frequent baseline performs parent retrieval by returning the input unchanged, and always guesses *Unmotivated* as the category. The other baseline is ChatGPT (OpenAI, 2021), which is given the following prompt:

> Perform parent retrieval (predict which word or words the input lemma is motivated by.) and word-formation classification (predict whether the input lemma is a compound, a derivative, or unmotivated) on the given words. For each word, you will also be given its language of origin as a language token {cs : Czech, ru : Russian, de : German, es : Spanish, fr : French, nl : Dutch, en : English}. Format the output as tsv.
>
> The words:
>
> <list of words>

ChatGPT formats the output differently on each query, or sometimes even misunderstands the task or outright refuses to perform it, so its output has to be manually checked, regenerated if needed, and then reformatted. As a result, evaluation of ChatGPT is performed on a small sample of $n = 300$ words. These were fed into ChatGPT in increments of 100 words, prepended by the prompt each time.

5. Discussion

Comparing Table 2 and Table 3, it seems that the performance of the model for a given language not only depends on the amount of data available, but also on the morphological complexity of the language. For instance, despite the sparsity of the data available for Spanish, the model achieves high Accuracy classifying its lexemes. In Czech, the situation is the opposite – the amount of data available is large, but the performance is lower.

5.1. Error analysis

In this section, we present a linguist-performed analysis and interpretation of the errors made by

PaReNT, in the hopes of not only shedding light on the tool's functioning, but also on the word-formation systems of the languages in question. The errors have been analyzed by a human expert on a 1% random sample of the $179,720$ item validation dataset.

5.1.1. Type 1: Data conflict

The output of the model is correct, but conflicts with the label in the data.

In the data, each lexeme is assigned a single set of parents. Language reality is however often ambiguous, and the parents of a lexeme can be assigned in different ways. As a result, the model sometimes returns a lexeme that is correct, but disagrees with the label in the data, like in the Spanish example 9, where the expected output is *alcoholización* "alcoholization". Additionally, the datasets we used sometimes contain typos or other errors. This was in fact the case in the English example 10, where the listed output is in fact correct – and the item is wrongly listed as *Unmotivated*.

(9) Spanish

desalcoholización → ***desalcoholizar***
dealcoholization.N to dealcoholize.V

(10) English

north-northeast → ***north north east***
N N N N

5.1.2. Type 2: Inflectional confusion

The model mishandles the behavior of inflectional morphemes in word formation.

The role of inflection in word formation, typically compounding, has been touched upon in Section 2. The example *womenfolk* was used to illustrate that sometimes inflected words enter a word-formation process. When *womenfolk* specifically is fed into the model, it fails to return *woman* and instead returns *women*. It also occasionally generates an inflectional ending in a context where it looks like it could have been dropped (Spanish example 11, where *PaReNT* attaches a verbal ending to the expected *poem* "poem").

(11) Spanish

poema → ****poemar***
poem.N NONSENSE

5.1.3. Type 3: Morphophonemic ambiguity

The model fails to compensate for a difficult-to-account-for morphophonemic process.

The bulk of the model's predictions are based on the reverse application of word-formation rules. For instance, the model notices that there exists a pattern in English ⟨*root*⟩+*ment*, so upon seeing *development*, it returns *develop*. The problem is that it may be unclear how the rule should be retroactively applied. As an example, in Czech, the addition of a a suffix can induce stem allomorphy, resulting in /s/ → /š/. The application of the same suffix on another word, however, can yield /š/ → /š/, so when analyzing a word of the pattern **s* + ⟨*suffix*⟩, the model has to guess whether to generate /s/ or /š/. In the Czech example 12, the expected result is *rychlokvasit* "to ferment quickly".

(12) Czech

rychlokvašený → ***rychlokvašit***
quickly fermented.A NONSENSE

5.1.4. Type 4: Neural hallucination

The model baselessly hallucinates non-existent structures.

Sometimes, the model for unclear reasons simply switches, skips, replaces, overgenerates a character or split, or does something else that is difficult to interpret. Occasionally, it even hallucinates an entire morpheme, like in the Czech example 13, where an adjectival suffix and ending is added to the first nonsensical parent, or in the Dutch example 14, where an infinitive ending is added to both parents.

(13) Czech

přezůvka → ****přazový*** ****lažba***
slipper.N NONSENSE NONSENSE

(14) Dutch

hoegrootheid → ****hoegen*** ***roten***
amount.N NONSENSE to rot.V

5.1.5. Type 5: Overretrieval

The model does not return the parent of the input, but the parent of the parent.

In example 15, we would expect *PaReNT* to output множитель "factor" – the parent of which is the actually received output *множить* "to multiply". Similarly, we would expect the verb *mouler* "to mold" as an intermediate step in example 16. This error often overlaps with **Type 1: Data conflict**, typically in cases where the model interprets a compound as parasynthetic when such an interpretation is unnecessary. This is the case in example 17, where *Wechsel* "change" would be a simpler interpretation. The consistency of this error's appearance is reflected by the fact that Family accuracy is considerably higher than raw Accuracy, as shown in Table 3.

(15) Russian

множительный → ***множить***
multiplicable.A multiply.V

(16) French

moulerie → ***moule***
molding.N mold.N

(17) German

Tempowechsel → ***Tempo wechseln***
change of tempo.N tempo.N to change.V

5.1.6. Type 6: False morphemehood

The model misjudges the presence of a morpheme, typically an interfix. In example 18, *PaReNT* mistakes part of the stem of the verb *wandern* "to hike; to wander" for an interface, drops it, andends up with the completely unrelated word *Wand* "wall".

(18) German

Wandermöglichkeit → ***Wand Möglichkeit***
hiking opportunity.N wall.N opportunity

5.1.7. Type 7: Semantic irrelevance

The model retrieves a word in a formally correct manner, but in a way that no human would find intuitive or meaningful.

The problem in example 19 is that while the compound is split on a morphological boundary, the given compound is not a compound of three words, but rather a recursive compound. A human knows that, because the first real parent *Wochenende* is a common word meaning "weekend", but the model has no way of inferring this from its training data.

(19) German

Oktoberwochenende → ***Oktoberwochen ende***
October weekend.N October weeks.N end.N

5.1.8. Type 8: Missplitting

A non-compound is split (example 20), a compound is left unsplit (example 21), or a compound is split outside of a morphological boundary (example 22).

(20) English

compartmental → ****compart mental***
N NONSENSE A

(21) English

bullring → ***bullring***
N N

(22) English

raceabout → ****racea bout***
N NONSENSE N

As a final note, it needs to be stressed that certain word-formation phenomena are easier for tools like }textitPaReNT to model than others. For instance, the highly productive and common English derivational pattern *development – development*, *enjoy – enjoyment* is generally easily reconstructed by simply dropping the *-ment* suffix, while the unproductive and much rarer *broad – breadth*, *deep – depth* pattern is much harder, since there exist fewer examples and the sound changes are much less predictable. In other words, In the future, we would therefore like to develop an evaluation methodology that takes the complexity and frequency of certain word-formation processes into account.

5.2. Comparison with other tools

Not only does *PaReNT* outperform ChatGPT by a considerable margin in both tasks for every language (cf. Table 3), but also performs the tasks much more consistently. ChatGPT formats the output slightly differently each time, and also sometimes refuses to perform the task in the first place. This makes *PaReNT* much more suitable for pipelining in downstream applications.

We also compare *PaReNT* to *Word-Formation Analyzer for Czech* (*WFA.ces*; Svoboda and Ševčíková 2021). *PaReNT*'s performance in parent retrieval of Czech words at 0.64 is slightly lower than that of *WFA.ces* at 0.67. We attribute this to our splitting the data set by lexicographical block (Section 3), which was not used in the case of *WFA.ces*.

We additionally attempt to compare *PaReNT*'s performance in splitting German compounds with Krotova et al.'s deep splitter, which was trained and evaluated solely on GermaNet. At face value, it achieves an Accuracy of 0.95. We subset our validation dataset so that only compounds from GermaNet are left, and we measure Accuracy on the subset and arrive at a markedly worse 0.69. However, we note that Krotova et al.'s tool is a split-point splitter, and is evaluated as such[2]; and also, *PaReNT* was trained not only on GermaNet, but also on CELEX, which has somewhat different annotation conventions. We use two different adjustments to try and take these differences into account. First, we adjust the performance of *PaReNT* to closer match the evaluation conventions of Krotova et al., as outlined in the Error Analysis section of their paper. We hand-annotate a 10% sample of the errors in the validation set in accordance with the classification set forth in Section 5.1, and only consider errors of type **3**, **4**, **6**, **7**, and **8**, which amount to 51% of all the errors. After this adjustment, *PaReNT*'s Accuracy climbs to 0.84 compared to the 0.95 of Krotova et al.. Second, we adjust the Accuracy of Krotova et al. to closer match our evaluation conventions, one of

[2]Interfixes are left attached to the left-hand parent.

which is that all predicted parents must be valid words in their lemma form. However, only 60% of compounds in GermaNet are formed by a simple concatenation of their parent lemmas. As a result, the rest cannot be handled by a split-point splitter according to the aforementionded criterion[3]. We therefore consider the 60% to be the oracle score (maximum attainable value) for the split-point splitter, and arrive at 0.57 Accuracy for Krotova et al.'s splitter compared to *PaReNT*'s 0.69.

The deep-learning nature of *PaReNT* makes its usage somewhat computationally intensive. It does not however require a GPU to be practical, since the model processes about 20 lexemes per second on a CPU with batch inference, and takes up about 400 MB of disk space. While this does make it more cumbersome than e.g. a Snowball-based stemmer or a similar rule-based tool, it can still be comfortably used on a consumer-grade computer.

6. Conclusion

In this paper, we presented *PaReNT* (Parent Retrieval Neural Tool), a deep learning-based tool for the parent retrieval and word-formation classification of isolated lexemes in their conventional dictionary form for eight different languages. It is based on the Tensorflow framework and runs on a dual-representation input, encoder/decoder with a classification module RNN neural network. *PaReNT* has achieved a total Accuracy of 0.62 in parent retrieval and a Balanced Accuracy of 0.74 in word-formation classification on an independent validation data set. In the future, we would like to extend *PaReNT*'s language set into other languages. Additionally, we observe that the standard dictionary form (lemma) of a given lexeme in a language is not necessarily the most relevant or informative from a word-formation perspective (as evidenced by e.g. the inclusion of an inflected word form in the word *womenfolk*), we would also like to include inflected word forms in the training data.

We also consider the ternary classification framing of the word-formation classification problem to be untenable in the long-term, simply because it is obvious that the reality of word formation is much more complex than the three categories of compound, derivative, unmotivated. In the future, we would therefore like to include more categories such as clipping or univerbization.

[3]Unless it has explicit interfix handling, which the authors do not mention and their Error analysis section seems to indicate otherwise. The reason is explained and exemplified in Section 2.3.

Download and use *PaReNT*

A public version of *PaReNT*, alongside the part of the training data crawled from Wiktionary, can be found on GitHub: `https://github.com/iml-r/PaReNT`.

Acknowledgments

The study was supported by the Charles University, project GA UK No. 128122, and by the Ministry of Education, Youth and Sports of the Czech Republic, Project No. LM2023062 LINDAT/CLARIAH-CZ.

7. Bibliographical References

Hamood Alshalabi, Sabrina Tiun, Nazlia Omar, Fatima N AL-Aswadi, and Kamal Ali Alezabi. 2022. Arabic light-based stemmer using new rules. *Journal of King Saud University-Computer and Information Sciences*, 34(9):6635–6642.

Khuyagbaatar Batsuren, Gábor Bella, Aryaman Arora, Viktor Martinovic, Kyle Gorman, Zdeněk Žabokrtský, Amarsanaa Ganbold, Šárka Dohnalová, Magda Ševčíková, Kateřina Pelegrinová, Fausto Giunchiglia, Ryan Cotterell, and Ekaterina Vylomova. 2022. The SIGMORPHON 2022 shared task on morpheme segmentation. In *Proceedings of the 19th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 103–116, Seattle, Washington. Association for Computational Linguistics.

Elizaveta L. Clouet and Béatrice Daille. 2014. Splitting of compound terms in non-prototypical compounding languages. In *Workshop on Computational Approaches to Compound Analysis*, pages 11–19.

Ryan Cotterell, Arun Kumar, and Hinrich Schütze. 2016. Morphological segmentation inside-out. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2325–2330, Austin, Texas. Association for Computational Linguistics.

Mathias Creutz and Krista Lagus. 2002. Unsupervised Discovery of Morphemes. *arXiv preprint cs/0205057*.

Fabienne Fritzinger and Alexander Fraser. 2010. How to avoid burning ducks: Combining linguistic analysis and corpus statistics for German compound processing. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 224–234, Uppsala, Sweden. Association for Computational Linguistics.

Stig-Arne Grönroos, Sami Virpioja, and Mikko Kurimo. 2018. Cognate-aware morphological segmentation for multilingual neural translation. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 386–393, Belgium, Brussels. Association for Computational Linguistics.

Martin Haspelmath. 2017. The indeterminacy of word segmentation and the nature of morphology and syntax. *Folia linguistica*, 51:31–80.

Benjamin Heinzerling and Michael Strube. 2017. BPEmb: Tokenization-free pre-trained subword embeddings in 275 languages. *arXiv preprint arXiv:1710.02187*.

Verena Henrich and Erhard Hinrichs. 2011. Determining immediate constituents of compounds in GermaNet. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing 2011*, pages 420–426.

Sanjeet Khaitan, Arumay Das, Sandeep Gain, and Adithi Sampath. 2009. Data-Driven Compound Splitting Method for English Compounds in Domain Names. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, page 207–214, New York, NY, USA. Association for Computing Machinery.

Oskar Kohonen, Sami Virpioja, and Krista Lagus. 2010. Semi-supervised learning of concatenative morphology. In *Proceedings of the 11th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology*, pages 78–86.

Irina Krotova, Sergey Aksenov, and Ekaterina Artemova. 2020. A joint approach to compound splitting and idiomatic compound detection. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4410–4417, Marseille, France. European Language Resources Association.

Hugo Mailhot, Maximiliano A Wilson, Joël Macoir, S Hélène Deacon, and Claudia Sánchez-Gutiérrez. 2020. MorphoLex-FR: A derivational morphological database for 38,840 French words. *Behavior Research Methods*, 52(3):1008–1025.

Fiammetta Namer. 2003. Automatiser l'analyse morpho-sémantique non affixale: le système Dérif. *Cahiers de grammaire*, 28:31–48.

OpenAI. 2021. ChatGPT.

Martin F Porter. 1980. An algorithm for suffix stripping. *Program: electronic library and information systems*, 14:130–137.

Martin F. Porter. 2001. Snowball: A language for stemming algorithms. Published online. Accessed 21.01.2022, 15.00h.

Martin Riedl and Chris Biemann. 2016. Unsupervised compound splitting with distributional semantics rivals supervised methods. In *Proceedings of The 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologie*, pages 617–622, San Diego, CA, USA.

Peter Smit, Sami Virpioja, Stig-Arne Grönroos, Mikko Kurimo, et al. 2014. Morfessor 2.0: Toolkit for statistical morphological segmentation. In *The 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL), Gothenburg, Sweden, April 26-30, 2014*. Aalto University.

Leslie N Smith. 2017. Cyclical learning rates for training neural networks. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pages 464–472. IEEE.

Emil Svoboda and Magda Ševčíková. 2021. Splitting and identifying Czech compounds: A pilot study. In *Proceedings of the Third International Workshop on Resources and Tools for Derivational Morphology (DeriMo 2021)*, pages 129–138.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Sami Virpioja, Peter Smit, Stig-Arne Grönroos, Mikko Kurimo, et al. 2013. Morfessor 2.0: Python implementation and extensions for Morfessor baseline.

8. Language Resource References

Baayen, Harald and Piepenbrock, Richard and Gulikers, Leon. 2014. *CELEX2 LDC96L14, 1995*. PID https://catalog.ldc.upenn.edu/LDC96L14.

Batsuren, Khuyagbaatar and Goldman, Omer and Khalifa, Salam and Habash, Nizar and Kieraś, Witold and Bella, Gábor and Leonard, Brian and Nicolai, Garrett and Gorman, Kyle and Ate, Yustinus Ghanggo and Ryskina, Maria and Mielke, Sabrina and Budianskaya, Elena and El-Khaissi, Charbel and Pimentel, Tiago and Gasser, Michael and Lane, William Abbott and Raj, Mohit and Coler, Matt and Samame, Jaime Rafael Montoya and Camaiteri, Delio Siticonatzi and Rojas, Esaú Zumaeta and López Francis,

Didier and Oncevay, Arturo and López Bautista, Juan and Villegas, Gema Celeste Silva and Hennigen, Lucas Torroba and Ek, Adam and Guriel, David and Dirix, Peter and Bernardy, Jean-Philippe and Scherbakov, Andrey and Bayyr-ool, Aziyana and Anastasopoulos, Antonios and Zariquiey, Roberto and Sheifer, Karina and Ganieva, Sofya and Cruz, Hilaria and Karahóǧa, Ritván and Markantonatou, Stella and Pavlidis, George and Plugaryov, Matvey and Klyachko, Elena and Salehi, Ali and Angulo, Candy and Baxi, Jatayu and Krizhanovsky, Andrew and Krizhanovskaya, Natalia and Salesky, Elizabeth and Vania, Clara and Ivanova, Sardana and White, Jennifer and Maudslay, Rowan Hall and Valvoda, Josef and Zmigrod, Ran and Czarnowska, Paula and Nikkarinen, Irene and Salchak, Aelita and Bhatt, Brijesh and Straughn, Christopher and Liu, Zoey and Washington, Jonathan North and Pinter, Yuval and Ataman, Duygu and Wolinski, Marcin and Suhardijanto, Totok and Yablonskaya, Anna and Stoehr, Niklas and Dolatian, Hossep and Nuriah, Zahroh and Ratan, Shyam and Tyers, Francis M. and Ponti, Edoardo M. and Aiton, Grant and Arora, Aryaman and Hatcher, Richard J. and Kumar, Ritesh and Young, Jeremiah and Rodionova, Daria and Yemelina, Anastasia and Andrushko, Taras and Marchenko, Igor and Mashkovtseva, Polina and Serova, Alexandra and Prud'hommeaux, Emily and Nepomniashchaya, Maria and Giunchiglia, Fausto and Chodroff, Eleanor and Hulden, Mans and Silfverberg, Miikka and McCarthy, Arya D. and Yarowsky, David and Cotterell, Ryan and Tsarfaty, Reut and Vylomova, Ekaterina. 2022. *UniMorph 4.0: Universal Morphology*. European Language Resources Association. PID https://unimorph.github.io/.

Henrich, Verena and Hinrichs, Erhard. 2010. *GernEdiT-the GermaNet editing tool*. PID https://uni-tuebingen.de/fakultaeten/philosophische-fakultaet/fachbereiche/neuphilologie/seminar-fuer-sprachwissenschaft/arbeitsbereiche/allg-sprachwissenschaft-computerlinguistik/ressourcen/lexica/germanet-1/.

Kyjánek, Lukáš and Žabokrtský, Zdeněk and Vidra, Jonáš and Ševčíková, Magda. 2021. *Universal Derivations v1.1*. PID http://hdl.handle.net/11234/1-3247. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Mailhot, Hugo and Wilson, Maximiliano A and Macoir, Joël and Deacon, S Hélène and Sánchez-Gutiérrez, Claudia. 2020. *MorphoLex-FR: A derivational morphological database for 38,840 French words*. Springer. PID http://hdl.handle.net/11234/1-3247.

Vidra, Jonáš and Žabokrtský, Zdeněk and Kyjánek, Lukáš and Ševčíková, Magda and Dohnalová, Šárka and Svoboda, Emil and Bodnár, Jan. 2021. *DeriNet 2.1*. PID http://hdl.handle.net/11234/1-3765. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Vodolazsky, Daniil and Petrov, Hermann. 2021. *Compound Splitting and Analysis for Russian*. PID http://hdl.handle.net/11234/1-3247.

Žabokrtský, Zdeněk and Bafna, Niyati and Bodnár, Jan and Kyjánek, Lukáš and Svoboda, Emil and Ševčíková, Magda and Vidra, Jonáš. 2022. *Towards Universal Segmentations: UniSegments 1.0*. European Language Resources Association. PID http://hdl.handle.net/11234/1-4629.

Zeller, Britta and Padó, Sebastian and Šnajder, Jan. 2014. *Towards Semantic Validation of a Derivational Lexicon*. Dublin City University and Association for Computational Linguistics. PID http://hdl.handle.net/11234/1-3247.