Open Library of Humanities

Part of the Ubiquity Partner Network

Research

How to Cite: Haverals, Wouter, Folgert Karsdorp, and Mike Kestemont. 2019. "Data-Driven Syllabification for Middle Dutch." *Digital Medievalist* 12(1): 2, pp. 1–23. DOI: https://doi.org/10.16995/dm.83

Published: 04 November 2019

Peer Review:

This is a peer-reviewed article in *Digital Medievalist*, a journal published by the Open Library of Humanities.

Copyright:

Open Access:

Digital Medievalist is a peer-reviewed open access journal.

Digital Preservation:

The Open Library of Humanities and all its journals are digitally preserved in the CLOCKSS scholarly archive service.

Haverals, Wouter, et al. 2019. "Data-Driven Syllabification for Middle Dutch." *Digital Medievalist* 12(1): 2, pp. 1–23. DOI: https://doi.org/10.16995/dm.83

RESEARCH

Data-Driven Syllabification for Middle Dutch

Wouter Haverals[1], Folgert Karsdorp[2] and Mike Kestemont[1]

[1] University of Antwerp, BE

[2] Meertens Institute – KNAW, NL

Corresponding author: Wouter Haverals (wouter.haverals@uantwerp.be)

The task of automatically separating Middle Dutch words into syllables is a challenging one. A first method was presented by Bouma and Hermans (2012), who combined a rule-based finite-state component with data-driven error correction. Achieving an average word accuracy of 96.5%, their system surely is a satisfactory one, although it leaves room for improvement. Generally speaking, rule-based methods are less attractive for dealing with a medieval language like Middle Dutch, where not only each dialect has its own spelling preferences, but where there is also much idiosyncratic variation among scribes. This paper presents a different method for the task of automatically syllabifying Middle Dutch words, which does not rely on a set of pre-defined linguistic information. Using a Recurrent Neural Network (RNN) with Long-Short-Term Memory cells (LSTM), we obtain a system which outperforms the rule-based method both in robustness and in effort.

Keywords: automatic syllabification; data-driven methods; recurrent neural network; Middle Dutch; orthographic variation

1 Introduction

§**1** The main aim of this study is to develop a tool for automatically syllabifying Middle Dutch words. It goes without saying that the best way to go about this task would be through a simple look-up query in a dictionary, where words are stored alongside their syllabified versions. This method, however, is unattainable for Middle Dutch because of mainly two reasons:

1. Although a dictionary for Middle Dutch does exist (Verdam and Verwijs 1885), it lacks information about syllable boundaries.

2. More importantly, what we today call "Middle Dutch" is a container concept, used for the various dialects spoken in the Low Countries (today: Flanders and the Netherlands) between ca. 1150 and 1500. Since there is no standardized spelling yet in this period, the same word can be spelled in many different ways, depending on where or when a text was written. Orthographic variation may even be present in the same text, written by one and the same scribe. For instance, the Middle Dutch word for "damsel" has the following – and more – spelling variants: *joncfrouwe*, *joncvrauwe*, *joncvrouwe*, *joncvrovwe*, *jonvrowe*, *ioncfrouwe*, *ionfrouwe*, *ioffrouwe*, etc. (the extensive orthographic in Middle Dutch is also the subject of a paper by Van Halteren and Rem (2013), who noted that the lemma *gelijk* ("similarly") has 24 different word forms in the Corpus Van Reenen-Mulder).

Since there is no list available with all the different spelling variants of every Middle Dutch word, and since the existing dictionary does not contain syllabified versions of lemmas, one would like an automatic system that is able to correctly determine syllable boundaries, while dealing with this multitude of spelling variation in a flexible way. To achieve this, we propose a syllabification method that takes a pre-annotated list of syllabified Middle Dutch words as input for an RNN-tagger.

2 Rules for syllabification of Modern Dutch

§**2** Before we discuss the results of the Middle Dutch syllabifier, it is important to gain insight into the rules that form the basis of correct syllabification for Dutch in general, and for Middle Dutch in particular. Syllable structure in Dutch has been the subject of various studies (Vennemann 1988; Booij 1999; Trommelen 2011). As it is beyond the scope of this paper to provide an exhaustive explanatory model, we will set out the general rules and principles that were followed when annotating the training data.

§**3** To give an example of the task at hand, consider the Dutch word *kerstavonden* (IPA: [kɛrstaːvɔndən]; English translation: "Christmas Eves"). On a naive and unsubstantiated basis, we can propose a couple of different syllabifications, presented in **Table 1**. On an intuitive basis, however, we get a feeling that some of these candidates are less likely than others. But then what are the requirements for

Table 1: Candidates for the syllabification of "kerstavonden".

ke-rsta-von-den	*ker-sta-von-den*	*kerst-a-vond-en*
ke-rstav-ond-en	*kers-ta-vond-en*	*kerst-a-von-den*
ker-sta-vond-en	*kers-tav-on-den*	*kerst-av-on-den*

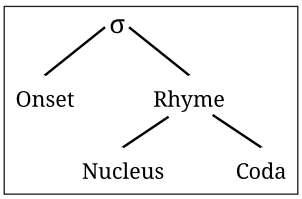

Figure 1: Syllable template.

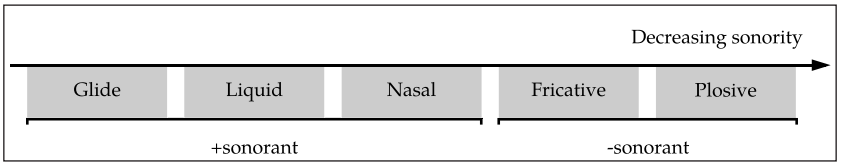

Figure 2: Sonority Ranking Hierarchy.

correct syllabification? Part of the solution lies in breaking down the question to 'what can be a possible syllable in Dutch?'

§**4** In any language, a syllable consists of up to three elements: an onset, a nucleus, and a coda. The nucleus is indispensable, and is – at least in modern Dutch – always a vowel or a diphthong. Onset and coda, both optional, are the collections of consonants that respectively precede or follow the nucleus. Nucleus and coda combined form the syllable's rhyme. Altogether, the internal structure of a syllable is represented as in **Figure 1** (with a lower-case sigma [σ] as the standard symbol for a syllable in phonology studies).

§**5** In order to obtain the correct syllabification of a Dutch word, we have to take into account a couple of principles and constraints that are imposed onto this template:

1. **Sonority Ranking Hierarchy** – The main constraint relates to the contents of onset and coda. The order of consonants in both clusters is determined by the Sonority Ranking Hierarchy (**Figure 2**). The sonority

of consonants has to decrease towards the outer edges of a well-formed syllable. Subsequently, a mirror effect can be perceived: the onset of the syllable will have the least sonorous consonants placed at the beginning, whereas in the coda, they have to be placed near the end (Selkirk 1982; Kager and Zonneveld 1986). With regard to the example *kerstavonden*, we can dismiss the proposed syllabification $(ke)_{\sigma}$ $(rsta)_{\sigma}$ $(von)_{\sigma}$ $(den)_{\sigma}$ and $(ke)_{\sigma}$ $(rstav)_{\sigma}$ $(on)_{\sigma}$ $(den)_{\sigma}$, since the consonant cluster /rst/ in the onset of the second syllable violates the Sonority Ranking Hierarchy.[1]

2. **Maximum Onset Principle** – Although in compliance with the Sonority Ranking Hierarchy, the proposed syllabification $(kers)_{\sigma}$ $(tav)_{\sigma}$ $(on)_{\sigma}$ $(den)_{\sigma}$ is incorrect. Like many other Germanic languages, Dutch syllabification adheres to the Maximum Onset Principle, which states that consonants are maximally assigned to the onset of a syllable as long as it is permitted by the Sonority Ranking Hierarchy (Selkirk 1982; Trommelen 2011). In the aforementioned example, the coda /v/ of the second syllable has to be appended to the onset of the next syllable, in compliance with the Maximum Onset Principle.
3. **Morpheme boundaries** – In our reasoning thus far, syllable boundaries are determined on the basis of the phonological properties of words. Both the Sonority Ranking Hierarchy and the Maximum Onset Principle are motivated by the (im)possibility to pronounce particular clusters of consonants. Superimposed on these prosodic constraints, however, is a principle that pertains to the domain of morphology, namely: within a prosodic word, morphemes are independent domains of syllabification (Booij 1999, 29–30). Or, in other words, syllable boundaries tend to follow morphological boundaries. Since *kerstavonden* is a compound

[1] Without going too deep into the matter, it should be noted that there are some exceptions to the Sonority Ranking Hierarchy. For Germanic languages, the most prominent one is the possibility of /s/ functioning as a sort of appendix, which, in some cases, can be added to the beginning of an onset, or to the end of a coda. This results in the possibility of having e.g. /str/, /skr/ and /spl/ as legitimate onsets in Dutch, and /lps/, /rks/, /rts/ as well-formed codas (Booij 1999, 26–29).

that consists of *kerst+avonden*, we can conclude that also the previously mentioned syllabification $(\text{kers})_\sigma (\text{ta})_\sigma (\text{von})_\sigma (\text{den})_\sigma$ is faulty. As a result, the only possible syllabification of the word *kerstavonden* is $(\text{kerst})_\sigma (\text{a})_\sigma (\text{von})_\sigma (\text{den})_\sigma$.

In conclusion, correct syllabification comes down to ticking boxes. Firstly, when syllable boundaries surpass morpheme boundaries, one can immediately reject the proposed syllabification. Secondly, consonants are maximally assigned to the onset, while – thirdly – respecting the Sonority Ranking Hierarchy. For the proposed syllabifications of *kerstavonden*, ticking these boxes produces a set of possibilities as in **Table 2**. The third column in this table indicates how well the onset is saturated according to the Maximum Onset Principle on a scale of one to three dots. For each consonant that could have been added to the onset but was not, one dot is removed.

3 Syllabification of Middle Dutch

§**6** Syllabification for Middle Dutch follows the same general principles as the ones outlined above for Modern Dutch. A challenge, however, is posed by the historic language's orthography. With respect to the task of syllabification, it is especially important that our model is able to distinguish consonants from vowels. However, in Middle Dutch the same grapheme can sometimes have both those sound values.

Table 2: Syllabification possibilities for the word "kerstavonden" evaluated.

Proposed syllabification	Sonority Ranking Hierarchy	Maximum Onset Principle	Morpheme Boundaries
ke-rsta-von-den	✗	•••	✗
ke-rstav-ond-en	✗	•	✗
ker-sta-von-den	✓	•••	✗
ker-stav-on-den	✓	••	✗
kers-tav-on-den	✓	•	✗
kers-ta-vond-en	✓	•	✗
kerst-av-on-den	✓	••	✓
kerst-a-vond-en	✓	••	✓
kerst-a-von-den	✓	•••	✓

For instance, the letters ⟨u⟩ and ⟨v⟩ are often used interchangeably. In a word like *huse/hvse* ("house") they both have the sound value of the vowel /y/, yet in the word *over/ouer* ("about") they should be pronounced as a consonant /v/. Sometimes even, one word can have both phonetic realizations: in *geualueert* ("evaluated"), the first ⟨u⟩ represents a consonant and the second one a vowel. When we will be evaluating the syllabifier in the end, we must pay special attention to cases like this where there is an increased risk of incorrectly appending a grapheme to the onset, coda or nucleus. **Table 3** provides an oversight of such "graphemic pitfalls" for Middle Dutch syllabification.

§**7** Dealing with the orthographic variation also means making decisions – some more arbitrary than others – with regard to what will be considered as a correct syllabification of a Middle Dutch word. An example of such a decision can be illustrated by our dealing with the graphemic cluster ⟨ie⟩. It is generally assumed that Middle Dutch script ⟨ie⟩ was pronounced as monophthongal /iː/. In

Table 3: Graphemic ambiguity in Middle Dutch. The words presented in this table are all part of the Corpus Van Reenen-Mulder, used in the syllabification experiment.

Phoneme	Graphemes	Syllabified examples
/y/	⟨u⟩, ⟨v⟩	*h**u**n-dert*, *h**v**n-dert* *ghe-des-tr**u**-eert*, *ghe-des-tr**v**-eert* ***u**-we*, ***v**-we*
/v/	⟨u⟩, ⟨v⟩	*o-**u**er*, *o-**v**er* *ad-**u**o-ca-te*, *ad-**v**o-ca-te* *he-mel-**u**aert*, *he-mel-**v**aert*
/i/	⟨i⟩, ⟨ie⟩, ⟨j⟩, ⟨y⟩, ⟨ij⟩	*v**i**-an-de*, *v**y**-an-de* *pr**j**n-ci-pael*, *pr**i**n-ci-pael* *pro-ch**i***, *pro-ch**ie***, *pro-ch**ij***, *pro-ch**y***
/j/	⟨i⟩, ⟨j⟩, ⟨y⟩	***i**aer-lecs*, ***j**aer-lecs* ***i**onc-frou*, ***j**onc-frou* *za-**y**en*, *za-**i**en*
/uː/	⟨u⟩, ⟨uu⟩, ⟨v⟩, ⟨vv⟩, ⟨w⟩	*h**u**-se*, *h**uu**-se*, *h**vv**-se*, *h**w**-se* *s**uu**t-si-de*, *s**w**t-si-de* *erf-h**u**-re*, *erf-h**w**-re*
/w/	⟨uu⟩, ⟨vv⟩, ⟨w⟩	*on-be-d**uu**on-ghen*, *on-be-d**w**on-ghen* ***vv**e-gen*, ***w**e-gen* *op-**uu**art*, *op-**w**aert*

thirteenth-century Flanders, however, one could argue that, when this cluster occurs at the outer-right edge of a word, ⟨ie⟩ was pronounced as bisyllabic /i.jə/ (van Loey 1976). *Partie* ("part"), for instance, a word frequently used by Maerlant in his rhymed chronicle *Spiegel Historiael* (ca.1284–ca.1289) most likely had to be pronounced as /par.ti.jə/. This can be deduced from rhyme pairs such as *partie* : *lije*, where the grapheme ⟨j⟩ has the sound value of a consonant and thus marks a syllable boundary. In the same text, we also find rhyme combinations such as *philosophie* : *lije*, *paertije* : *normendie*, indicating that in Maerlant's *Spiegel Historiael* the syllabified versions of these words should therefore be: $(par)_{\sigma}$ $(ti)_{\sigma}$ $(e)_{\sigma}$, $(li)_{\sigma}$ $(je)_{\sigma}$, $(phi)_{\sigma}$ $(lo)_{\sigma}$ $(so)_{\sigma}$ $(phi)_{\sigma}$ $(e)_{\sigma}$, $(nor)_{\sigma}$ $(men)_{\sigma}$ $(di)_{\sigma}$ $(e)_{\sigma}$. Since these words never form rhyme pairs with monosyllabic words like *die*, *drie*, *zie* or *wie* in Maerlant's work, it reinforces our assumption of ⟨ie⟩ as bisyllabic /i.jə/. It has been argued, however, that during the fourteenth century word-final bisyllabic ⟨ie⟩ evolved into a single syllable (van Loey 1976, p. 54–55, p. 60). Here as well, arguments can be formed on the basis of rhyme combinations. In the rhymed chronicle *De Grimbergse oorlog* (ca.1350), *partie* frequently rhymes with monosyllabic *die*, *drie*, *sie*, *vrie*, *nie*. Although this does not undisputedly prove that from the fourteenth century onwards all word-final graphemic clusters ⟨ie⟩ in all dialect regions have to be considered as monosyllabic, there surely is a tendency towards this phonetic realization. Because of this, we follow the pragmatic convention to consider all word-final clusters ⟨ie⟩ to be monophthongal /i:/.

§**8** Another noteworthy decision that was made has to do with clitic forms and contractions, which are syllabified according to their phonetic realizations. For example, *sbisscops* (*des*+*bisscops*) is syllabified as *sbis-scops*, like one would also syllabify the English possessive *bishop's* as *bi-shops*. The same goes for enclitics such as *gaedi* (*gaet*+*di*) and *sidi* (*sijt*+*ghi*). Here, however, it becomes rather difficult to delineate the exact morpheme boundaries, since the grapheme ⟨d⟩ in both words is the result of an assimilation process. Because of this, we decided to place a syllable boundary in compliance to the Maximum Onset Principle: *gae-di*, *si-di*.

§**9** Finally, the three major principles for syllabification (as outlined in section 2) were adhered to when encountering words that contain a double consonant with the same sound value. For example, the modern-day Dutch word *achttien* ("eighteen")

occurs in our corpus of Middle Dutch words in various spellings. When it is spelled as *achtien* with one ⟨t⟩, that consonant is added to the second syllable. Thus, respecting the Maximum Onset Principle, we syllabify *ach-tien*. In the case of the spelling *achttien*, a syllable boundary occurs between ⟨tt⟩. In compliance with the Sonority Ranking Hierarchy, we therefore syllabify words like *achttien*, *ancker* and *avondde* as *acht-tien*, *anc-ker* and *a-vond-de*.

4 Previous research by Bouma and Hermans

§**10** A first system of automatic syllabification of Middle Dutch was developed by Bouma and Hermans (2012), whose approach comprises two stages: a finite-state transducer and data-driven error correction. The first stage heavily relies on definitions of grapheme sequences that constitute possible nuclei and onsets in Middle Dutch (the groundwork of the final-state transducer used by Bouma and Hermans for Middle Dutch has been laid by Bouma in 2003). Superimposed onto these definitions, Bouma and Hermans included various rules that are meant to deal with Middle Dutch's orthographic variation (as outlined in **Table 3**). An example of such a rule is the one that has to prevent ⟨u⟩ from being recognized as a vowel (/y/), when it is actually a consonant (/v/):

> In the sequences ⟨aue⟩, ⟨eue⟩, and ⟨oui⟩, ⟨u⟩ almost always functions as a ⟨v⟩. Therefore, we replace such sequences with ⟨aUe⟩, ⟨eUe⟩, and ⟨oUi⟩, respectively, where we use ⟨U⟩ as the character that denotes a ⟨u⟩ functioning as a consonant (Bouma and Hermans 2012, p. 33).

Although the premise of this rule is legitimate, a significant risk is lurking: the prospect of being incomplete. And indeed, the above-mentioned rule disregards many other possible grapheme clusters. To name a few: ⟨aui⟩, ⟨auo⟩, ⟨euo⟩, ⟨eui⟩, ⟨eua⟩, ⟨oui⟩ etc. are all environments where the character ⟨u⟩ can also function as a ⟨v⟩.

§**11** It is mainly because definitions and rules like the one above have been drafted without striving towards exhaustiveness that syllabification errors are still prevalent after completing the first stage of their experiment. An intermediate

evaluation of Bouma & Hermans' rule-based system, carried out on a sample of 20,139 Middle Dutch words, shows that 90.1% of words are correctly syllabified. The implication here is that nearly 10% needs to be corrected in the second stage, during manual error correction. The task of skimming through ca. 20,000 words that have been syllabified correctly in 90% of the cases, is one that puts a great deal of strain on an annotator. Upon inspection, 107 words were still incorrectly hyphenated (that is 0.53% out of the total number of 20,139 words) in Bouma and Hermans' manually corrected word list.

§**12** Finally, transformation-based learning, as developed for POS tagging by Brill (1995), was applied. Through this machine learning method, new rules were automatically deduced by comparing the output of the automatic finite-state transducer with the human-corrected word list. After implementing the highest scoring, new-found rules, Bouma and Hermans' syllabification model yields a hyphenation accuracy score of 97.9% and a word accuracy of 96.5%.[2]

5 Experiment and results

§**13** The Middle Dutch automatic syllabifier presented in this paper builds on the research carried out by Bouma and Hermans but moves away from manually specified rules: without explicitly hard-coding various rules, our model is able to apply syllabification rules and conventions solely from a supervised learning experience.

5.1 Data set

§**14** Like Bouma and Hermans' experiment, the data used as training material for our syllabification model stems from the *Corpus Van Reenen-Mulder* (CRM). This corpus, created by Van Reenen and Mulder (1993) at the Free University Amsterdam, contains approximately 2,700 charters, written in the Netherlands and Flanders between 1300 and 1400 (all charters are freely accessible in plain text format on the corpus page of *Diachronie.nl*, hosted by the Meertens Institute). Content-wise, most

[2] The *hyphenation accuracy* is the percentage of correctly inserted hyphens in total, whereas *word accuracy* is the percentage of correctly hyphenated words. E.g. *kerst-av-on-den* has 2/3 hyphens placed correctly. The hyphenation accuracy is thus 66%. However, since the full word is syllabified incorrectly, word accuracy is 0%.

charters deal with administrative law, disputes or regulations that are made between two or more parties. The medieval charter can therefore best be compared with the contemporary notarial deed.

§**15** The great benefit of working with the CRM lies in the widespread geographic distribution of the material. From the north of Holland to the south of Flanders, all major Middle Dutch dialectal varieties are represented. This geographic diversity thus guarantees linguistic diversity, which is especially useful when – in the end – we want our syllabifier to perform well across all variants of the Middle Dutch language.

§**16** Compared to the CRM word list used by Bouma and Hermans, the word list in our experiment was modified in three ways. **(1)** First and foremost, whereas Bouma and Hermans used only 20,139 words, alphabetically ranging from *a* to *kerstauonde*, we expanded the word list so that it includes all 43,710 unique words from the CRM corpus.[3] These words were all syllabified; i.e. hyphens were inserted in accordance with the guidelines outlined above. **(2)** Since the first half of our word list overlaps with the manually reviewed word list from in Bouma and Hermans' experiment, we assumed that this part of the list was without errors. However, as already mentioned, syllabification errors were still prevalent. The second modification made to the data was therefore: correction. In total, 107 erroneous syllabifications were rectified (for some examples, see **Table 4**). Additionally, some inconsistent syllabifications were amended. For example, in their corrected word list, Bouma and Hermans sometimes syllabified the word ending ⟨iaen⟩ as $(i)_{\sigma}$ $(aen)_{\sigma}$, yet at other times they left $(iaen)_{\sigma}$ as one syllable. Since we want a machine learning algorithm to learn from a list of syllabified words, this consistency is especially important. In total, 95 words from Bouma and Hermans' list were found to be inconsistently syllabified (for some examples, see **Table 5**). When manually reviewing the 43,710 syllabified words used in our experiment, we made sure that this consistency was continued throughout the entire data set. **(3)** Finally, several words were deleted from the list

[3] Some tokens in the CRM contain diacritic symbols to indicate abbreviations, clitic forms, or unclear parts in the original charter. Striving towards orderliness, such tokens were excluded when collecting the data.

Table 4: Examples of erroneous syllabifications present in the data used by Bouma and Hermans, along with their corrections. In total, 107 erroneous syllabifications were rectified.

Incorrect syllabification	Correct syllabification	English Translation
bau-in-cho-ue	*ba-uinc-ho-ue*	*Bavikhove* [town name]
be-r-ven	*ber-ven*	*righteous*
ceyn-su-ri	*ceyns-uri*	*duty-free*
dau-ijds	*da-uijds*	*David's*
die-se-lue	*die-sel-ue*	*the same*
dwer-se-rue	*dwers-er-ue*	*farmyard*
erd-u-ast	*erd-uast*	*attached to the land*
ghe-uu-oen-te	*ghe-uuoen-te*	*customary*
hen-rixe	*hen-ri-xe*	*Hendrik's*
iair-lixe	*iair-li-xe*	*yearly*
kers-au-on-de	*kers-a-uon-de*	*Christmas Eve*

Table 5: Examples of inconsistent syllabifications present in the data used by Bouma and Hermans, along with their corrections. In total, 95 inconsistencies were amended.

Inconsistent syllabification	Improved consistency	English Translation
*a-po-**st**el*	*a-po**s-t**el*	*apostle*
*ap-po**s-t**el*	*ap-po**s-t**el*	
*fa-b**i-an**s*	*fa-b**i-an**s*	*Fabian's*
*fa-b**iaen**s*	*fa-b**i-aen**s*	
*straa**t-e***	*straa-**te***	*street*
*stra-**te***	*stra-**te***	
*eerf-**lic**-heit*	*eerf-**lic**-heit*	*hereditary*
*erf-fe-**lee-c**heit*	*erf-fe-**leec**-heit*	

when the orthography did not match the phonetic realization. Roman numerals and ordinals such as *cccxc*, *lxxvij*, *xxiiiisten* or *xxvjsten* do not require syllabification, since they are not pronounced according to their graphemic representations. Their pronunciations are respectively: *driehonderdnegentig* ("three hundred ninety"), *zevenenzeventig* ("seventy-seven"), *vierentwintigste* ("twenty-fourth") and *negenentwintigste* ("twenty-ninth").

§**17** In conclusion, the data used as training material for the syllabification experiment presented in this paper comprises 43,710 Middle Dutch syllabified words, alphabetically ranging from *a* to *zy-wer-des* ("sideways"). Statistics on the average length of words, the average number of syllables per word and the average number of characters per syllable are provided in **Table 6**. The entire data set is also made freely available for exploration and research purposes (Haverals 2018).

5.2 Model

§**18** As an alternative to the rule-based methodology of Bouma and Hermans (2012), we employ a data-driven method, which is able to infer correct syllable boundaries solely from evidence. The evidence in this case is the human-reviewed list of 43,710 syllabified Middle Dutch words. Our model architecture is based on a fairly straightforward character-level Recurrent Neural Network (RNN) to produce syllable segmentations on the basis of the output of a stack of Long-Short Term Memory (LSTM) layers, as illustrated in **Figure 3**. The original paper on LSTM machine learning is by Hochreiter and Schmidhuber (1997). One of the great benefits of the LSTM-model lies in its capability to take into account the larger context: by letting information flow throughout the entire sequence model, the model learns not only from the immediately adjacent graphemes, but has the ability to also retain information about the entire sequence. This way, the model is especially efficient at learning about e.g. the maximal saturation of the onset of a syllable and morpheme boundaries.

Table 6: Statistics on the data, used as training material for the syllabification experiment. Total number of words = 43,710. Total number of syllables = 115,398.

Distribution of...	characters per word	syllables per word	characters per syllable
Mean	8.05	2.64	3.05
Std	2.55	0.96	1.01
Min	1	1	1
25%	6	2	2
50%	8	3	3
75%	10	3	4
Max	27	9	9

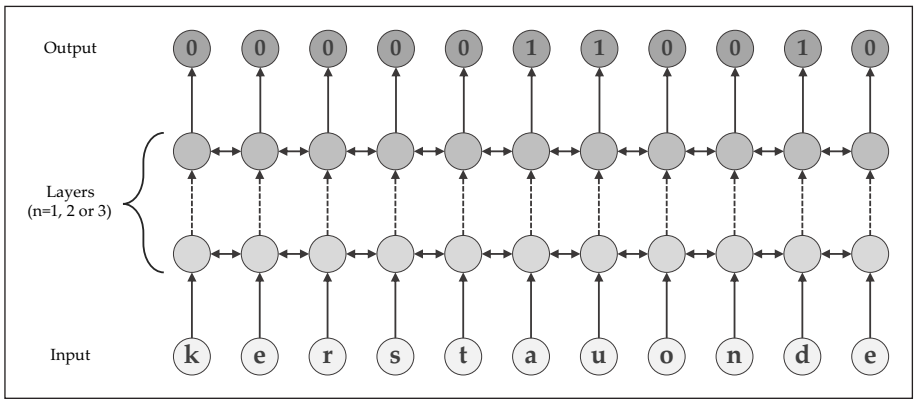

Figure 3: Model architecture of the LSTM machine learning algorithm, taking the word "kerstauonde" as input and predicting the output to be "kerst-a-uon-de".

§**19** For training the model, we split up the available data into three subsets: a training set, a development set and a test set. We applied stratification on the basis of the number of syllables per word, making sure that all splits contained a comparable per-word syllable distribution. The model is optimized on the training set, which represents 80% of all the data (34,968 words) and its performance is monitored (at the end of each epoch) on the development set, holding 10% (4,371 words) of the original data. We only store the model which minimizes the loss on the development set. Finally, the best model is evaluated on the test set, which takes up another 10% of the total amount of data.

§**20** Each word in our model is presented to the model at the character-level, i.e. as a sequence of graphemes. As customary in this sort of models, we do not represent graphemes with a so-called one-hot encoding, but we use embeddings to represent each grapheme (with a fixed dimensionality of 64). Additionally, special symbols are appended to the beginning and end of each token (BOS and EOS). Finally, words get padded to a standard length (i.e. the size of the longest training token + 2, to accommodate for the BOS and EOS symbols) using a dedicated padding symbol (PAD). Naturally, the predictions for these dummy symbols were not included in the final evaluation. Characters that were not encountered in the training material receive a special encoding (UNK). The task of the model, then, is to predict either 0 or 1 for each grapheme. When 0 is predicted there is no indication of a syllable

boundary before this particular grapheme. The prediction of 1 means that a syllable boundary is detected right before this grapheme.

§**21** All code for this paper is available from GitHub; the code uses Python 3.6+ and has the following major dependencies: NumPy (Oliphant 2006), SciKit Learn (Pedregosa et al. 2011), Keras (Chollet et al. 2015) and TensorFlow (Abadi et al. 2016).

5.3 Results

§**22** When evaluating the model, we make a distinction between *word accuracy* and *hyphenation accuracy*. Word accuracy is the percentage of fully correct syllabified words, whereas hyphenation accuracy is the percentage of correctly inserted hyphens across all words and across all syllable boundaries. The latter can be calculated at the character-level. An illustration of both concepts is provided in the word list, shown in **Table 7**. Word accuracy in this fictitious example is fairly low at 60% since only 3/5 words are correctly syllabified. With 9 out of 11 hyphens placed correctly, hyphenation accuracy is at 82%.

§**23** The results presented in **Table 8** and **9** are obtained after a training regime of 30 epochs, with a batch size of 50 words. We used the cross-entropy loss in

Table 7: Exemplary words list with predictions and correct syllabifications. This list serves as an example for the purpose of explaining the difference between word and hyphenation accuracy. Examples are not actual predictions made by the model.

Prediction	Correct	Word	Hyphens
smou-te-nee-ren	*smou-te-nee-ren*	✓	3/3
wa-ter-loes-weru-en	*wa-ter-loes-wer-uen*	✗	3/4
aen-slaen	*aen-slaen*	✓	1/1
arm	*arm*	✓	0/0
cos-te-liken	*cos-te-li-ken*	✗	2/3

Table 8: Word accuracy (epochs = 30, dropout = 0.25, embedding dimension = 64).

Test score	**1 layer**		**2 layers**		**3 layers**	
Dev score						
64 dimensions	95.61%	95.08%	96.84%	96.59%	97.16%	96.96%
128 dimensions	95.70%	95.29%	97.19%	97.46%	96.87%	96.71%
256 dimensions	96.57%	96.00%	**97.55%**	**97.52%**	97.19%	97.52%

combination with the Adam optimizer (Kingma and Ba 2014), with an initial learning rate of 0.001. This learning rate was reduced by a factor of 0.3 after each epoch if the validation loss did not decrease. The model was trained under different conditions, during which we varied the number of hidden layers (1 up to 3) and the number of dimensions (64 up to 256). We insert a moderate level of so-called dropout ($p = .25$) after the embedding layer, between the recurrent layers, and before the final dense layer (Srivastava et al. 2014). The key idea here is that by randomly dropping units (along with their connections) during training, overfitting can be prevented.

5.4 Model inspection

§**24** The best results are obtained with the two-layered model combined with 256 dimensions. Under these circumstances, our model yields a word accuracy of 97.55% (**Table 8**) and a hyphenation accuracy of 99.50% (**Table 9**) on the test set. Overall, scores improve most when stepping up from a 1-layered to a 2-layered model. When adding a third layer, some minor improvements are also noticeable, but overall scores appear to have already stabilized in the 2-layered model. As to the number of dimensions, the difference of scores between 64 and 256 dimensions never exceeds an improvement of 1.00% on the word level and 0.20% on the hyphenation level.

§**25** We compared the output of our best LSTM-model with the output of Bouma and Hermans' rule-based model (both on the same test set of 4,371 words). From this comparison, we gained the results shown in **Table 10**: on the level of word accuracy, the LSTM-model (97.55%) outperforms Bouma and Hermans' model (91.33%) by 6.25%. On the hyphenation accuracy-level, the improvement is more subtle with an increase of 1.53% over the rule-based model.

Table 9: Hyphenation accuracy (epochs = 30, dropout = 0.25, embedding dimension = 64).

Test score	**1 layer**		**2 layers**		**3 layers**	
Dev score						
64 dimensions	99.19%	99.07%	99.38%	99.31%	99.44%	99.40%
128 dimensions	99.24%	99.16%	99.47%	99.47%	99.37%	99.32%
256 dimensions	99.34%	99.22%	**99.50%**	**99.48%**	99.46%	99.46%

Table 10: Comparison of results between Bouma and Hermans' rule-based model and the LSTM-model on the test set (n of words = 4,371).

	Bouma & Hermans (2012)	Our model
Word accuracy	91.33%	97.55%
Hyphen accuracy	97.99%	99.50%
Levenshtein distance	.17	.04
F_1-score	.95	.99

§**26** Additionally, both the Levenshtein distance and F_1-score were calculated for the above-mentioned best model. The Levenshtein distance is a useful metric, often applied in spell checking algorithms, for measuring the amount of difference between two sequences (Levenshtein 1966). Briefly put, the logic behind this metric is defined by the number of edits that are required to convert one string into the other.[4] In our case, we apply the Levenshtein distance in order to compare the correct, gold standard syllabifications to the predictions made by our model. As one can see in **Table 10**, the Levenshtein distance of our model is very low with an average of .04 edits, which is more than three times as low as the distance calculated for Bouma and Hermans' model (.17). With the F_1-score, finally, we wanted to gain insight in the balance between precision and recall on the character level, because there is a significant imbalance for the two classification labels in our model. Here also, the score obtained for the LSTM-model is nearing the perfect score of 1.0.

§**27** Surely, developing an automatic syllabifier is only really interesting if it can also be effectively deployed onto other corpora. In order to get a good understanding of the syllabifier's potential, we evaluated our model on an out-of-corpus sample of Middle Dutch words. To this end, we randomly selected 2,000 words from the *Cd-rom Middelnederlands* (1998). Unlike the legal and administrative character of the *Corpus Van Reenen-Mulder*, the *Cd-rom Middelnederlands* is a corpus of literary texts, both rhymed and prose. In order to make sure that our evaluation was carried out

[4] As a clarifying example for the Levenshtein distance, consider the following two syllabifications: *af-ter-wards* and *aft-er-ward-s*. The Levenshtein distance here is 3, since it would require three edits in order to transform one sequence into the other (twice the deletion and once the insertion of a "-").

Table 11: Comparison of results between Bouma and Hermans' rule-based model and the LSTM-model on an out-of-corpus sample of 1,748 Middle Dutch words.

	Bouma & Hermans (2012)	**Our model**
Word accuracy	89.24%	98.74%
Hyphen accuracy	97.64%	99.76%

on new, unseen words to the model, we cross-checked the sample from the *Cd-rom Middelnederlands* with the CRM-word list. In total, we found that 1,748 out of the 2,000 words did not occur in the CRM, and thus were unseen to the model. From those 1,748 words, only 22 (1.26%) were syllabified incorrectly by the LSTM-model, whereas Bouma and Hermans' model made 188 mistakes (10.75%) (**Table 11**). Also on the level of hyphenation accuracy, the LSTM-model achieves a remarkable result of 99.76%.

§**28** The observation that our model's performance is even better on a random sample of unseen words is likely due to the fact that word frequency was not taken into account when training the model. Because the LSTM-model was trained on *all* the words from the CRM-corpus (i.e. it has no knowledge of which words are more common and which ones are more rare), mistakes are most likely made against low-frequent words. The scores shown in **Table 10** can therefore be an underestimation of the model's performance "in the wild".

5.5 Model criticism

§**29** Where does it still go wrong? From an inspection of the mistakes made by both our LSTM-model and Bouma and Hermans' model, we learn the following (**Table 12**): **(1)** the LSTM-model is very accurate at respecting morpheme boundaries. We notice this especially from adjectives ending in *-heit* and adverbs ending in *-like*. In almost all cases, such words are syllabified correctly by the LSTM-model, which rightly treats the suffixes of these words as independent domains of syllabification (e.g. *domp-**li-ke***, *ern-stic-**heyt***, *rijp-**heyt***, *siec-**he-de***, etc.). In syllabifications produced by Bouma and Hermans' model, we notice that the final letter of the stem sometimes gets added to a morpheme that it does not belong to (e.g. *dom-**pli-ke***, *ern-sti-**cheyt***, *rij-**pheyt***, *sie-**che-de***, etc.). Also prefixes like *and-*, *ver-* and *on-* are kept intact by the

Table 12: Examples of syllabification errors made by both Bouma and Hermans' rule-based model and the LSTM-model. Color code: white cells are correctly syllabified words; grey cells are syllabification errors.

Correct syllabification	Bouma and Hermans (2012)	Our system
aert-se-bis-cop	*aert-se-bi-scop*	*aert-se-bi-scop*
and-war-de	*an-dwar-de*	*and-war-de*
bae-re-uoet	*bae-reu-oet*	*bae-re-uoet*
be-ruer-ten	*be-ruer-ten*	*ber-uer-ten*
bloeit	*bloe-it*	*bloeit*
con-uen-tu-a-le	*co-nuen-tu-a-le*	*con-uen-tu-a-le*
cri-eer-den	*cri-eer-den*	*crieer-den*
des-tru-e-ren	*de-strue-ren*	*des-true-ren*
domp-li-ke	*dom-pli-ke*	*domp-li-ke*
dy-o-cle-ti-aen	*dy-o-cle-tiaen*	*dy-o-cle-ti-aen*
ern-stic-heyt	*ern-sti-cheyt*	*ern-stic-heyt*
ghe-en-det	*gheen-det*	*gheen-det*
ko-ninck-ri-ken	*ko-ninc-kri-ken*	*ko-ninc-kri-ken*
moey-te	*moe-y-te*	*moey-te*
on-uer-hoe-len	*o-nuer-hoe-len*	*on-uer-hoe-len*
recht-ueer-dic-heit	*rech-tue-er-di-cheit*	*rech-t-ueer-dic-heit*
rijp-heyt	*rij-pheyt*	*rijp-heyt*
sach-ic	*sa-chic*	*sa-chic*
siec-he-de	*sie-che-de*	*siec-he-de*
twij-uel	*twi-juel*	*twij-uel*
vet-heit	*ve-theit*	*vet-heit*
ver-uairt	*ve-ru-airt*	*ver-uairt*
vray-lijc	*vra-y-lijc*	*vray-lijc*
vreemt-he-de	*vreem-the-de*	*vreem-the-de*
waeyt	*wae-yt*	*waeyt*
wraec	*w-raec*	*wraec*
wrac-ghier	*w-rac-ghier*	*wrac-ghier*
zot-hei-de	*zo-thei-de*	*zot-hei-de*

LSTM-model, (e.g. ***and****-wer-de*, ***on-uer****-hoe-len*, ***ver****-uairt*). **(2)** The latter examples also show that the LSTM-model is highly efficient at discerning whether the grapheme ⟨u⟩ has to be pronounced as either /y/ or /v/, which was one of the challenges raised in section 3. **(3)** However, the LSTM sometimes also lapses. This is the case with words

that have a syllable boundary between two vowels, like *crieerden*, *destrueren* and *gheendet*. It is not surprising that things go wrong here. The LSTM-model has seen more instances where the graphemes ⟨ee⟩ and ⟨ue⟩ remain together, than where they are separated. Although it does syllabify *crieerden* correctly, Bouma and Hermans' model also frequently makes mistakes against words that contain syllable boundaries between vowels. Overall, it is important to note that where the LSTM goes wrong, so does Bouma and Hermans' model. Out of the 26 word-level mistakes made by the LSTM-model on the entire out-of-corpus sample, 20 of those mistakes are also made by Bouma and Hermans' model. The reverse, however, is not true. Where Bouma and Hermans' model goes wrong, the LSTM usually has it right.

6 Conclusion

§**30** Essentially, there are two approaches to the task of automatic syllabification: rule-based and data-driven. An automatic syllabifier for Middle Dutch was first developed by Bouma and Hermans (2012), whose approach fundamentally is a rule-based one. The way they approach the task is very elegant and the scores they achieve are high. Nevertheless, one could argue that specifically for Middle Dutch, their model is not a very robust one. By heavily relying on a set of rules that describe possible nuclei, onsets and codas, their model underestimates the somewhat erratic nature of Middle Dutch orthography. Because the spelling of Middle Dutch allows a lot of variation both in diachronic and synchronic terms, it is risky business to hard-code this information. The automatic syllabifier presented in this paper responds to the need of *not* having to explicitly describe any definitions, and thus guaranteeing more flexibility when it comes to spelling variation. By resorting to a purely data-driven method, our model is extremely effective at predicting syllable boundaries while respecting morpheme boundaries. Using LSTM machine learning techniques, we obtain high results at the word level: 97.55% on the test set of the training material corpus, and 98.74% on an out-of-corpus sample. The results of the automatic syllabifier for Middle Dutch are therefore in line with comparative research on different syllabification methods, finding data-driven methods to outperform rule-based techniques usually by huge margins Marchand et al. (2009).

§**31** As Bouma and Hermans (2012) have established in their article, an automatic syllabifier for Middle Dutch can serve a great deal of interesting research questions. In the domain of historical linguistics, for example, the possibility to syllabify an entire corpus of Middle Dutch text allows the researcher to trace both spelling conventions and phonological change. At the same time, accurately syllabifying Middle Dutch words can be of interest to the literary scholar. As shown by Hench (2017) for Middle High German, syllabification is essential for gaining insight in the soundscapes of medieval poetry. Finally, syllabification is an essential stepping stone in metrical studies. Before one can determine the rhythmical aspects of e.g. rhymed medieval poetry, precise and consistent syllabification is mandatory.

Contributors

Editorial

Recommending editor: Franz Fischer, Università Ca' Foscari Venezia

Recommending referees: Katrien Depuydt, Instituut voor de Nederlandse Taal; Felix Rau, Universität zu Köln

Authorial

The corresponding author is wh. Authorship is alphabetical after the drafting author and principal technical lead. Author contributions, described using the CASRAI CRedIT typology (Consortia Advancing Standards in Research Administration Information, 2018), are as follows:

Conceptualization: wh, fk, mk

Methodology: wh, mk

Software: wh, fk, mk

Investigation: wh

Data curation: wh

Writing – Original Draft Preparation: wh

Writing – Review and Editing: wh, fk, mk

Visualization: wh

Supervision: fk, mk

Competing Interests

The authors have no competing interests to declare.

References

Abadi, M. et al. 2016. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems* (https://tensorflow.org).

Booij, G. 1999. *The Phonology of Dutch*. Oxford: Oxford University Press.

Bouma, G. 2003. "Finite State Methods for Hyphenation." *Natural Language Engineering* 9(1): 5–20. DOI: https://doi.org/10.1017/S1351324903003073

Bouma, G., and **B. Hermans.** 2012. "Syllabification of Middle Dutch." In: F. Mambrini, M. Passarotti, and C. Sporleder (Eds.), *Proceedings of the Second Workshop on Annotation of Corpora for Research in the Humanities,* 27–39.

Brill, E. 1995. "Transformation-based Error-driven Learning and Natural Language Processing: A Case Study in Part-of-speech Tagging." *Computational Linguistics* 21(4): 543–565.

Chollet, F. et al. 2015. *Keras* (https://keras.io).

van Halteren, H., and **M. Rem.** 2013. Dealing with Orthographic Variation in a Tagger-lemmatizer for Fourteenth Century Dutch Charters. *Language Resources and Evaluation* 47(4): 1233–1259. DOI: https://doi.org/10.1007/s10579-013-9236-1

Haverals, W. 2018. Middle Dutch syllabified words. [Data set]. DOI: http://doi.org/10.5281/zenodo.2402048

Hench, C. 2017. Phonological Soundscapes in Medieval Poetry. In: *Proceedings of the Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, Vancouver, 46–56. DOI: https://doi.org/10.18653/v1/W17-2207

Hochreiter, S., and **J. Schmidhuber.** 1997. Long Short-Term Memory. *Neural Computation* 9(8): 1735–1780. DOI: https://doi.org/10.1162/neco.1997.9.8.1735

Instituut voor de Nederlandse taal. 1998. *Cd-rom Middelnederlands. Woordenboek en teksten.* Den Haag/Antwerpen: Sdu Uitgevers.

Kager, R., and **W. Zonneveld.** 1986. Schwa, Syllables, and extrametricality in Dutch. *The Linguistic Review* 5(3): 197–222. DOI: https://doi.org/10.1515/tlir.1986.5.3.197

Kingma, D. P., and **J. Ba.** 2014. Adam: A Method for Stochastic Optimization. In: *arXiv preprint arXiv:1412.6980, Volume abs/1412.6980. Published as a conference paper at the 3rd International Conference for Learning Representations*, San Diego, 2015.

Levenshtein, V. 1966. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady* 10: 707–710.

van Loey, A. 1976. *Middelnederlandse spraakkunst. II: Klankleer* (7 ed.). Groningen: Wolters.

Marchand, Y., C. Adsett, and **R. Damper.** 2009. Automatic Syllabification in English: A Comparison of Different Algorithms. *Language and Speech* 52(1): 1–27. DOI: https://doi.org/10.1177/0023830908099881

Oliphant, T. E. 2006. *A guide to NumPy* (vol. 1). USA: Trelgol Publishing.

Pedregosa, F. et al. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12: 2825–2830.

van Reenen, P., and **M. Mulder.** 1993. Een gegevensbank van 14de-eeuwse Middelnederlandse dialecten op computer. *Lexikos* 3: 259–281. DOI: https://doi.org/10.5788/3-1-1110

Selkirk, E. 1982. The Syllable. *The Structure of Phonological Representations* 2: 337–383.

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever, and **R. Salakhutdinov.** 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15: 1929–1958.

Trommelen, M. 2011. *The Syllable in Dutch.* Berlin/New York: De Gruyter Mouton.

Vennemann, T. 1988. *Preference Laws for Syllable Structure, and the Explanation of Sound Change with Special Reference to German, Germanic, Italian, and Latin.* Berlin/Boston: De Gruyter Mouton. DOI: https://doi.org/10.1515/9783110849608

Verdam, J., and **E. Verwijs.** (Eds.) 1885. *Middelnederlandsch Woordenboek (MNW).* Dan Haag: Nijhoff. Also accessible on-line through the 'Gentegreerde Taalbank' (http://gtb.inl.nl).

How to cite this article: Haverals, Wouter, Folgert Karsdorp, and Mike Kestemont. 2019. "Data-Driven Syllabification for Middle Dutch." *Digital Medievalist* 12(1): 2, pp. 1–23. DOI: https://doi.org/10.16995/dm.83

Published: 04 November 2019

Digital Medievalist is a peer-reviewed open access journal published by Open Library of Humanities.

OPEN ACCESS