Delemmatization Strategies for Dutch

Louis Onrust and **Hans van Halteren**

Abstract

In this paper we investigate whether, for Dutch open class words, it is possible to generate the surface form on the basis of the lemma and the POS tag, using a lexicon and a machine learning system. When testing on an annotated corpus, we are able to generate more than 97% of the gold standard types correctly and over 99% of the gold standard tokens. The most efficient strategy appears to be the pure machine learning one, even for those words that do occur in the lexicon. Apart from overall statistics, we look at specific machine learner settings in more detail and also investigate the errors made by the best scoring strategy.

1. Introduction

In recent years, more and more attention is given to text generation, e.g. in applications such as summarization, translation, reformulation, and subtitle generation. The final stage of the generation, however, the creation of surface forms from a lemma and desired morphosyntactic properties, has not received much attention in the literature. In this paper, we describe and evaluate various strategies for what we call delemmatization for Dutch. Our primary reason to start this project was that we have a system which annotates 14th century Dutch texts with POS tags and modern lemmas (van Halteren and Rem 2013), and we would like to extend this into a system which can rephrase such texts into modern form. However, the component we are building can be used in any application where surface form generation is needed for an open-ended vocabulary, and not just for a static (and possibly limited) lexicon. In principle, this includes all the abovementioned applications. A possible exception might be translation, as this is mostly statistical these days, and translation is therefore directly to the surface form, rather than to a lemma. However, one can also imagine a kind of statistical translation where the transfer step is at a more abstract level than surface forms, for which our surface generation system would become useful again. Another application, which we had (perhaps naively) not originally envisioned, is quality control and correction of lexicons and annotated corpora. As we will see below, discrepancies between system output and existing corpora rather often point to errors in the corpus annotation or in the lexicon used for creating the delemmatizer. In this paper, though, we will not perform any extrinsic evaluation by investigating any actual applications of the delemmatizer, but will restricting ourselves to an intrinsic evaluation.

As for many other languages, the words in Dutch can be separated into closed and open class words (Booij and Van Santen 1998). The closed class words, such as articles, pronouns and auxiliary verbs, show often idiosyncratic forms and delemmatization will have to done simply by lexicon lookup. As the list of closed class words is indeed closed and well-described, the task is known to be feasible. Open class words, on the other hand, cannot be expected to always be present in a lexicon. Fortunately, their inflection tends to be regular in Dutch. There are irregular inflections also for neologisms, but these would be applied only when the word in question is a compound containing a known irregularly inflected word. An example would be *winstgenomen* ("sold stock to realize profit") as the past participle of *winstnemen.* A special class is formed by loan words and their compounds, which might be inflected with Dutch morphology as well as with the morphology of the source language, e.g. the plural of *centrum* ("center") can be either *centra* (Latin) or *centrums* (Dutch). We want to investigate how we can generate the surface form from a lemma and a POS tag for open class words, so that we can perform this task also for unseen proto-text, potentially containing unknown lemmas or lemma-tag combinations.

We expect that software for this task must have been implemented many times before. However, publications relating to such implementations and their testing prove hard to find. For Dutch, e.g., we do know of work by Vandeghinste (personal communication), and of course there are various tools using Finite State Transducers, e.g. those from Xerox (Beesley and Karttunen 2003, Beesley 2004). Mostly, it is remarked that tools are bidirectional, but experimental results are generally based on analysis tasks. As an example Vaneyghen et al. (2006) generate surface forms on the basis of lemmas from Celex (Baayen et al. 1995), using an FST system and restrictions on the output, but then measure the quality of the system in an analysis setting, namely in terms of coverage on corpus texts. As another example, for English, Carl et al. (2005) present a rule-based delemmatisation system for English within the context of a corpus-based machine translation system. They too assume a process in which lemmatisation and delemmatisation are reversible. They report an accuracy of 99.5% for delemmatising lemmas into tokens, but do not explain their experimental setup, nor discuss any of the results. Taking the 99.5% at its face value, we should remark that English is less susceptible to nominal inflection than Dutch, which means that noun inflection is an easier task for English than for Dutch. Finally, most of the reported work involves manually constructed rules, while we want to focus on machine learning from existing resources, with the option of adding in further resources if and when they become available.

To be precise, we study the surface form generation for nouns, with specific number, case and diminuization, for adjectives, with specific degree and conjugation, and for verbs, with all possible verb forms. We investigate three strategies. In the pure lexicon strategy, we simply look up the desired form in a lexicon, for which we have chosen e-Lex (see http://tst-centrale.org/nl/producten/lexica/e-lex/7-25). In the pure machine learning strategy, we learn transformation rules from the information in the same lexicon and apply the rules to generate forms. In the hybrid strategy, we use the lexicon information for known words and apply the learned rules for unknown words. We evaluate the strategies on the basis of the manually corrected part (one million words) of the syntactically annotated LASSY corpus (Van Noord et al. 2006). We supply our system with the lemma and the desired morphosyntactic information that is present in the annotation and compare the form suggested by the system with the form actually present in the corpus.

In the following sections, we first (Section 2) describe our experimental data in more detail and outline the evaluation method. We then (Section 3) describe the various delemmatization strategies that we investigated. We continue with a discussion of the results, starting with a comparison of the main strategies (Section 4) and following with a comparison of various settings for the best strategy (Section 5). Then we examine the difference between the forms suggested by the best system and the (gold standard) forms in the test corpus (Section 6). Section 7 concludes the paper.

2. Experimental Data and Evaluation

In this section, we first describe the tagset that we will use in our delemmatizer (Section 2.1). Then we describe our experimental data, being the e-Lex lexicon for development (Section 2.2) and the Lassy Klein corpus for testing (Section 2.3), and finally how we evaluate the various strategies and settings (Section 2.4).

2.1 Tagset used in the Experiments

In our experiments, we are attempting to apply specific types of inflection. It is therefore these types of inflection which we must express in a tagset so that we can supply a POS tag together with a lemma as complete input for the system.

We indicate nouns with the major class N, adding attributes for diminutive or not (dim=dim and dim=norm), for singular and plural number (num=sing and num=plu), and for nominative, dative and genitive case (case=nom, case=dat and case=gen). Not all combinations occur and especially the dative is limited as it occurs only in a few archaic expressions in modern Dutch. A full list of occurring tags can be found in Section 6.2, where we examine the error rate per tag. For adjectives and adverbs, which largely overlap, we use the main POS tag Adjv, with attributes for positive, comparative and superlative degree (deg=pos, deg=comp and deg=sup), as well as for the syntactically determined suffix (form=norm, form=e, form=en). For verbs, indicated with V, there are more options. The finite forms (fin=fin) have attributes for present or past tense (tense=pres and tense=past), and for singular or plural number (num=sing and num=plu). In addition, there is an attribute for form, which is normally unused (form=norm), but becomes active for 3rd person singular present tense (form=t). Then there is the infinitive (fin=infin), which in principle can have the basic form (form=norm) as well as a suffix -s (form=s), which might occur in expressions like *tot huilens toe* ("until she was crying"), but which we have not found in our data. Finally we have the present and past participles (fin=part, with tense=pres and tense=past), which can get the same forms as adjectives when used adjectively, leading to a final attribute (form=norm, form=e and form=en).

This tagset being the basis of our experiments, we projected the POS tags found in both e-Lex and Lassy, which are in principle already mutually compatible as they are based on the CGN tagset (Van Eynde et al. 2000). Although this may mean some loss of information, it makes it easier to add further lexicons and corpora in potential later experiments. As an example, the tag for *woordje* ("little word") was mapped from N(soort,ev,dim,onz,stan) (i.e. Noun, common, singular, diminutive, neutral, standard form) to N(dim=dim,num=sing,case=nom).

2.2 e-Lex

Our training material is taken from e-Lex (v1.1), a large lexicon for Dutch, based mostly on Celex (Baayen et al. 1995) and the CGN (Oostdijk 2000), but also incorporating information from various other sources. It can be ordered via http://tst-centrale.org/nl/producten/lexica/e-lex/7-25.

For the word classes we are interested in, we find the statistics in Table 1. For each statistic, we list the number for the original e-Lex lexicon as well as for the version we created by removing diacritics and transforming all letters to lower case (normalization, see Section 2.4). As we can see, normalization mostly affects nouns. On the other hand, nouns are hardly affected by the projection of the POS tag. Here, the main impact is on adjectives/adverbs, for which the CGN tagset includes e.g. attributes indicating the syntactic use.

Word class	Lemmas		Lemma-tag pairs (e-Lex tags)		Lemma-tag-form triples (e-Lex tags)		Lemma-tag-form triples (projected tags)	
	original	normalized	original	normalized	original	normalized	original	normalized
Noun	153,510	149,979	237,982	236,841	243,381	241,388	238,773	234,345
Verb	17,894	17,869	163,745	163,716	194,486	194,283	121,983	121,806
Adj/Adv	18,958	18,839	133,913	133,748	143,427	143,036	57,578	57,230

Table 1: Open Class Words in e-Lex

Of the various information fields in e-Lex, we only use the lemma-tag-form triples and their corresponding frequency counts, which have been derived from, among others, the Corpus of Spoken Dutch (CGN; Oostdijk 2000). Note that, because of its use of corpus material, e-Lex also contains non-conventional forms encountered there, e.g. *ademhalingsmoeilijkheid* ("trouble breathing") not only has the normal plural *ademhalingsmoeilijkheden*, but also *ademhalings-uh-moeilijkheden*. As a result, the listed frequency becomes important information. Generally, more frequent forms should give a more reliable picture of inflection than rare forms. Then there are some forms that were present in source lexica but have not been observed in corpus material. As a result, e-Lex also lists these, but with frequency zero. We do use these forms in our experiments. To handle competition between observed and unobserved forms, we add one to all frequencies.

2.3 Lassy Klein

Our test material consists of the Lassy Klein corpus, a subset of about 1Mw of the Dutch Lassy corpus (Van Noord et al. 2006). It was automatically tagged with the Tadpole tagger (van den Bosch et al. 2007) and parsed with the Alpino parser (Bouma et al. 2001, Van Noord 2006), and subsequently completely manually verified and corrected. It can be ordered at http://tst-centrale.org/nl/producten/corpora/lassy-klein-corpus/6-66.

For the word classes we are interested in, we find the statistics in Table 2. The relative numbers of the various word classes is as expected, as is the number of types and the form ambiguity. None of these is overly important for our current experiment, though. For that, we need to know whether the word forms in the corpus can be generated, and here a comparison of the rightmost two columns is necessary. Here we see that a substantial percentage of the lemma-tag-form triples that we encounter in Lassy cannot be found in e-Lex.

Now we know that especially the noun class is very open, so that a high percentage of unseen nouns is likely, but we had not expected that almost half the nouns would be absent from the lexicon. It should be noted, of course, that we are examining types here and not tokens. For adjectives/adverbs and especially verbs, the situation is better, but still not very good. Even without running any experiments, we can already conclude that a pure lexicon lookup strategy will not be sufficient for a production system that is supposed to process unseen text.

Word class	Tokens	Form Types		Lemma-tag-form triple types		Lemma-tag-form triples present in e-Lex	
		original	normalized	original	normalized	original	normalized
Noun	246,765	44,075	40,028	44,540	40,363	24,266 (54.5%)	24,818 (61.4%)
Verb	142,294	11,283	10,502	12,537	11,761	11,153 (89.0%)	11,251 (95.6%)
Adj/Adv	80,398	8,952	7,750	9,060	7,846	6,240 (68.9%)	6,276 (79.9%)

Table 2: Open Class Words in Lassy Klein

2.4 Evaluation

During development, we only access the e-Lex lexicon and any parameters learned by the machine learning system are also based on the lexicon. Where appropriate (see e.g. weighting in Section 3.2), we do access frequency information provided in the lexicon. When testing, we process each token in the Lassy Klein corpus individually. We provide each system with the lemma and the tag and let the system suggest a surface form. For the main evaluation, we will take the surface form in the Lassy Klein corpus as the gold standard. Later (Section 6.1), though, we will find that some discrepancies between system and corpus find their cause in corpus errors.

There is some orthographic variation that hinders a straightforward evaluation. Upper case may be a characteristic of a word, but often is also just caused by the start of a new sentence, a desire to stress a word or the word being part of a name. Similarly, use of diacritics greatly varies with the medium or the personal writing style of an author. As we are interested in morphology rather than syntax, discourse and orthographic ideolect, we ignore case and diacritics in most of our evaluation. We remove all diacritics and transform all letters to lower case. However, in Section 5.4, we will specifically investigate the effect of normalization on the accuracy of the system.

3. Delemmatization Strategies

In this section, we describe the three main strategies that we investigated for the delemmatization task. We start with the Lexicon Strategy (Section 3.1), continue with the Machine Learning Strategy (Section 3.2). In addition, we use a Hybrid Strategy (HYB) which combines the first two. It uses the form provided by LEX if the input lemma-tag pair is known and the form provided by ML otherwise. The idea is that, when present, the information in the lexicon is more exact than the suggestions from the machine learner.

3.1 Lexicon Strategy

The lexicon strategy consists, simply, of looking up the lemma-tag pair in the projected e-Lex lexicon. If the pair is present, the form with the highest noted corpus frequency is chosen. In case of ties, we choose randomly from among the tying forms. If the pair is not present in the lexicon, we use the heuristic fallback of taking the lemma itself as the form. A great many of the unknown words in running text should be singular nouns and non-inflected adjectives/adverbs, and for both these classes, the lemma is exactly what we need.

3.2 Machine Learning Strategy

In the machine learning strategy, we treat delemmatization as a classification problem, with the classes representing the edit script that must be applied to the lemma to create the surface form. As an example, plural formation could be with the script +.s (with the dot representing the body of the word) to form *molens* ("windmills") from *molen* or +.en to create *boeren* ("farmers") from *boer*. However, there may be more needed than suffixation of the plural suffix, as e.g. +.ten to create *katten* ("cats") from *kat*, -.s/+.zen to create *huizen* ("houses") from *huis*, or -.ot/+.ten to create *boten* ("boats") from *boot*. The most complex operation for normal words is the creation of the past participle, which may involve prefixation of *ge-* in addition to a suffixation. For example, -.en/+ge.d will create the past participle *gebouwd* ("built") from *bouwen*. In case of a separable verb, the prefix *ge-* becomes an infix *-ge-* and more has to be taken off, e.g. -.bouwen/+.gebouwd is needed to create *opgebouwd* ("built up") from *opbouwen* and *uitgebouwd* ("extended") from *uitbouwen*. This approach, using the current lexicon, yields around 8,500 different edit scripts and therefore machine learning classes. The classes show a mostly Zipfian distribution. At the top we find the null-script (almost 200,000 occurrences), then plural noun formation with straightforward addition of -s and -en (both about 25,000 occurrences) and further regular forms. At the bottom we find over 6,000 classes with only one occurrence, among which many past participle forms.

With this number of classes, and given the expected number and type of features (see below), there are still various machine learning systems that could be used. We decided to use Timbl (v6.4.2; Daelemans et al. 2004), mostly because of ease of availability (but we should note that an excursion to support vector machines yielded far lower accuracy). Our choice for Timbl does force us to provide a fixed number of columns with feature information for each case. We represent the tag as a single feature, just containing the tag itself. No attempt at splitting the tag into different information fields is made. The most important features for the lemma are taken to be various substrings at the end of the lemma. As an example, the lemma *generaal* ("general") leads to the features "l", "al", "aal", "raal", "eraal", etc. The maximum length is a system setting. In the experiments below we have used a maximum of 40, which should be enough to cover the whole word for practically all tokens in the corpus. Note that even "eraal" would not be sufficient for *generaal*, as its plural is *generaals*, while the plural of *mineraal* ("mineral") is *mineralen*. And, as already indicated above, for verbs it may be necessary to address the whole word. In addition, there is a feature indicating a possible prefix, such as "op" in the example *opbouwen* above, and a feature indicating whether the lemma starts with a capital letter. The latter is not used in our experiments as we have transformed all words to lower case.

As has already been explained above, e-Lex may contain several possible surface forms associated with the same lemma-tag pair, and some of these forms may be rare corpus occurrences. As a result, we want to be able to weight the importance of the various training cases. Unfortunately, Timbl does not allow user weighting of cases. The only way to provide weights is by including the same case more times. Including each case as many times as the corpus frequency listed in the lexicon would lead to a case base which is far too big for efficient processing. We have therefore decided to count the frequency of each suggested form relative to the sum of all frequencies listed with the corresponding lemma-tag pair, and then round this fraction to the nearest 10 percent. For each 10 percent covered, the form receives one case in the case base. This means that every lemma-tag pair leads to 10 cases and the case base as a whole is 10 times the number of lemma-tag pairs. This multiplication by 10 is also done for lemma-tag pairs for which only a single form is present, as we want the weight also taken into account in the generalization over all lemma-tag pairs.

Given the resulting size of the case base, we would prefer to use the decision tree algorithm in Timbl (IGtree; Daelemans et al. 1997), which is much faster than the normal k-nn algorithm (IB1; Daelemans et al. 1999). However, before taking a final decision, we first want to test whether the increased speed comes at the cost of decreased accuracy. We therefore test both algorithms, and moreover IB1 with k=1, 2, 3 and 5. As testing will show that IGtree sometimes suggests classes whose edit script cannot actually be applied to the lemma in question, we furthermore add a fallback mechanism which takes the suggestion of IB1 in cases where IGtree supplies an impossible script.

4. Results, the Main Strategies Compared

We measured the effectiveness of all strategies with various settings on Lassy Klein. In this and the following section, we present the results. In all tables, we show the percentage of agreement between the system suggestions and the form found in Lassy Klein, even though we will see in Section 6.1 that this measurement is less exact than we had hoped.

Strategy	Types			Tokens		
	Nouns	Verbs	Adjectives/Adverbs	Nouns	Verbs	Adjectives/Adverbs
LEX	88.70	95.79	85.79	97.09	98.96	97.26
ML	97.43	97.00	97.77	99.18	99.16	99.26
HYB	97.43	96.98	97.77	99.17	99.07	99.26

Table 3: Accuracy Scores for the Main Strategies

Strategy	Types			Tokens		
	Nouns	Verbs	Adjectives/Adverbs	Nouns	Verbs	Adjectives/Adverbs
LEX	98.25	97.64	98.21	99.53	99.35	99.41
ML	98.25	97.66	98.21	99.54	99.44	99.41

Table 4: Accuracy Scores for Known Words for LEX and ML (HYB is not included as, for these words, HYB is equal to LEX)

In Table 3 we compare the overall results for the main strategies. The rows present the various strategies and the columns the agreement with Lassy Klein for the three types of open class words. For all strategies, the best scoring settings are represented; below we will examine the settings in more detail.

As we can see, the lexicon strategy is not doing badly, especially given the coverage shown in Table 2. Apparently, the fallback option of using the lemma if the lexicon contains no information is effective. Still, the lexicon strategy is clearly outperformed by both the machine learning and the hybrid strategy, with statistically significant differences with all $p < 0.00001$, except for LEX-HYB for verb tokens where we find a mere $p = 0.003$. If we do the comparison for only those lemma-tag pairs which are present in e-Lex (Table 4), the differences are much smaller, and only statistically significant for the verb tokens ($p = 0.002$). Furthermore, when we look at the differences, we mostly see lemma-tag pairs where both suggested forms are possible (see also Section 6.1). After removing these, we are left with 9 types (13 tokens) where HYB does better than ML and 8 types (11 tokens) where ML does better than HYB. For all practical purposes, the two perform equally well. The main cause for the apparent difference is the past tense of *willen* ("to want"), where LEX and hence HYB choose the more informal form *wou* and ML the more formal form *wilde*. And since the corpus contains 124 more occurrences of *wilde* than of *wou*, ML scores 0.09% better on verbs.

The difference between ML and LEX also etranslates directly into a difference between ML and HYB, and explains why the pure machine learning strategy appears to perform slightly better than the hybrid strategy. This too is a data-based illusion. The important point, though, is that HYB does not yield an improvement over ML, so that we can choose the less complex pure machine learning strategy. Furthermore, even though the ML strategy is already extremely effective with error rates well under 1% of the tokens in running text for all three open class word types, it can still be improved. The lexicon strategy on the other hand is fixed and hence limits both lexicon and hybrid strategies. The only improvement could lie in cleaning up errors in the lexicon, but from this the machine learning strategy would improve equally.

5. Results, Settings for the Machine Learner

Now we have seen that the machine learning strategy is (at the moment) the best one, it becomes all the more important that we examine which settings to use. We first look at the basic search algorithm within Timbl (Section 5.1). Then we continue with the use of frequency weighting (Section 5.2) and the length of word-end substrings to be taken into account (Section 5.3). Finally, we examine whether normalization of case and diacritics is as important as we expected (Section 5.4).

5.1 Learning Algorithm

The main choices one has to make when using Timbl are for the matching algorithm and the size of the case neighbourhood that will be examined (k). Although the basic matching algorithm, the lazy learning algorithm IB1, is usually preferred, it is slower than the more greedy decision tree algorithm IGtree. As the delemmatizer will tend to be embedded in a larger system, with varying speed requirements, processing time may be important. Similarly, the larger k, the more time will be needed for processing. We therefore measured the processing time and delemmatization quality of IGtree as well as IB1 with k=1, 2, 3 and 5. As we found that IGtree will sometimes suggest an edit script that is inappropriate for the lemma in question, we add a fallback mechanism that uses the opinion of IB1 when this happens. We test this fallback mechanism with k=1, 2 and 3.

As we can see (Table 5), we are lucky in that speed and quality go hand in hand in these experiments. By itself, IGtree works not only faster but also better than IB1. And IB1 works better with smaller rather than larger k. The fallback option does help solve most of the problem with the inappropriate scripts, which however does not turn out to be very frequent. In the 469,457 cases processed, we run into this problem 109 times (0.02%). Fallback to IB1 with k=1 often does not help as it too chooses the inappropriate script (46 times). Going to k=2 brings new options, not always correct, but often. All in all, the fallback to IB1 with k=2 brings correct results for 78 of the 109 problematic cases. In all further experiments, we will therefore use IGtree with a fallback to IB1 with k=2.

Algorithm/k	Processing time	Types			Tokens		
		Nouns	Verbs	Adjectives/ Adverbs	Nouns	Verbs	Adjectives/ Adverbs
IB1-1	27 min	96.71	96.81	97.08	98.96	99.14	99.01
IB1-2	1.5 hr	96.23	80.78	96.48	97.12	69.33	97.30
IB1-3	2 hr	96.28	86.55	96.30	97.36	68.14	96.83
IB1-5	4 hr	95.50	85.17	95.63	96.09	65.77	93.97
IGtree	16 min	97.34	96.80	97.69	99.16	99.14	99.25
IGtree/IB1:k1		97.37	96.95	97.69	99.16	99.16	99.25
IGtree/IB1:k2		**97.43**	**97.00**	**97.77**	**99.18**	**99.16**	**99.26**
IGtree/IB1:k3		**97.43**	96.98	**97.77**	**99.18**	**99.16**	**99.26**

Table 5: Accuracy Scores per Algorithm and Neighbourhood Size k (in bold are shown the highest accuracy, plus all that are not significantly different at the $p < 0.05$ level)

We can conclude that a production system should use the IGtree algorithm with, if speed requirements permit, a fallback to IB1 with k=2 for those cases where IGtree is misbehaving.

5.2 Frequency Weighting

As explained in Section 3.2, we tried to improve the generalization quality of the machine learning algorithm by using various lexicon suggestions for lemma-tag pairs, weighted with the observed frequency of the suggestions. Since this too increases processing time, and since the weighting could only be implemented rather coarsely, we wanted to measure whether weighting was indeed effective.

Weighting	Types			Tokens		
	Nouns	Verbs	Adjectives/Adverbs	Nouns	Verbs	Adjectives/Adverbs
unweighted	97.01	95.70	97.31	98.22	82.25	92.41
weighted	97.35	96.80	97.69	99.16	99.14	99.25

Table 6: Accuracy Scores with and without Frequency Weighting

As we see in Table 6, the effect does not seem very pronounced when looking at types (and not even always statistically significant, with $p = 0.002$, $p = 0.00001$, and $p = 0.1$, respectively), but all the more so when looking at tokens (all $p < 0.000001$). Especially the high frequency types profit by the generalization. The reason is that e-Lex, being based partly on annotated corpus material, contains various erroneous forms. These are of course marked as having a low frequency, but may be selected if we do not apply frequency weighting. The most influential errors are the production of the plural *zijn* instead of *is* as the third person singular present of *zijn* ("to be") for verbs (leading to 10,522 token errors), *kinder* instead of *kinderen* as the plural of *kind* ("child") for nouns (454 token errors) and *ander* instead of *andere* as the *+e* form of *ander* for adjectives and adverbs (1,567 token errors).

In a production system, weighting should certainly be applied. It might even be worthwhile to use more fine-grained weighting, but this would mean either a much larger Timbl case base or a switch to another learning system.

5.3 Word-end Substring Length

As already mentioned above, one of the reasons that ML manages to be as good as, if not better than HYB, is that ML takes into account up to 40 characters from the end of the lemma. For most lemmas in the experiment, this is simply the whole string so that this is in fact equal to what can be found in the lexicon. The question is, therefore, how important the characters at the various points in the lemma are. To measure this, we repeated the experiments for the best scoring ML strategy with word-end substring length gradually decreasing from the original 40 all the way down to 1.

Maximum Word-end Substring Length	Types Nouns	Types Verbs	Types Adjectives/ Adverbs	Tokens Nouns	Tokens Verbs	Tokens Adjectives/ Adverbs
1	88.24	53.89	90.70	89.61	44.52	88.80
2	91.93	54.87	95.48	93.49	49.94	96.62
3	95.60	66.67	97.54	97.18	63.43	98.57
4	96.69	81.06	97.80	98.47	74.42	**99.58**
5	97.16	87.60	97.83	98.88	81.30	99.26
6	97.40	92.26	**97.85**	99.07	94.12	99.27
7	**97.51**	95.15	97.82	99.15	97.45	99.27
8	97.49	96.43	97.81	99.14	98.64	99.26
9	97.46	96.87	97.78	99.17	99.09	99.26
10	97.44	96.95	97.77	99.17	99.15	99.26
11	97.44	96.97	97.78	**99.18**	99.16	99.26
12	97.44	96.97	97.78	**99.18**	99.16	99.26
13	97.44	96.98	97.78	**99.18**	**99.16**	99.26
14	97.44	96.99	97.78	**99.18**	**99.16**	99.26
15	97.44	**97.00**	97.77	**99.18**	**99.16**	99.26
16	97.44	**97.00**	97.77	**99.18**	**99.16**	99.26
17	97.44	**97.00**	97.77	**99.18**	**99.16**	99.26
18	97.44	**97.00**	97.77	**99.18**	**99.16**	99.26
19	97.44	**97.00**	97.77	**99.18**	**99.16**	99.26
20	97.44	**97.00**	97.77	**99.18**	**99.16**	99.26
21	97.44	**97.00**	97.77	**99.18**	**99.16**	99.26
22	97.43	**97.00**	97.77	**99.18**	**99.16**	99.26
23	97.43	**97.00**	97.77	**99.18**	**99.16**	99.26
⋮		⋮			⋮	
40	97.43	**97.00**	97.77	**99.18**	**99.16**	99.26

Table 7: Accuracy Scores for varying Maximum Word-end Substring Length (in bold are shown the highest accuracy, plus all that are not significantly different at the $p < 0.05$ level)

We present the results in Table 7. We see that the full 40 characters are not needed for optimal quality. In fact, we see a slight degradation when moving into the higher ranges of word-end substring length. As more characters become available, the system may choose a different class, using more specific rather than more general scripts. Sometimes this is better but sometimes also not. As the numbers show, from some point on there are more changes for worse than for better, although it must be said that the number of changes involved is extremely low.

The measurements peak at various points, depending on POS as well as on the choice between types and tokens. For Adjectives/Adverbs, around 5 characters of word-end substring appears to be enough. Nouns need more, 7 when looking at types and 11 when looking at tokens. We expect that between 7 and 11 there are both new forms going right and new forms going wrong, but with the ones now going right having higher frequency. For verbs, the peak is clearly reached at 15, with the higher number probably connected to past participle formation.

With the current data, we would tend to vary the word-end substring length for a production system: 5 for Adjectives/Adverbs, 10 for Nouns and 15 for Verbs. However, as we will see below, various improvements can be made to lexicon, learner and evaluation data, and it would be wise to repeat this measurement after these improvements are in place.

5.4 Effects of Normalization

Another choice we made for our experiments is that we normalized all strings by removing diacritics and transforming all letters to lower case (see Section 2.4). In Table 8, we see that this was indeed necessary. Without normalization, the system would not be usable in a production environment. With the current, limited data, delemmatization should be performed on normalized strings and application of diacritics and case should be separate processes. Furthermore, as we will see in Section 6.1, there are a few more processes that might be split off, such as spelling conventions and partial bracketing.

Normalization	Types			Tokens		
	Nouns	Verbs	Adjectives/Adverbs	Nouns	Verbs	Adjectives/Adverbs
not normalized	85.49	89.65	82.76	91.96	97.30	92.27
normalized	97.35	96.80	97.69	99.16	99.14	99.25

Table 8: Accuracy Scores with and without Normalization of Case and Diacritics

6. Error Analysis

In this section, we investigate the errors made by the best system (ML, weighted, IGtree with a fallback to IB1 with k=2, using a maximum word-end substring length of 40). At the end of the section, we will list the errors per POS tag (Section 6.2), but first (Section 6.1) we will classify the types of errors in general, as this will also provide information that is useful for the interpretation of the numbers per tag.

6.1 Error Classes

When comparing the output of the best system with the actual forms in the Lassy Klein corpus, we find 3,817 discrepancies (of 1,564 unique types). Within these discrepancies, we can distinguish several, sometimes rather large classes. We present an overview in Table 9.

Class	Types	Tokens	Tokens/Total Discrep.
Annotation Error	462	1,062	27.8%
Corpus Error	228	253	6.6%
Variants	436	1,543	40.4%
Non-Dutch, Abbreviations	90	314	8.2%
Lexicon Error	198	433	11.3%
ML Error	150	212	5.6%
Total	3,817	1,564	100%

Table 9: System-Corpus Discrepancy Classes

Interestingly, most of the discrepancies are not really errors of our delemmatizer. We find two classes where the gold standard is wrong rather than the system output. In the first of these classes (Annotation Error), the annotation in Lassy Klein is erroneous, either for the tag (e.g. the plural *liefhebbers* ("amateurs") being tagged as singular) or for the lemma (e.g. the noun *Nederlands* being tagged as nominative but then receiving the lemma *Nederland*, which would only be correct if *Nederlands* was interpreted as a genitive). In such cases, it is impossible for the delemmatizer to create the desired form as it either starts with the wrong lemma or attempts to produce the wrong inflection. The second kind of error in the gold standard is a deviating surface form (Corpus Error), mostly typographical and orthographical errors such as *kwamer* instead of *kwamen* as past tense of *komen* ("to come") and *perrimenteren* instead of *permitteren* ("permit").

Similar to the latter class is the class where corpus and system disagree, but they are both right (Variants). The variation may be present in the standard Dutch, e.g. as it allows various plurals (*gemeenten* vs *gemeentes*, "municipalities") or allow more and less formal forms (*wilde* vs *wou*, "wanted"; *goede* vs *goeie*, "good"). But we also find dialectal forms, especially for diminutive formation; the dialect can be present both in the corpus (dialectal *bollekes* vs *bolletjes*, "little balls") and in the prediction (*kapelletje* vs dialectal *kapelleke*, "little chapel"). Then there are orthographical variants. These tend to be present in the corpus (*opge-eist* ("demanded") vs *opgeeist*; *al-qaeda* vs *al-qaida*; *partij(en)* ("party/parties") instead of *partijen*). Finally, a rather special subclass of variants is formed by homographs that show differing inflection. An example is *bedelen* ("beg" or "hand out"), where the former sense (where word stress is on the first syllable) leads to the past tense *bedelde* and the latter (where the word stress is on the second syllable) to *bedeelde*. Here a judgement on the correctness of the delemmatizer would need an inspection of the context. We have not done this, but the number of these cases is extremely low.

Another class where we could decide to be lenient is formed by non-Dutch words and abbreviations. One can hardly expect that a delemmatizer for Dutch knows that the plural of *outlier* is *outliers*; knowing what it knows, it will predict *outlieren*. The same is true for abbreviations, especially when the lemma can be either the full or the abbreviated form. As an example, the corpus form *blz.* is given the lemma *bladzij* ("page"), 28 times with singular number and 66 times with plural number, which leads to the system producing the normal correct forms *bladzij* and *bladzijden*.

Moving on to discrepancies which are caused by system errors, we start with two large classes of systematic errors. By far the largest is the formation of the genitive. Very often, we see that the genitive produced by the system is just the nominative form. A look at the e-Lex lexicon quickly brings to light the cause: 1,733 out of 2,868 genitives in e-Lex, about 60%, are not genitives at all but nominative forms. The machine learner was taught the wrong forms. The same thing happens with the adjective *Engels* ("English"), for which the form with suffix *-en* in e-Lex is *Engelse*, with *-e* rather than *-en*. The system repeats this, leading to 19 discrepancies for *Engels*, but also to 45 discrepancies for *Frans* ("French"). And there are more system errrors that stem from erroneous forms in e-Lex, such as *winkelketen* ("shop chain") getting the singular form as plural instead of *winkelketens*, or the name *schultz* getting the singular form *schiltz*.

However, not all system errors can be explained by blaming the data sets. The biggest subclass concerns plural formation, 113 errors of 69 unique types, most of which by confusion of the two standard Dutch plural formation suffixes *-en* and *-s*, 76 errors of 43 unique types. Another big subclass is past participle formation, 60 errors of 54 unique types. This proved to be partly due to a historical artefact in the software, which can be easily removed. The remaining errors are rather varied, although it is mostly clear that for some reason the wrong edit script was applied. Some improvement could still be reached in those cases where the predicted form contains n-grams unknown in Dutch (e.g. *tegenzittt* with three t's at the end as a 3rd person present tense of *tegenzitten* ("not working smoothly")), so that we can apply a similar fallback as with IGtree to IB1, but these cases are rare.

All in all, only 348 type discrepancies can be confirmed as system errors, and only 150 of these as system-based rather than data-based errors. On the other hand, we have only inspected those tokens where there was a discrepancy between the system output and the corpus. It is possibly, even likely, that there are also tokens where the system does make a mistake, but this is not observed as the corpus form or annotation are also wrong. For now, we can only conclude that the error rate is most probably well under 1%. Furthermore, we can observe that it would be fruitful to clean up both e-Lex and Lassy Klein, in which experiments like ours can obviously be of assistance, analogous to the identification of tagging errors by generating a tagger on the basis of the corpus (van Halteren 2000).

6.2 Errors per POS tag

Apart from the reasons that errors are being made, it is also interesting to see which types of cases go wrong. This lets us decide where effort should be concentrated to solve existing problems or, when the problems cannot be solved, put a warning in the instructions for any future production system that specific problems exist. In Tables 10 to 12, we list the number of errors made per tag. In columns 3 to 5, we show all discrepancies. In columns 6 to 8, we show only those listed as Lexicon Error or ML Error in Section 6.1, i.e. the ones confirmed as actual system errors.

Tag	Tokens	Discreps.	Discreps./ Tokens	Discreps./ All Discreps.	System Errors	Sys.Errs./ Tokens	Sys.Errs./ /All Sys.Errs
N(dim=dim,num=plu,case=nom)	597	18	3.0%	0.9%	1	0.0002%	0.2%
N(dim=dim,num=sing,case=nom)	1,016	31	3.1%	1.5%	-	-	-
N(dim=norm,num=plu,case=nom)	57,866	1,303	2.3%	62.8%	164	0.04%	36.2%
N(dim=norm,num=sing,case=dat)	596	3	0.5%	0.1%	2	0.0004%	0.4%
N(dim=norm,num=sing,case=gen)	1,860	388	20.9%	18.7%	279	0.06%	61.6%
N(dim=norm,num=sing,case=nom)	184,830	333	0.2%	16.0%	7	0.002%	1.5%
Total N	246,765	2,076	0.8%	100%	453	0.2%	100%

Table 10: Errors per noun tag for best ML system

Tag	Tokens	Discreps.	Discreps./ Tokens	Discreps./ All Discreps.	System Errors	Sys.Errs./ Tokens	Sys.Errs./ /All Sys.Errs
V(fin=fin,tense=past,num=plu,form=norm)	7,176	41	0.6%	3.4%	6	0.001%	5.9%
V(fin=fin,tense=past,num=sing,form=norm)	20,585	131	0.6%	10.8%	5	0.001%	5.5%
V(fin=fin,tense=pres,num=plu,form=norm)	16,893	26	0.2%	2.1%	-	-	-
V(fin=fin,tense=pres,num=sing,form=norm)	19,853	269	1.4%	22.1%	1	0.0002%	1.0%
V(fin=fin,tense=pres,num=sing,form=t)	22,286	233	1.1%	19.1%	15	0.003%	14.9%
V(fin=infin,form=norm)	25,659	100	0.4%	8.2%	-	-	-
V(fin=part,tense=past,form=e)	2,908	51	1.8%	4.2%	22	0.005%	21.8%
V(fin=part,tense=past,form=en)	262	14	5.3%	1.2%	7	0.002%	6.9%
V(fin=part,tense=past,form=norm)	23,397	257	1.1%	21.1%	29	0.006%	28.7%
V(fin=part,tense=pres,form=e)	2,263	18	0.8%	1.5%	-	-	-
V(fin=part,tense=pres,form=en)	32	1	3.1%	0.1%	1	0.0002%	1.0%
V(fin=part,tense=pres,form=norm)	980	76	7.8%	6.2%	15	0.003%	14.9%
Total V	142,294	1,217	0.9%	100%	101	0.07%	100%

Table 11: Errors per verb tag for best ML system

As already seen in Section 6.1, there are a few tags with larger numbers of errors, for which we already found partial solutions, mostly consisting of lexicon cleanup. The other errors are spread out over the remaining tags and will be harder to correct.

Tag	Tokens	Discreps.	Discreps./ Tokens	Discreps./ All Discreps.	System Errors	Sys.Errs./ Tokens	Sys.Errs./ /All Sys.Errs
Adjv(deg=comp,form=e)	1,030	3	0.3%	0.5%	-	-	-
Adjv(deg=comp,form=en)	258	-	-	-	-	-	-
Adjv(deg=comp,form=norm)	2,809	7	0.3%	1.2%	2	0.0004%	2.2%
Adjv(deg=pos,form=e)	40,815	408	1.0%	67.8%	10	0.002%	11.0%
Adjv(deg=pos,form=en)	880	71	8.1%	11.8%	65	0.01%	71.4%
Adjv(deg=pos,form=norm)	32,358	86	0.3%	14.3%	14	0.003%	15.4%
Adjv(deg=sup,form=e)	2,015	24	1.2%	4.0%	-	-	-
Adjv(deg=sup,form=norm)	233	3	1.3%	0.5%	-	-	-
Total Adjv	80,398	602	0.7%	100%	91	0.1%	100%

Table 12: Errors per adjective/adverb tag for best ML system

7. Conclusion and Future Work

In this paper we have investigated whether, for Dutch open class words, it is possible to generate the surface form on the basis of the lemma and a POS tag. We have tested various strategies, combining a lexicon (e-Lex) and a machine learning system (Timbl), against an annotated corpus (Lassy Klein). We found that for all three parts of speech tested, nouns, verbs, and adjectives/adverbs, the best strategy has an error rate well under 1% when comparing the system output against the form in the corpus. Furthermore, when examining the errors, we find that only one in six discrepancies between system output and corpus could really be counted as a system error. The other discrepancies are due to errors in the annotation of the corpus, erroneous surface forms in the corpus, different choices from possible variants rather than errors, or the fact that the words in question are not subject to normal Dutch morphology. On the other hand, there may well be many other system errors which we did not find because of similar errors in annotation and/or corpus forms. This means that an exact accuracy score cannot be given, but in all probability the error rate is closer to 0% than to 1%.

We find that the pure lexicon strategy is not viable, seeing the unavoidable lack of coverage of the lexicon. The two other strategies, involving a machine learning system, lead to equally good results in the current experiments. For the words known to the lexicon, we can use either the lexicon or the machine learner. Both make more or less the same number of mistakes. However, looking something up in a lexicon cannot be improved upon, but a machine learning technique can, so that we tend to prefer the pure machine learning strategy over the hybrid which first checks the lexicon entries. For the machine learning strategy with Timbl, the best results are found when using IGtree with a fallback to IB1 with k=2 in those cases where IGtree suggests an impossible edit script for transforming the lemma to the form. In addition, all lemmas and forms should be normalized by removing any diacritics and transforming the words to lower case. If desired, a separate module could reinstate diacritics and upper case on the suggested form. The optimal maximum word-end substring length appears to vary between the word classes, with around 5 for adjectives/adverbs, 10 for nouns, and 15 for verbs. Finally, the corpus frequencies of the various forms for the same lemma-tag pair should be taken into account when learning, by some kind of weighting technique.

Although delemmatization errors have also been found in the learning mechanism, most errors by far have been caused by noise in the e-Lex lexicon. It would seem wise to first clean up the lexicon before attempting to understand and fix the remaining learning technique errors. After all, at the moment we can only recognize the errors caused directly by lexicon noise, and not the errors that have been caused by generalizing from that noise. However, we have already seen that even with a perfect lexicon, the system would still make mistakes and we should consider how to adapt our learning method so that these errors no longer occur.

Apart from improving the accuracy for the open classes, we also need to add the processing of closed class words in order to build a full production system. Once that is in place, we can start considering actual applications. One application is already eagerly waiting, namely the generation of modern word forms for 14th Century Dutch text. This work can start with the annotated corpus van Reenen-Mulder (Rem 2003) and, if this is succesful, by creating a pipeline where as yet unannotated 14th Century Dutch text is tagged-lemmatized with Adelheid (van Halteren and Rem 2013) and then extended with modern word forms by way of delemmatization.

References

Baayen, Harald R., Richard Piepenbrock, and Leon Gulikers (1995), *The CELEX Lexical Database. Release 2 (CD-ROM)*, Linguistic Data Consortium, University of Pennsylvania, Philadelphia, Pennsylvania.

Beesley, Ken (2004), Morphological analysis and generation: a first step in natural language processing, *Proc. of the SALTMIL Workshop at LREC 2004*, pp. 1–8.

Beesley, Kenneth R. and Lauri Karttunen (2003), Finite-state morphology: Xerox tools and techniques, *CSLI, Stanford.*

Booij, Geert Evert and Ariane Van Santen (1998), *Morfologie: de woordstructuur van het Nederlands*, Vol. 2e geheel, Amsterdam University Press.

Bouma, Gosse, Gertjan van Noord, and Robert Malouf (2001), Alpino: Wide-coverage computational analysis of Dutch, *Language and Computers* **37** (1), pp. 45–59.

Carl, M., P. Schmidt, , and J. Schütz (2005), Reversible template-based shake & bake generation, *Proceedings of the Example-Based Machine Translation Workshop held in conjunction with the 10th Machine Translation Summit, Phuket, Thailand, September 16*, pp. 17–26.

Daelemans, Walter, Antal van Den Bosch, and Jakub Zavrel (1999), Forgetting exceptions is harmful in language learning, *Machine Learning* **34** (1-3), pp. 11–41, Springer.

Daelemans, Walter, Antal van Den Bosch, and Ton Weijters (1997), Igtree: Using trees for compression and classification in lazy learning algorithms, *Artificial Intelligence Review* **11** (1-5), pp. 407–423, Springer.

Daelemans, Walter, Jakub Zavrel, Ko van der Sloot, and Antal Van den Bosch (2004), Timbl: Tilburg memory-based learner, *Technical Report ILK-0209*, Tilburg University.

Oostdijk, Nelleke (2000), The spoken dutch corpus: overview and first evaluation, *Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC)*, Vol. 2, pp. 887–894.

Rem, Margit (2003), *De taal van de klerken uit de Hollandse grafelijke kanselarij (1300-1340): naar een lokaliseringsprocedure voor het veertiende-eeuws Middelnederlands*, PhD thesis, Vrije Universiteit, Amsterdam.

van den Bosch, Antal, Bertjan Busser, Sander Canisius, and Walter Daelemans (2007), An efficient memory-based morphosyntactic tagger and parser for Dutch, *Computational linguistics in the Netherlands: Selected papers from the Seventeenth CLIN Meeting*, pp. 99–114.

Van Eynde, Frank, Jakub Zavrel, and Walter Daelemans (2000), Part of speech tagging and lemmatisation for the spoken dutch corpus, *Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC)*, Vol. 3, pp. 1427–1433.

van Halteren, Hans (2000), The detection of inconsistency in manually tagged text, *Proceedings of the COLING-2000 Workshop on Linguistically Interpreted Corpora*, International Committee on Computational Linguistics, Centre Universitaire, Luxembourg, pp. 48–55.

van Halteren, Hans and Margit Rem (2013), Dealing with orthographic variation in a tagger-lemmatizer for fourteenth century dutch charters, *Language Resources and Evaluation*, Springer.

Van Noord, Gertjan (2006), At Last Parsing Is Now Operational, *TALN06. Verbum Ex Machina. Actes de la 13e conference sur le traitement automatique des langues naturelles*, pp. 20–42.

Van Noord, Gertjan, Ineke Schuurman, and Vincent Vandeghinste (2006), Syntactic annotation of large corpora in STEVIN, *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC).*

Vaneyghen, Joris, Guy De Pauw, Dirk Van Compernolle, and Walter Daelemans (2006), A mixed word / morphological approach for extending celex for high coverage on contemporary large corpora, *Proc. 5th International Conference on Language Resources and Evaluation, Genoa, Italy, May 2006*, pp. 931–934.