Improved morpho-phonological sequence processing with constraint satisfaction inference

Antal van den Bosch and **Sander Canisius**
ILK / Language and Information Science
Tilburg University, P.O. Box 90153, NL-5000 LE Tilburg, The Netherlands
{Antal.vdnBosch,S.V.M.Canisius}@uvt.nl

Abstract

In performing morpho-phonological sequence processing tasks, such as letter-phoneme conversion or morphological analysis, it is typically not enough to base the output sequence on local decisions that map local-context input windows to single output tokens. We present a global sequence-processing method that repairs inconsistent local decisions. The approach is based on local predictions of overlapping trigrams of output tokens, which open up a space of possible sequences; a data-driven constraint satisfaction inference step then searches for the optimal output sequence. We demonstrate significant improvements in terms of word accuracy on English and Dutch letter-phoneme conversion and morphological segmentation, and we provide qualitative analyses of error types prevented by the constraint satisfaction inference method.

1 Introduction

The fields of computational phonology and morphology were among the earlier fields in computational linguistics to adopt machine learning algorithms as a means to automatically construct processing systems from data. For instance, letter-phoneme conversion was already pioneered, with neural networks initially, at the end of the 1980s (Sejnowski and Rosenberg, 1987), and was shortly after also investigated with memory-based learning and analogical approaches (Weijters, 1991; Van den Bosch and Daelemans, 1993; Yvon, 1996) and decision trees (Torkkola, 1993; Dietterich et al., 1995). The development of these data-driven systems was thrusted by the early existence of lexical databases, originally compiled to serve (psycho)linguistic research purposes, such as the CELEX lexical database for Dutch, English, and German (Baayen et al., 1993). Many researchers have continued and are still continuing this line of work, generally producing successful systems with satisfactory, though still imperfect performance.

A key characteristic of many of these early systems is that they perform decomposed or simplified versions of the full task. Rather than predicting the full phonemization of a word given its orthography in one go, the task is decomposed in predicting individual phonemes or subsequences of phonemes. Analogously, rather than generating a full word-form, many morphological generation systems produce transformation codes (e.g., "add -er and umlaut") that need to be applied to the input string by a post-processing automaton. These task simplifications are deliberately chosen to avoid sparseness problems to the machine learning systems. Such systems tend to perform badly when there are many low-frequent and too case-specific classes; task decomposition allows them to be robust and generic when they process unseen words.

This task decomposition strategy has a severe drawback in sequence processing tasks. Decomposed systems do not have any global method to check whether their local decisions form a globally

Proceedings of the Eighth Meeting of the ACL Special Interest Group on Computational Phonology at HLT-NAACL 2006, pages 41–49,
New York City, USA, June 2006.

coherent output. If a letter-phoneme conversion system predicts schwas on every vowel in a polysyllabic word such as *parameter* because it is uncertain about the ambiguous mapping of each of the *a*s and *e*s, it produces a bad pronunciation. Likewise, if a morphological analysis system segments a word such as *being* as a prefix followed by an inflection, making the locally most likely guesses, it generates an analysis that could never exist, since it lacks a stem.

Global models that coordinate, mediate, or enforce that the output is a valid sequence are typically formulated in the form of linguistic rules, applied during processing or in post-processing, that constrain the space of possible output sequences. Some present-day research in machine learning of morpho-phonology indeed focuses on satisfying linguistically-motivated constraints as a post-processing or filtering step; e.g., see (Daya et al., 2004) on identifying roots in Hebrew word forms. Optimality Theory (Prince and Smolensky, 2004) can also be seen as a constraint-based approach to language processing based on linguistically motivated constraints. In contrast to being motivated by linguistic theory, constraints in a global model can be learned automatically from data as well. In this paper we propose such a data-driven constraint satisfaction inference method, that finds a globally appropriate output sequence on the basis of a space of possible sequences generated by a locally-operating classifier predicting output subsequences. We show that the method significantly improves on the basic method of predicting single output tokens at a time, on English and Dutch letter-phoneme conversion and morphological analysis.

This paper is structured as follows. The constraint satisfaction inference method is outlined in Section 2. We describe the four morpho-phonological processing tasks, and the lexical data from which we extracted examples for these tasks, in Section 3. We subsequently list the outcomes of the experiments in Section 4, and conclude with a discussion of our findings in Section 5.

2 Class trigrams and constraint satisfaction inference

Both the letter-phoneme conversion and the morphological analysis tasks treated in this paper can be seen as sequentially-structured classification tasks, where sequences of letters are mapped to sequences of phonemes or morphemes. Such sequence-to-sequence mappings are a frequently reoccurring phenomenon in natural language processing, which suggests that it is preferable to take care of the issue of classifying sequential data once at the machine learning level, rather than repeatedly and in different ways at the level of practical applications. Recently, a machine learning approach for sequential data has been proposed by Van den Bosch and Daelemans (2005) that is suited for discrete machine-learning algorithms such as memory-based learners, which have been shown to perform well on word phonemization and morphological analysis before (Van den Bosch and Daelemans, 1993; Van den Bosch and Daelemans, 1999). In the remainder of this paper, we use as our classifier of choice the IB1 algorithm (Aha et al., 1991) with feature weighting, as implemented in the TiMBL software package[1] (Daelemans et al., 2004).

In the approach to sequence processing proposed by Van den Bosch and Daelemans (2005), the elements of the input sequence (in the remainder of this paper, we will refer to words and letters rather than the more general terms sequences and sequence elements) are assigned overlapping subsequences of output symbols. This subsequence corresponds to the output symbols for a *focus* letter, and one letter to its left and one letter to its right. Predicting such trigram subsequences for each letter of a word eventually results in three output symbol predictions for each letter. In many cases, those three predictions will not agree, resulting in a number of potential output sequences. We will refer to the procedure for selecting the final output sequence from the space of alternatives spanned by the predicted trigrams as an inference procedure, analogously to the use of this term in probabilistic sequence classification methods (Punyakanok and Roth, 2001). The original work on predicting class trigrams implemented a simple inference procedure by voting over the three predicted symbols (Van den Bosch and Daelemans, 2005).

Predicting trigrams of overlapping output symbols has been shown to be an effective approach

[1] TiMBL URL: http://ilk.uvt.nl/timbl/

to improve sequence-oriented natural language processing tasks such as syntactic chunking and named-entity recognition, where an input sequence of tokens is mapped to an output sequence of symbols encoding a syntactic or semantic segmentation of the sentence. Letter-phoneme conversion and morphological analysis, though sequentially structured on another linguistic level, may be susceptible to benefiting from this approach as well.

In addition to the practical improvement shown to be obtained with the class trigram method, there is also a more theoretical attractiveness to it. Since the overlapping trigrams that are predicted are just atomic symbols to the underlying learning algorithm, a classifier will only predict output symbol trigrams that are actually present in the data it was trained on. Consequently, predicted trigrams are guaranteed to be syntactically valid subsequences in the target task. There is no such guarantee in approaches to sequence classification where an isolated local classifier predicts single output symbols at a time, without taking into account predictions made elsewhere in the word.

While the original voting-based inference procedure proposed by Van den Bosch and Daelemans (2005) manages to exploit the sequential information stored in the predicted trigrams to improve upon the performance of approaches that do not consider the sequential structure of their output at all, it does so only partly. Essentially, the voting-based inference procedure just splits the overlapping trigrams into their unigram components, thereby retaining only the overlapping symbols for each individual letter. As a result, the guaranteed validity of the trigram subsequences is not put to use. In this section we describe an alternative inference procedure, based on principles of constraint satisfaction, that does manage to use the sequential information provided by the trigram predictions.

At the foundation of this constraint-satisfaction-based inference procedure, more briefly constraint satisfaction inference, is the assumption that the output symbol sequence should preferably be constructed by concatenating the predicted trigrams of output symbols, rather than by chaining individual symbols. However, as the underlying base classifier is by no means perfect, predicted trigrams should not be copied blindly to the output sequence; they may be incorrect. If a trigram prediction is considered to be of insufficient quality, the procedure backs off to symbol bigrams or even symbol unigrams.

The intuitive description of the inference procedure is formalized by expressing it as a weighted constraint satisfaction problem (W-CSP). Constraint satisfaction is a well-studied research area with many diverse areas of application. Weighted constraint satisfaction extends the traditional constraint satisfaction framework with soft constraints; such constraints are not required to be satisfied for a solution to be valid, but constraints a given solution does satisfy are rewarded according to weights assigned to them. Soft constraints are perfect for expressing our preference for symbol trigrams, with the possibility of a back off to lower-degree n-grams if there is reason to doubt the quality of the trigram predictions.

Formally, a W-CSP is a tuple (X, D, C, W). Here, $X = \{x_1, x_2, \ldots, x_n\}$ is a finite set of variables. $D(x)$ is a function that maps each variable to its domain, that is, the set of values that variable can take on. C is the set of constraints. While a variable's domain dictates the values a single variable is allowed to take on, a constraint specifies which simultaneous value *combinations* over a number of variables are allowed. For a traditional (non-weighted) constraint satisfaction problem, a valid solution would be an assignment of values to the variables that (1) are a member of the corresponding variable's domain, and (2) satisfy *all* constraints in the set C. Weighted constraint satisfaction, however, relaxes this requirement to satisfy all constraints. Instead, constraints are assigned weights that may be interpreted as reflecting the importance of satisfying that constraint.

Let a constraint $c \in C$ be defined as a function that maps each variable assignment to 1 if the constraint is satisfied, or to 0 if it is not. In addition, let $W\colon C \rightarrow \mathbb{R}^+$ denote a function that maps each constraint to a positive real value, reflecting the weight of that constraint. Then, the optimal solution to a W-CSP is given by the following equation.

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} \sum_c W(c)c(\mathbf{x})$$

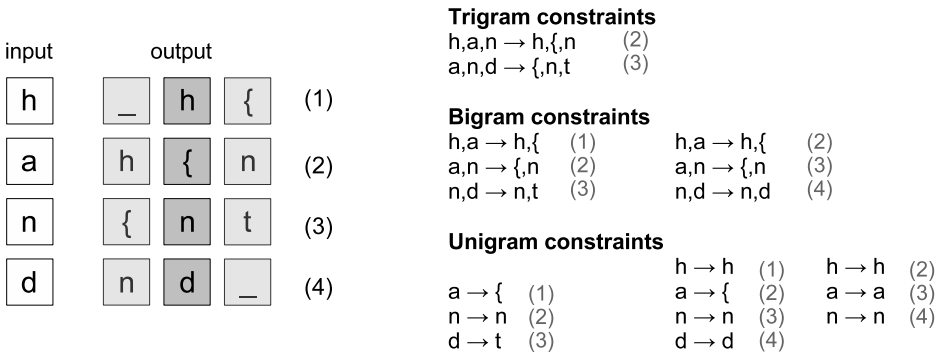

Figure 1: Illustration of the constraints yielded by a given sequence of predicted class trigrams for the word *hand*. The constraints on the right have been marked with a number (between parentheses) that refers to the trigram prediction on the left from which the constraint was derived.

That is, the assignment of values to its variables that maximizes the sum of weights of the constraints that have been satisfied.

Translating the terminology used in morpho-phonological tasks to the constraint satisfaction domain, each letter maps to a variable, the domain of which corresponds to the three overlapping candidate symbols for this letter suggested by the trigrams covering the letter. This provides us with a definition of the function D, mapping variables to their domain. In the following, $y_{i,j}$ denotes the candidate symbol for letter x_j predicted by the trigram assigned to letter x_i.

$$D(x_i) = \{y_{i-1,i}, y_{i,i}, y_{i+1,i}\}$$

Constraints are extracted from the predicted trigrams. Given the goal of retaining predicted trigrams in the output symbol sequence as much as possible, the most important constraints are simply the trigrams themselves. A predicted trigram describes a subsequence of length three of the entire output sequence; by turning such a trigram into a constraint, we express the wish to have this trigram end up in the final output sequence.

$$(x_{i-1}, x_i, x_{i+1}) = (y_{i,i-1}, y_{i,i}, y_{i,i+1}), \forall i$$

No base classifier is flawless though, and therefore not all predicted trigrams can be expected to be correct. Yet, even an incorrect trigram may carry some useful information regarding the output sequence: one trigram also covers two bigrams, and three unigrams. An incorrect trigram may still contain smaller subsequences of length one or two that are correct. Therefore, all of these are also mapped to constraints.

$$\begin{aligned}(x_{i-1}, x_i) &= (y_{i,i-1}, y_{i,i}), && \forall i\\ (x_i, x_{i+1}) &= (y_{i,i}, y_{i,i+1}), && \forall i\end{aligned}$$

$$\begin{aligned}x_{i-1} &= y_{i,i-1}, && \forall i\\ x_i &= y_{i,i}, && \forall i\\ x_{i+1} &= y_{i,i+1}, && \forall i\end{aligned}$$

To illustrate the above procedure, Figure 1 shows the constraints yielded by a given output sequence of class trigrams for the word “hand”. With such an amount of overlapping constraints, the satisfaction problem obtained easily becomes over-constrained, that is, no variable assignment exists that can satisfy all constraints without breaking another. Even only one incorrectly predicted class trigram already leads to two conflicting candidate symbols for one of the letters at least. In Figure 1, this is the case for the letter “d”, for which both the symbol “d” and “t” are predicted. On the other hand, without conflicting candidate symbols, no inference would be needed to start with. The choice for the weighted constraint satisfaction method always allows a solution to be found, even in the presence of conflicting constraints. Rather than requiring all constraints to be satisfied, each constraint is assigned a certain weight; the optimal solution to the problem is an assignment of values to the variables that optimizes the

Left context			Focus letter	Right context			Trigram output classes	
							Phonemization	Morph. analysis
_	_	_	b	o	o	k	_ b u	_ s -
_	_	b	o	o	k	i	b u -	s - -
_	b	o	o	k	i	n	u - k	- - -
b	o	o	k	i	n	g	- k I	- - i
o	o	k	i	n	g	_	k I N	- i -
o	k	i	n	g	_	_	I N -	i - -
k	i	n	g	_	_	_	N - _	- - _

Table 1: Seven labeled examples of phonemization and morphological analysis trigram mappings created for the word *booking*.

sum of weights of the constraints that are satisfied.

As weighted constraints are defined over overlapping subsequences of the output sequence, the best symbol assignment for each letter with respect to the weights of satisfied constraints is decided upon on a global sequence level. This may imply taking into account symbol assignments for surrounding letters to select the best output symbol for a certain letter. In contrast, in non-global approaches, ignorant of any sequential context, only the local classifier prediction with highest confidence is considered for selecting a letter's output symbol. By formulating our inference procedure as a constraint satisfaction problem, global output optimization comes for free: in constraint satisfaction, the aim is also to find a globally optimal assignment of variables taking into account all constraints defined over them. Yet, for such a constraint satisfaction formulation to be effective, good constraint weights should be chosen, that is, weights that favor good output sequences over bad ones.

Constraints can directly be traced back to a prediction made by the base classifier. If two constraints are in conflict, the one which the classifier was most certain of should preferably be satisfied. In the W-CSP framework, this preference can be expressed by weighting constraints according to the classifier confidence for the originating trigram. For the memory-based learner, we define the classifier confidence for a predicted class as the weight assigned to that class in the neighborhood of the test instance, divided by the total weight of all classes.

Let x denote a test instance, and c^* its predicted class. Constraints derived from this class are weighted according to the following rules:

- for a trigram constraint, the weight is simply the base classifier's confidence value for the class c^*;
- for a bigram constraint, the weight is the sum of the confidences for all trigram classes in the nearest-neighbor set of x that assign the same symbol bigram to the letters spanned by the constraint;
- for a unigram constraint, the weight is the sum of the confidences for all trigram classes in the nearest-neighbor set of x that assign the same symbol to the letter spanned by the constraint.

This weighting scheme results in an inference procedure that behaves exactly as we already described intuitively in the beginning of this section. The preference for retaining the predicted trigrams in the output sequence is translated into high rewards for output sequences that do so, since such output sequences not only receive credit for the satisfied trigram constraints, but also for all the bigram and unigram constraints derived from that trigram; they are necessarily satisfied as well. Nonetheless, this preference for trigrams may be abandoned if composing a certain part of the output sequence from several symbol bigrams or even unigrams results in higher rewards than when trigrams are used. The latter may happen in cases where the base classifier is not confident about its trigram predictions.

3 Data preparation

In our experiments we train classifiers on English and Dutch letter-phoneme conversion and morphological analysis. All data for the experiments described in this paper are extracted from the CELEX lexical databases for English and Dutch (Baayen et al., 1993). We encode the examples for our base classifiers in a uniform way, along the following procedure. Given a word and (i) an aligned phonemic transcription or (ii) an aligned encoding of a morphological analysis, we generate letter-by-letter windows. Each window takes one letter in focus, and includes three neighboring letters to the left and to the right. Each seven-letter input window is associated to a trigram class label, composed of the focus class label aligned with the middle letter, plus its immediately preceding and following class labels. Table 1 displays the seven examples made on the basis of the word *booking*, with trigram classes (as explained in Section 2) both for the letter-phoneme conversion task and for the morphological analysis task. The full aligned phonemic transcription of *booking* is [bu-kIN-] (using the SAMPA coding of the international phonetic alphabet), and the morphological analysis of *booking* is [*book*]$_{stem}$[ing]$_{inflection}$. The dashes in the phonemic transcription are inserted to ensure a one-to-one mapping between letters and phonemes; the insertion was done by automatical alignment through expectation-maximization (Dempster et al., 1977).

The English word phonemization data, extracted from the CELEX lexical database, contains 65,467 words, on the basis of which we create a database of 573,170 examples. The Dutch word phonemization data set consists of 293,825 words, totaling to 3,181,345 examples. Both data sets were aligned using the expectation-maximization algorithm (Dempster et al., 1977), using a phonemic null character to equalize the number of symbols in cases in which the phonemic transcription is shorter than the orthographic word, and using "double phonemes" (e.g. [X] for [ks]) in cases where the phonemic transcription is longer, as in *taxi* – [tAksi].

CELEX contains 336,698 morphological analyses of Dutch (which we converted to 3,209,090 examples), and 65,558 analyses of English words (573,544 examples). We converted the available morphological information for the two languages in a coding scheme which is rather straightforward in the case of English, and somewhat more complicated for Dutch. For English, as exemplified in Table 1, a simple segmentation label marks the beginning of either a stem, an inflection ("s" and "i" in Table 1), a stress-affecting affix, or a stress-neutral affix ("1" and "2", not shown in Table 1). The coding scheme for Dutch incorporates additional information on the part-of-speech of every stem and non-inflectional affix, the type of inflection, and also encodes all spelling changes between the base lemma forms and the surface word form.

Left context			Focus letter	Right context			Trigram class
_	_	_	a	b	n	o	_ A 0
_	_	a	b	n	o	r	A 0 0
_	a	b	n	o	r	m	0 0 0
a	b	n	o	r	m	a	0 0 0
b	n	o	r	m	a	l	0 0 0
n	o	r	m	a	l	i	0 0 0
o	r	m	a	l	i	t	0 0 0+Da
r	m	a	l	i	t	e	0 0+Da A_→N
m	a	l	i	t	e	i	0+Da A_→N 0
a	l	i	t	e	i	t	A_→N 0 0
l	i	t	e	i	t	e	0 0 0
i	t	e	i	t	e	n	0 0 0
t	e	i	t	e	n	_	0 0 plural
e	i	t	e	n	_	_	0 plural 0
i	t	e	n	_	_	_	plural 0 _

Table 2: Examples with morphological analysis trigram classes derived from the example word *abnormaliteiten*.

To illustrate the more complicated construction of examples for Dutch morphological analysis, Table 2 displays the 15 instances derived from the Dutch example word *abnormaliteiten* (abnormalities) and their associated classes. The class of the first instance is A, which signifies that the morpheme starting in *a* is an adjective (A). The class of the eighth instance, 0+Da, indicates that at that position no segment starts (0), but that an *a* was deleted at that position (+Da, "delete a" here). Next to deletions, insertions (+I) and replacements (+R, with a deletion and an insertion argument) can also occur. Together

Language	Task	Unigrams	Trigrams
English	Letter-phon.	58	13,005
	Morphology	5	80
Dutch	Letter-phon.	201	17,538
	Morphology	3,831	14,795

Table 3: Numbers of unigram and trigram classes for the four tasks.

these two classification labels code that the first morpheme is the adjective *abnormaal*. The second morpheme, the suffix *iteit*, has class A_→N. This complex tag, which is in fact a rewrite rule, indicates that when *iteit* attaches right to an adjective (encoded by A_), the new combination becomes a noun (→N). Rewrite rule class labels occur exclusively with suffixes, that do not have a part-of-speech tag of their own, but rather seek an attachment to form a complex morpheme with the part-of-speech tag. Finally, the third morpheme is *en*, which is a plural inflection that by definition attaches to a noun.

Logically, the number of trigram classes for each task is larger than the number of atomic classes; the actual numbers for the four tasks investigated here are displayed in Table 3. The English morphological analysis task has the lowest number of trigram classes, 80, due to the fact that there are only five atomic classes in the original task, but for the other tasks the number of trigram classes is quite high; above 10,000. With these numbers of classes, several machine learning algorithms are practically ruled out, given their high sensitivity to numbers of classes (e.g., support vector machines or rule learners). Memory-based learning algorithms, however, are among a small set of machine learning algorithms that are insensitive to the number of classes both in learning and in classification.

4 Results

We performed experiments with the memory-based learning algorithm IB1, equipped with constraint satisfaction inference post-processing, on the four aforementioned tasks. In one variant, IB1 was simply used to predict atomic classes, while in the other variant IB1 predicted trigram classes, and constraint satisfaction inference was used for post-processing the output sequences. We chose to measure the generalization performance of our trained classifiers on a single 90% training set – 10% test set split of each data set (after shuffling the data randomly at the word level), and measuring the percentage of fully correctly phonemized words or fully correctly morphologically analyzed words – arguably the most critical and unbiased performance metric for both tasks. Additionally we performed bootstrap resampling (Noreen, 1989) to obtain confidence intervals.

Language	Method	Word accuracy
English	Unigram	80.0 ± 0.75
	CSInf	85.4 ± 0.71
Dutch	Unigram	41.3 ± 0.48
	CSInf	51.9 ± 0.48

Table 4: Word accuracies on English and Dutch morphological analysis by the default unigram classifier and the trigram method with constraint satisfaction inference, with confidence intervals.

Language	Method	Word accuracy
English	Unigram	79.0 ± 0.82
	CSInf	84.5 ± 0.76
Dutch	Unigram	92.8 ± 0.25
	CSInf	94.4 ± 0.22

Table 5: Word accuracies on English and Dutch letter-phoneme conversion by the default unigram classifier and the trigram method with constraint satisfaction inference, with confidence intervals.

Table 4 lists the word accuracies obtained on the English and Dutch morphological analysis tasks. Constraint satisfaction inference significantly outperforms the systems that predict atomic unigram classes, by a large margin. While the absolute differences in scores between the two variants of English morphological analysis is 5.4%, the error reduction is an impressive 27%.

Table 5 displays the word phonemization accuracies of both variants on both languages. Again, significant improvements over the baseline classifier can be observed; the confidence intervals are widely apart. Error reductions for both languages are impressive: 26% for English, and 22% for Dutch.

5 Discussion

We have presented constraint satisfaction inference as a global method to repair errors made by a local classifier. This classifier is a memory-based learner predicting overlapping trigrams, creating a space of possible output sequences in which the inference procedure finds the globally optimal one. This globally optimal sequence is the one that adheres best to the trigram, bigram, and unigram sub-sequence constraints present in the predictions of the local classifier, weighted by the confidences of the classifier, in a back-off order from trigrams to unigrams.

The method is shown to significantly outperform a memory-based classifier predicting atomic classes and lacking any global post-processing, which has previously been shown to exhibit successful performance (Van den Bosch and Daelemans, 1993; Van den Bosch and Daelemans, 1999). (While this was the reason for using memory-based learning, we note that the constraint satisfaction inference and its underlying trigram-based classification method can be applied to any machine-learning classifier.) The large improvements (27% and 26% error reductions on the two English tasks, 18% and 22% on the two Dutch tasks) can arguably be taken as an indication that this method may be quite effective in general in morpho-phonological sequence processing tasks.

Apparently, the constraint-satisfaction method is able to avoid more errors than to add them. At closer inspection, comparing cases in which the atomic classifier generates errors and constraint satisfaction inference does not, we find that the type of avoided error, when compared to the unigram classifier, differs per task. On the morphological analysis task, we identify repairs where (1) a correct segmentation is inserted, (2) a false segmentation is not placed, and (3) a tag is switched. As Table 6 shows in its upper four lines, in the case of English most repairs involve correctly inserted segmentations, but the other two categories are also quite frequent. In the case of Dutch the most common repair is a switch from an incorrect tag, placed at the right segmentation position, to the correct tag at that point. Given that there are over three thousand possible tags in our complicated Dutch morphological analysis task, this is indeed a likely area where there is room for improvement.

Morphological analysis repairs	English	Dutch
Insert segmentation	193	1,087
Delete segmentation	158	1,083
Switch tag	138	2,505
Letter-phoneme repairs	**English**	**Dutch**
Alignment	1,049	239
Correct vowel	32	94
Correct consonant	275	73

Table 6: Numbers of repaired errors divided over three categories of morphological analysis classifications (top) and letter-phoneme conversions (bottom) of the constraint satisfaction inference method as compared to the unigram classifier.

The bottom four lines of Table 6 lists the counts of repaired errors in word phonemization in both languages, where we distinguish between (1) alignment repairs between phonemes and alignment symbols (where phonemes are corrected to phonemic nulls, or vice versa), (2) switches from incorrect non-null phonemes to correct vowels, and (3) switches from incorrect non-null phonemes to correct consonants. Contrary to expectation, it is not the second vowel category in which most repairs are made (many of the vowel errors in fact remain in the output), but the alignment category, in both languages. At points where the local unigram classifier sometimes incorrectly predicts a phoneme twice, where it should have predicted it along with a phonemic null, the constraint satisfaction inference method never generates a double phoneme. Hence, the method succeeds in generating sequences that are *possible*, and avoiding impossible sub-sequences. At the same time, a *possible* sequence is not necessarily the *correct* sequence, so this method can be expected to still make errors on the identity of labels in the output sequence.

In future work we plan to test a range of n-gram widths exceeding the current trigrams. Preliminary results suggest that the method retains a positive effect over the baseline with $n > 3$, but it does not outperform the $n = 3$ case. We also intend to test the method with a range of different machine learning methods, since as we noted before the constraint-satisfaction inference method and its underlying n-gram output subsequence classification method can

be applied to any machine learning classification algorithm in principle, as is already supported by preliminary work in this direction.

Also, we plan comparisons to the work of Stroppa and Yvon (2005) and Damper and Eastmond (1997) on sequence-global analogy-based models for morpho-phonological processing, since the main difference between this related work and ours is that both alternatives are based on working units of variable width, rather than our fixed-width n-grams, and also their analogical reasoning is based on interestingly different principles than our k-nearest neighbor classification rule, such as the use of analogical proportions by Stroppa and Yvon (2005).

Acknowledgements

This research was funded by NWO, the Netherlands Organization for Scientific Research, as part of the *IMIX* Programme. The authors would like to thank Walter Daelemans for fruitful discussions, and three anonymous reviewers for their insightful comments.

References

D. W. Aha, D. Kibler, and M. Albert. 1991. Instance-based learning algorithms. *Machine Learning*, 6:37–66.

R. H. Baayen, R. Piepenbrock, and H. van Rijn. 1993. *The CELEX lexical data base on CD-ROM*. Linguistic Data Consortium, Philadelphia, PA.

W. Daelemans, J. Zavrel, K. Van der Sloot, and A. Van den Bosch. 2004. TiMBL: Tilburg memory based learner, version 5.1.0, reference guide. Technical Report ILK 04-02, ILK Research Group, Tilburg University.

R. I. Damper and J. F. G. Eastmond. 1997. Pronunciation by analogy: impact of implementational choices on performance. *Language and Speech*, 40:1–23.

E. Daya, D. Roth, and S. Wintner. 2004. Learning Hebrew roots: Machine learning with linguistic constraints. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 357–364, Barcelona, Spain, July. Association for Computational Linguistics.

A.P. Dempster, N.M. Laird, and D.B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B (Methodological)*, 39(1):1–38.

T. G. Dietterich, H. Hild, and G. Bakiri. 1995. A comparison of ID3 and backpropagation for English text-to-speech mapping. *Machine Learning*, 19(1):5–28.

E. Noreen. 1989. *Computer-intensive methods for testing hypotheses: an introduction*. John Wiley and sons.

A. Prince and P. Smolensky. 2004. *Optimality Theory: Constraint Interaction in Generative Grammar*. Blackwell Publishers.

V. Punyakanok and D. Roth. 2001. The use of classifiers in sequential inference. In *NIPS-13; The 2000 Conference on Advances in Neural Information Processing Systems*, pages 995–1001. The MIT Press.

T.J. Sejnowski and C.S. Rosenberg. 1987. Parallel networks that learn to pronounce english text. *Complex Systems*, 1:145–168.

N. Stroppa and F. Yvon. 2005. An analogical learner for morphological analysis. In *Proceedings of the 9th Conference on Computational Natural Language Learning*, pages 120–127. Association for Computational Linguistics.

K. Torkkola. 1993. An efficient way to learn English grapheme-to-phoneme rules automatically. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (*ICASSP*)*, volume 2, pages 199–202, Minneapolis.

A. Van den Bosch and W. Daelemans. 1993. Data-oriented methods for grapheme-to-phoneme conversion. In *Proceedings of the 6th Conference of the EACL*, pages 45–53.

A. Van den Bosch and W. Daelemans. 1999. Memory-based morphological analysis. In *Proceedings of the 37th Annual Meeting of the ACL*, pages 285–292, San Francisco, CA. Morgan Kaufmann.

A. Van den Bosch and W. Daelemans. 2005. Improving sequence segmentation learning by predicting trigrams. In I. Dagan and D. Gildea, editors, *Proceedings of the Ninth Conference on Computational Natural Language Learning*.

A. Weijters. 1991. A simple look-up procedure superior to NETtalk? In *Proceedings of the International Conference on Artificial Neural Networks* - ICANN-*91, Espoo, Finland*.

F. Yvon. 1996. *Prononcer par analogie: motivation, formalisation et évaluation*. Ph.D. thesis, Ecole Nationale Supérieure des Télécommunication, Paris.