Deliverable D4.2

Online learning for improving one, or more than one component

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 287678.

Participant no.	**Participant organisation name**	**Part. short name**	**Country**
1 (Coordinator)	University of Edinburgh	UEDIN	UK
2	Aalto University	AALTO	Finland
3	University of Helsinki	UH	Finland
4	Universidad Politécnica de Madrid	UPM	Spain
5	Technical University of Cluj-Napoca	UTCN	Romania

Project reference number	FP7-287678
Proposal acronym	SIMPLE4ALL
Status and Version	Complete, proofread, ready for delivery: version 1
Deliverable title	Online learning for improving one, or more than one component
Nature of the Deliverable	Report (R)
Dissemination Level	Public (PU)
This document is available from	`http://simple4all.org/publications/`
WP contributing to the deliverable	WP4
WP / Task responsible	WP4 / Task T4.2
Editor	Mikko Kurimo (AALTO)
Editor address	mikko.kurimo@aalto.fi
Author(s), in alphabetical order	Mircea Giurgiu, Dhananjaya N Gowda, Stig-Arne Grönroos, Reima Karhila, Mikko Kurimo, Juan Manuel Montero Martínez, Peter Smit, Adriana Stan, Oliver Watts
EC Project Officer	Leonhard Maqua

Abstract

This is the second deliverable of WP4 and it describes work done in M25 – M30 for the second task T4.2. The workplan was structured into topics where either one or more than one component of the TTS system are optimised by learning from user feedback. The main results for single component optimisation are interactive construction of letter-to-sound rules, polarity prediction, and semi-supervised morphological analysis. Work on topics where multiple components are optimised – a much harder task – based on user feedback is still in progress, but plans have become significantly more concrete and we expect to complete the planned experiments before the end of the project. These experiments include monitoring TTS quality while improving the system, analysing consistency among users in their feedback on the nature of a system's errors, and correction of non-native pronunciation using native spoken feedback.

Contents

1 Introduction

Progress in WP4 during the last six months has followed the lines agreed at in project-wide meetings held in Cluj in October 2013. Plans for the final six months of the project were made in project-wide meetings held in Madrid in April 2014.

In the course of the work it has become clear that ultimate goal of WP4 – the online learning of every component of a full system, based on end-user feedback – is very ambitious and probably too hard to achieve within one project. However, progress has been made in improving one or more components of a system based on user feedback and it must be emphasised that these approaches do of course also improve the end-to-end performance of a full TTS system.

As agreed earlier in the project, work has progressed on multiple fronts in parallel in order to investigate a wide variety of ideas about what kind of user feedback could be collected and learned from. This method provided successful results (see D4.1), so has been continued. During the final months of the project we will determine the feasibility and desirability of integrating some of the best learning-from-feedback methods into the final public release of our tools.

This is the second deliverable of this WP, and describes the work done for the second task of this WP. Much of the work reported here was already started during Year 2 with some preliminary results being reported in D4.1.

Although this deliverable describes the continuation of the same lines of research that were included in D4.1 as on-going and planned work, the structure of D4.2 is different to that of D4.1. The content has here been divided into optimisation of one component (Section 2) and more than one component (Section 3). The simple reason for this is to better match the original Description of Work. Results that have already been published or submitted for publication are described simply by appending the corresponding manuscript to this deliverable, as suggested by the reviewers at the Year 2 review. This deliverable contains, in addition to reports of completed or ongoing work, plans for experiments which will take place in the final 6 months of the project.

Because WP2 (development of the unsupervised TTS front-end) finished in month M24, some of the work that still needed to be done has continued within WP4. The work described here on morphological analysis and polarity prediction concerns improving performance by learning from user feedback, so fits naturally into WP4.

2 Optimisation of a single component

2.1 Interactive construction of letter-to-sound rules for Malay using active learning

The work on interactive construction of letter-to-sound rules for Malay mentioned in Deliverable 4.1 has been finalised and was submitted to Interspeech 2014. The submitted paper is appended to the current document (see Section 4).

2.2 Diarization

This topic was not continued further after deliverable D4.1 and there is nothing to report here. The performance of the diarization system is now considered good enough for our purposes.

2.3 Morfessor

Since the release of Morfessor 2.0 [1], a demo interface has been built and the toolkit has been demonstrated at the EACL conference [2]. The conference paper is included as an appendix to this deliverable.

Currently the focus of Morfessor work has shifted to finding suitable techniques for data selection. The first technique considered is the selection of words for which annotation should be requested. If this selection can be done in an unsupervised manner it will reduce the number of annotated words needed to produce a given improvement in the model.

Another technique being considered is that of finding unsupervised measures for identifying incorrect or foreign words in the dataset that is used to train Morfessor. When these words are identified, removal might improve the model after retraining, or the information can be used for other purposes, e.g. as information that propagates through the SIMPLE4ALL synthesis system.

If these techniques prove successful in informal evaluation, they will be evaluated more formally both against standard morphological error measures (e.g., segmentation boundary precision and recall) and by their performance in applications like the SIMPLE4ALL front-end.

2.3.1 Morfessor FlatCat

The work on Morfessor FlatCat was submitted to the Coling 2014 conference. The draft is attached as an appendix of the current document (Section 4).

2.4 Polarity Prediction

Experimental results of dimensional and categorical models on a sentence polarity prediction task using English corpora (SemEval, MovieReview, ISEAR) were previously presented in [3]. That work has been extended by the use of statistical measures (chi-square and relevance factor) to characterise words' semantic orientation and by evaluating the categorical polarity prediction model on two Spanish corpora: a corpus of *Avatar* movie reviews and one of Spanish tweets. Detailed results are presented in Deliverable D5.2 (Section 4).

However, the dimensional model described relies substantially on extensive hand-labelled resources – in particular, the English systems made use of a set of canonical values of valence, arousal, and dominance which were hand-coded for 14,000 English lemmas. Another line of work therefore aims to overcome the need for such extensive hand-labelling. We reduce the amount of supervision needed to create such polarity lexicons by using user feedback to incrementally adapt a polarity model. Initial small sets of seed words with positive and negative polarity are assumed. Unsupervised learning is used to suggest lists of similar words; users are asked to provide feedback regarding the rating of a found word as being positive or negative. In this way, a WordNet-style affective database for a new language can be generated in a lightly-supervised way. The candidate words presented to the user are selected by using a vector space model (VSM). The current focus is on finding the best way to present candidate words to the user so that a large number of relevant words are labeled.

To study the feasibility of the approach in different languages we have used text databases in English (Wall Street Journal news – 1.2 million tokens, and Wikipedia text – 250,000 tokens), Romanian (a collection of online news – 2.7 million tokens), and Spanish (*Avatar* movie reviews – 170,000 tokens). In a preprocessing stage, text is split into sentences, then numbers, dates and special characters are removed, and words either made up of fewer than 3 characters or found in a user-provided stopword list are discarded. Then, vector space models of word types in the corpus are built. We have generated various representations by modifying the number of words with which co-occurrence is counted (between 50 to 300, in steps of 50).

Some examples of discovered candidate words for several polar seed words are presented below. Words in bold typeface are initial seed words; other seed words are chosen by the user from the candidate list in the previous iteration of learning:

Positive seeds	*Candidates generated*
great	incredible, wonderful, good, terrific, fantastic, loved, amazing, enjoyable, touching, superb
incredible	awesome, fantastic, loved, touching, excellent, good, delightful, watched, obviously, underrating
amazing	terrific, decent, exceptional, everything, brilliant, superb, gave, wonderful, good, full
enjoyable	shown, watched, loved, party, saw, found, splatter, date, obvious, worthy
Negative seeds	*Candidates generated*
madness	connection, complicated, tearful, violent, flood, thieves, flashy, actual, returning, counterparts
thieves	mute, fabulious, cancer, useless, unexplainable, therapy, selfcentered, ranger, proofs
problem	splatter, hates, cheek, informative, acceptance, balanced, loose, forgotten, widow
offend	react, choke, miserable, incited, disturbed, capitalism, angry, comedians, react

During this pilot experiment in user feedback we have noticed the following things:

- positive words are discovered in a larger numbers than negative ones (this could be because of the nature and content of the text corpus used)
- among the discovered word candidates for a negative seed there are many neutral words, or even positive words; user feedback will be particularly important in such cases;
- after several iterations asking for user feedback we have noticed that the system tends to select words which have been already ranked. To avoid such situations we will look for methods to identify new words which are in a different region of the vector space.

Ongoing work will focus on fine-tuning of the VSM, evaluating the possibility of using Morfessor for unsupervised word stemming, developing an interface for presenting the user with candidate words, and getting user feedback via an online interface.

2.5 Polarity and emotional speech

In order to incorporate polarity prediction into a text-to-speech system, our proposal is to transplant emotions into the neutral speech of the selected speaker in our application, and modulate the emotional strength of the synthesised sentences over the course of the text to be synthesised. However, the relationship between text polarity and speech emotion needs to be established and fine-tuned for every domain of application, and user-feedback is a flexible approach for this fine-tuning process.

The overall architecture of the polarity-based emotional TTS system is a cascade of four modules:

- a polarity predictor,
- a domain-dependent polarity smoothing filter,
- a domain-dependent polarity-to-emotion mapper,
- an emotional TTS synthesiser with controllable emotional strength.

The filter aims to smooth the variations in polarity from one sentence to another, taking into account the overall polarity of the text or paragraph, and not only the predicted polarity of each sentence. The implemented approach can be formulated as follows: given a sequence of sentences $S = \{s_1...s_n...s_N\}$, the corresponding sequence of polarities $P = \{p_1...p_n...p_N\}$ (Figure 2.5a, left) and the average polarity $\bar{P}$, the inter-sentence polarity difference $p_n - p_{n-1}$ is smoothed by means of a non-linear smoothing function F. Currently, F is a generalised parametric logistic function:

$$F(p_n - p_{n-1}) = \frac{1}{(1 + e^{B(|p_n - p_{n-1}| - M)})^{1/v}} \tag{1}$$

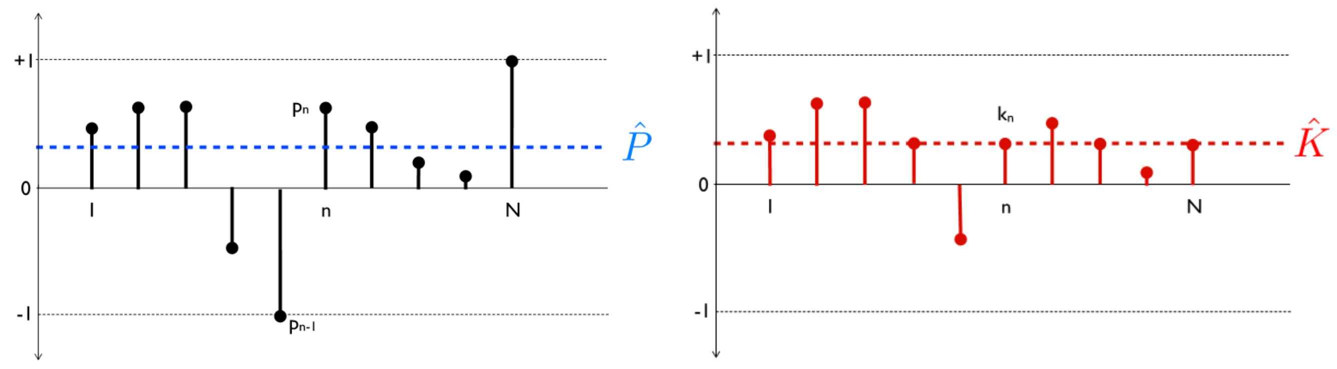


Figure 2.5a: Predicted polarity contour (left) and filtered polarity contour (right).

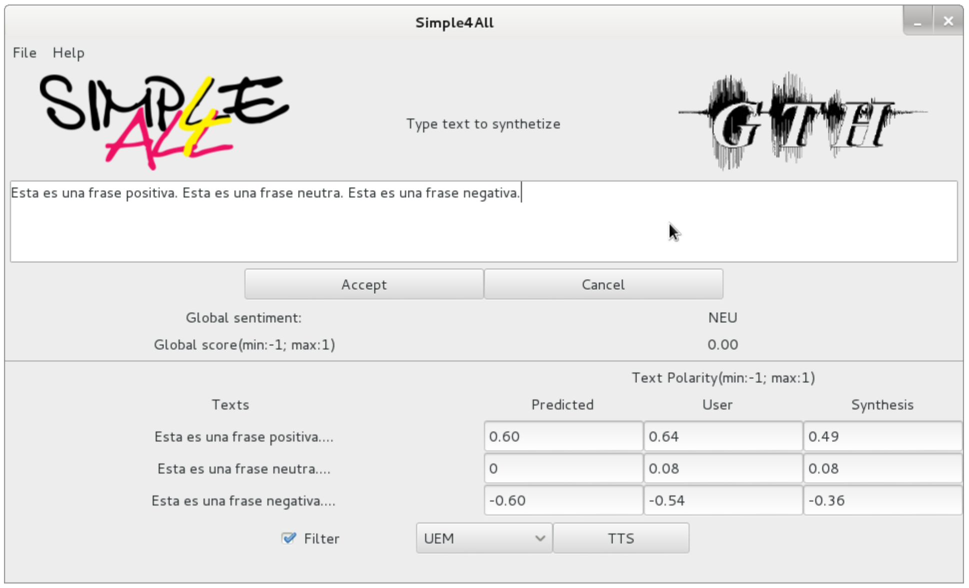
Finally, the filtered polarity contour (Figure 2.5a, right) is obtained by applying the following formula to the predicted values:

$$k_n = \bar{P} + (p_n - \bar{P})F(p_n - p_{n-1})) \tag{2}$$

Although the parameters of the smoothing function could be automatically estimated from polarity values corrected with user feedback gathered via our graphical user interface (input boxes in the User column in Figure 2.5b), this can also be done using an automatically estimated post-processing regression.

The mapper is another domain-dependent key component of the architecture. The mapper is modelled as a linear regression function which converts the filtered polarity values into emotional strength values appropriate to the sentences and the application domain. This regression function can be automatically estimated from values found using user feedback obtained via the GUI (from the input boxes in the synthesis column).

Regarding the emotional synthesiser, the GUI allows a target speaker (such as UEM on the bottom of Figure 2.5b) to be selected for the synthesis of the sentence sequence. Once the GUI has been fully implemented, the evaluation will be carried out in the remaining time of the project.



Esta es una frase positiva. Esta es una frase neutra. Esta es una frase negativa.

Global sentiment: NEU

Global score(min:-1; max:1) 0.00

Text Polarity(min:-1; max:1)

Texts	Predicted	User	Synthesis
Esta es una frase positiva....	0.60	0.64	0.49
Esta es una frase neutra....	0	0.08	0.08
Esta es una frase negativa....	-0.60	-0.54	-0.36

UEM

Figure 2.5b: Graphical user interface for emotional TTS.

3 Optimisation of more than one component

3.1 An objective measure of TTS quality trained on user feedback

A problem when developing and tuning TTS systems is that no well-established method of automatically rating the quality of TTS output exists; this is in contrast to, for example, ASR where word error rate is a well-established measure of system performance. Automatically computed measures such as mel cepstral distortion exist, and are sometimes used compare the quality of systems. However, such measures often do not correlate well with scores from perceptual evaluations of systems using human listeners. Automatic measures are therefore treated with caution by most TTS researchers. However, evaluations with human subjects are expensive and time-consuming to organise, and so cannot be used for intensive system tuning.

One obvious line of research would therefore to obtain a new automated measure of TTS output quality which correlates better with listener judgements than conventional objective measures. This measure would be computed by a system which is trained on listener scores such as those from the Blizzard Challenge, where TTS output from many systems is available together with natural speech and listener responses. The module would predict listener scores for TTS paired with reference natural speech for novel utterances.

We plan to work on this in the remainder of SIMPLE4ALL and some limitations of conventional quality measures which our system will try to overcome include:

1. Framewise nature of conventional objective measures – more global patterns, important for perception of speech naturalness, are ignored (e.g. F_0 contour over the course of a syllable)

2. Optionality: there may be multiple acceptable ways of realising a section of speech, weakening conventional objective measures. Training a predictor of listeners responses with speech from many TTS systems with many alternative realisations might be a way to overcome this issue.

3. Local vs. global degradation: general degradation of spectrum can have overall effect on perceived quality, and this type of degradation is captured well by conventional measures which average over all frames of an utterance. But these general degradations exist (in TTS) alongside local problems (sudden wrong F_0 excursion on a certain word, wrong inserted pause) which can outweigh the overall good quality of an utterance. The system needs to be able to handle and integrate information about these multiple types of errors.

4. Features used are not inherently geared to predicting subjects responses.

Rather than incorporating a large number of hand-engineered features into a predictive cascade, the module will be implemented as a neural network with learned features at every level. The network will be trained on time-aligned and concatenated TTS and reference speech to directly predict the listeners response on a per utterance basis.

Convolutional neural network methods might be able to deal with problem 3 – the use of multiple pooling layers using different subsampling methods could capture the difference between global low-level degradation (mean of a set of units is propagated) and local problematic hotspots (maximum of a set of units propagated). The fact that all layers of the network are trained to predict perceived quality overcomes problems 2 and 4: it will inherently learn features relevant to subjects responses, and should also learn something about optionality.

The ultimate goal of this work would be to integrate the predictor of perceived quality into TTS system training, although this may not be possible within SIMPLE4ALL . The predictor will provide a perceptually relevant error signal so that systems can be trained in an end-to-end manner. This work will be carried out by SIMPLE4ALL researchers at UEDIN, most probably in the form of a Masters dissertation project.

3.2 Speech-based feedback

The most intuitive way to correct perceived mistakes in spoken utterances would be to speak out a corrected version of the incorrect part of the utterance; the system would then use this speech feedback to learn what went wrong in

	Good	Will do	Mis-pronunc.	Bad rhythm	Bad audio	Incompr. segments	Audio length
1. Ok, quality is good	-	0.64	-0.19	-0.34	-0.37	-0.42	0.12
2. Ok, it's not great but it will do	0.64	-	-0.24	-0.26	-0.42	-0.57	0.06
3. Not ok: Mispronunciation of word(s)	-0.19	-0.24	-	-0.01	-0.22	0.09	-0.06
4. Not ok: Bad rhythm or intonation	-0.34	-0.26	-0.01	-	-0.26	-0.26	0.01
5. Not ok: Bad audio quality (artefacts etc)	-0.37	-0.42	-0.22	-0.26	-	0.01	0.03
6. Not ok: Incomprehensible segments	-0.42	-0.57	0.09	-0.26	0.01	-	-0.13

Table 3.2a: Correlations between different error types in the evaluated utterances. The rightmost column shows the correlation between utterance length and the different error types.

the utterance and attempt to correct it. To do this in a supervised manner might be possible, with a system developer doing the analysis and revising rule sets used in the lexical processing and model selection, so that the error will not be produced any more in the same context. In the field of unsupervised or semi-supervised speech synthesis this is a very challenging approach, as an automated analysis of the correction and ways to incorporate the corrections in the inner workings of the systems are required.

As described in D4.1, our investigations are still limited to updating the acoustic model. Aside from the technical details of how the collected correction data should be used to influence a system's future behaviour, we also need to investigate how the effectiveness of user-given feedback should be evaluated. Questions about quality tend to be highly personal, and so a statistical approach is required to find out 1) what kind of utterances need to be corrected and 2) if the methods for improving the voice work. The first question was investigated in a listening test; work is in progress to answer the second question.

3.2.1 Experiment on listener agreement on the nature of errors

By enabling a group of users to provide feedback to a system, we can gather a large quantity of feedback, but there is a potential problem that the quality of the feedback varies, just as the user base does. As a preliminary investigation into this 'quantity over quality'-approach to feedback, an experiment was conducted to estimate the consistency of listener opinions.

A low-quality voice was prepared, based on a UK English speech database. Of the 9500 available spoken sentences, only 300 sentences were used for training to ensure that the synthesised sentences would be of sufficiently low quality. STRAIGHT-based acoustic features and conventional linguistic features obtained from the Festival front-end were used to build the voice. The synthesised sentences have wide range of quality issues from mispronunciations to completely incomprehensible audio segments. 200 sentences were synthesised for evaluation. The text prompts used for synthesis correspond to a small part of the 9200 sentences remaining of the training data.

A listening test was run at UEDIN in which 34 listeners participated, and where each sentence was evaluated 9–19 times (17 times on average). Each listener heard a set of 100 sentences and assigned each of them into the most suitable category from the options shown in Table 3.2a. When there were multiple issues with a single utterance, listeners were instructed to select the most critical issue.

Figure 3.2a shows the distribution of classifications for each synthetic sentence, and Table 3.2a shows the correlations between different classifications. As was expected due to the small amount of data used in training the voice, the sentences that have been rated as good or acceptable are in a minority.

The next observation is that listeners have widely differing opinions about which aspect of the synthetic speech most needs to be improved. Table 3.2a also lists the correlation between utterance length and the different error types and shows that the length of the utterance does not play a critical role in the quality evaluation. Listener-specific analysis is in progress and by taking into account individual listener preferences, it might be possible to pinpoint more accurately the most important issues and with that information, to refine the training (and retraining)

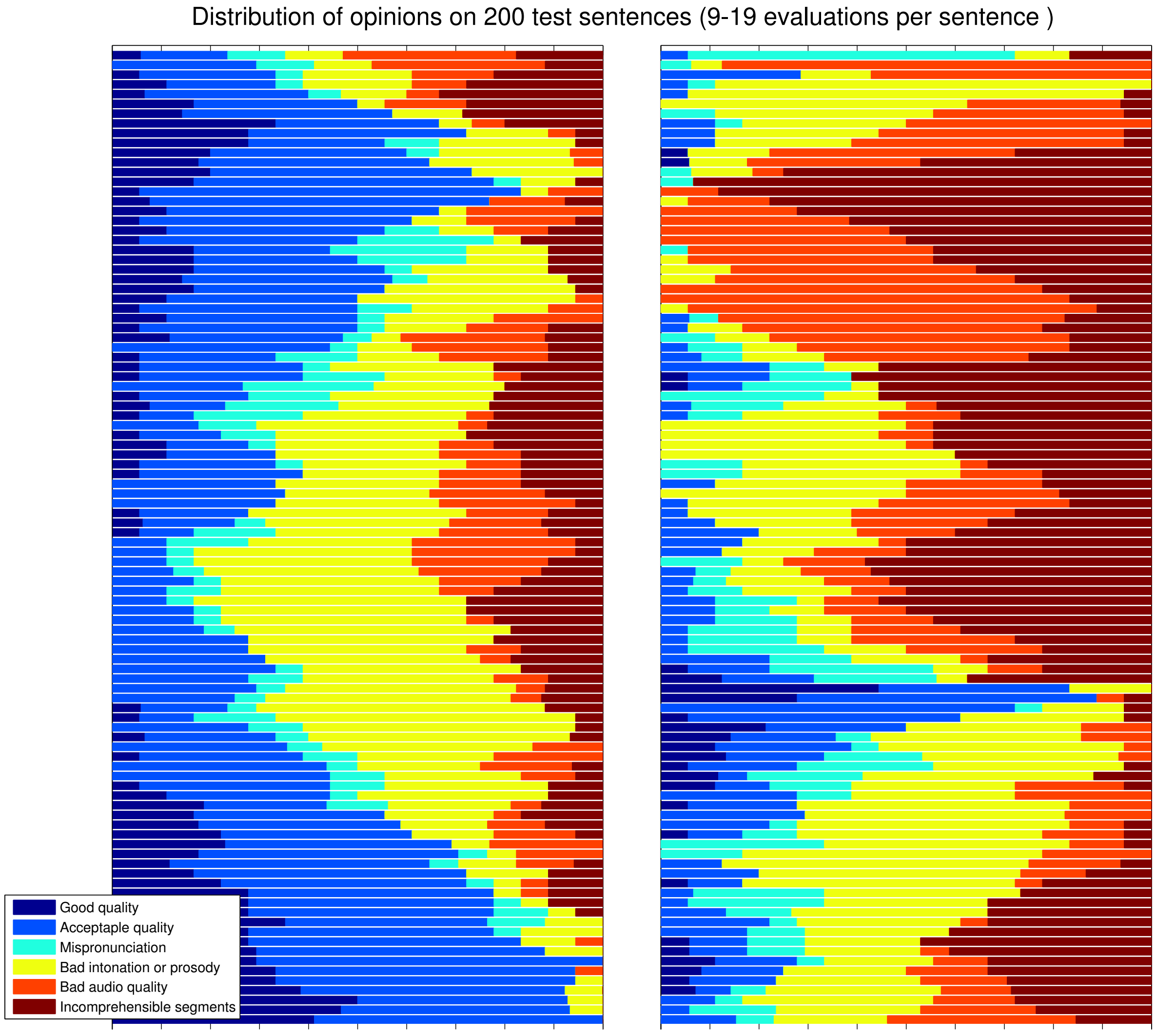


Figure 3.2a: Listening experiment results. Each row represents one of the rated utterances. Colours represent the classifications by listeners. The order of the utterances in the two columns is based on visual impression.

strategies to best improve the synthetic voice.

After the results have been analysed, the next phase of this work will investigate how the information on listener observations can best be used to improve the synthetic voice. There are very large amounts of training data available for this particular speaker, which we can use to simulate a situation in which the original speaker is asked to read out corrections. We will try to improve the voice as much as possible with as little additional data as possible, investigating the various possibilities outlined in Section 4.1.3 of D4.1.

3.2.2 Correction of non-native pronunciation using native spoken feedback

A straightforward way of improving speaking ability in a non-native language is to learn the pronunciation of words or short utterances by listening to examples from native speakers. The objective of this experiment is to study the feasibility of incorporating this style of learning into the framework of statistical speech synthesis.

Experiment 1:

Identifying and evaluating the significance of different streams in transplanting the pronunciation of one speaker onto another without affecting the speaker's identity. The prosody, as controlled by the duration of phones as well as the gain and F0 contours, is an important aspect of pronunciation. Transplanting these prosodic parameters may be sufficient if the accent of the non-native speaker quite close to that of the native speaker. In the case where the accent of the non-native speaker is very different, it may be necessary to modify the vocal tract parameters as well. Two speaker-dependent voices will be built using reasonable amount of data (say 800 utterances), one for a native English speaker and another for a non-native English speaker. The effect of substituting suitably normalized and/or transformed streams, one or several at a time, from the native speaker and the non-native speaker will be studied in terms of change in pronunciation and retention of speaker identity.

Experiment 2:

The idea is to use existing morphing techniques to morph the spoken user feedback from the native speaker to that of the non-native speaker. There are two possible scenarios. Case 1: User provides only examples of the words whose pronunciation need to be changed. Case 2: User provides a few neutral sentences along with the words that need to be corrected. The additional sentences in Case 2 will be used to learn a better mapping between the native and non-native speakers. The morphed utterances and/or the spoken examples of words from the native speaker will be used to either retrain the models or update only the models under consideration.

4 Conclusion

The work is progressing according to the plans specified in Deliverable D4.1. A number of the research topics belonging to WP4 have led to manuscripts that have been published or submitted for publication. These include the lightly-supervised learning and user feedback experiments to improve Malay TTS and the new algorithm for semi-supervised learning of morphology. Both of these topics belong to the category of optimisation of a single component relevant to TTS.

New experiments have been planned and performed to monitor TTS quality while learning from user feedback, to analyse listener agreement on the nature of TTS errors, and correction of non-native pronunciation using native spoken feedback. Both of these topics belong to the category of optimisation of multiple TTS components. The new experiments are designed to provide results that can reported in the next deliverable D4.3.

References

[1] Sami Virpioja, Peter Smit, Stig-Arne Grönroos, and Mikko Kurimo. Morfessor 2.0: Python implementation and extensions for Morfessor Baseline. Report 25/2013 in Aalto University publication series SCIENCE + TECHNOLOGY, Department of Signal Processing and Acoustics, Aalto University, Helsinki, Finland, 2013.

[2] Peter Smit, Sami Virpioja, Stig-Arne Grönroos, and Mikko Kurimo. Morfessor 2.0: Toolkit for statistical morphological segmentation. In *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 21–24, Gothenburg, Sweden, April 2014. Association for Computational Linguistics.

[3] I. Muresan, A. Stan, M. Giurgiu, and R. Potolea. Evaluation of sentiment polarity prediction using a dimensional and a categorical approach. In *Proc of the 7th Int Conf on Speech Technology and Human-Computer Dialogue, SpeD2013, ISBN: 978-1-4799-1063-2*, pages 23–28, 2013.

Appendix: Published Demo Paper (EACL 2014)

Morfessor 2.0: Toolkit for statistical morphological segmentation

Peter Smit[1]
peter.smit@aalto.fi

Sami Virpioja[2]
sami.virpioja@aalto.fi

Stig-Arne Grönroos[1]
stig-arne.gronroos@aalto.fi

Mikko Kurimo[1]
mikko.kurimo@aalto.fi

[1]Department of Signal Processing and Acoustics, Aalto University
[2]Department of Information and Computer Science, Aalto University

Abstract

Morfessor is a family of probabilistic machine learning methods for finding the morphological segmentation from raw text data. Recent developments include the development of semi-supervised methods for utilizing annotated data. Morfessor 2.0 is a rewrite of the original, widely-used Morfessor 1.0 software, with well documented command-line tools and library interface. It includes new features such as semi-supervised learning, online training, and integrated evaluation code.

1 Introduction

In the morphological segmentation task, the goal is to segment words into morphemes, the smallest meaning-carrying units. Morfessor is a family of methods for unsupervised morphological segmentation. The first version of Morfessor, called Morfessor Baseline, was developed by Creutz and Lagus (2002) its software implementation, Morfessor 1.0, released by Creutz and Lagus (2005b). A number of Morfessor variants have been developed later, including Morfessor Categories-MAP (Creutz and Lagus, 2005a) and Allomorfessor (Virpioja et al., 2010). Even though these algorithms improve Morfessor Baseline in some areas, the Baseline version has stayed popular as a generally applicable morphological analyzer (Spiegler et al., 2008; Monson et al., 2010).

Over the past years, Morfessor has been used for a wide range of languages and applications. The applications include large vocabulary continuous speech recognition (e.g. Hirsimäki et al., 2006), machine translation (e.g. Virpioja et al., 2007), and speech retrieval (e.g. Arisoy et al., 2009). Morfessor is well-suited for languages with concatenative morphology, and the tested languages include Finnish and Estonian (Hirsimäki et al., 2009), German (El-Desoky Mousa et al., 2010), and Turkish (Arisoy et al., 2009).

Morfessor 2.0 is a new implementation of the Morfessor Baseline algorithm.[1] It has been written in a modular manner and released as an open source project with a permissive license to encourage extensions. This paper includes a summary of the Morfessor 2.0 software and a description of the demonstrations that will be held. An extensive description of the features in Morfessor 2.0, including experiments, is available in the report by Virpioja et al. (2013).

2 Morfessor model and algorithms

Models of the Morfessor family are generative probabilistic models that predict compounds and their analyses (segmentations) given the model parameters. We provide a brief overview of the methodology; Virpioja et al. (2013) should be referred to for the complete formulas and description of the model and its training algorithms.

Unlike older Morfessor implementations, Morfessor 2.0 is agnostic in regard to the actual data being segmented. In addition to morphological segmentation, it can handle, for example, sentence chunking. To reflect this we use the following generic terms: The smallest unit that can be split will be an *atom* (letter). A *compound* (word) is a sequence of atoms. A *construction* (morph) is a sequence of atoms contained inside a compound.

2.1 Model and cost function

The cost function of Morfessor Baseline is derived using maximum a posteriori estimation. That is, the goal is to find the most likely parameters $\boldsymbol{\theta}$

[1]Morfessor 2.0 can be downloaded from the Morpho project website (`http://www.cis.hut.fi/projects/morpho/`) or GitHub repository (`https://github.com/aalto-speech/morfessor`).

given the observed training data $\boldsymbol{D}_W$:

$$\boldsymbol{\theta}_{\text{MAP}} = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta}) p(\boldsymbol{D}_W \mid \boldsymbol{\theta}) \qquad (1)$$

Thus we are maximizing the product of the model prior $p(\boldsymbol{\theta})$ and the data likelihood $p(\boldsymbol{D}_W \mid \boldsymbol{\theta})$. As usual, the cost function to minimize is set as the minus logarithm of the product:

$$L(\boldsymbol{\theta}, \boldsymbol{D}_W) = -\log p(\boldsymbol{\theta}) - \log p(\boldsymbol{D}_W \mid \boldsymbol{\theta}). \quad (2)$$

During training, the data likelihood is calculated using a hidden variable that contains the current chosen analyses. Secondly, it is assumed that the constructions in a compound occur independently. This simplifies the data likelihood to the product of all construction probabilities in the chosen analyses. Unlike previous versions, Morfessor 2.0 includes also the probabilities of the compound boundaries in the data likelihood.

For prior probability, Morfessor Baseline defines a distribution over the lexicon of the model. The prior assigns higher probability to lexicons that store fewer and shorter constructions. The lexicon prior consists of to parts, a product over the *form* probabilities and a product over the *usage* probabilities. The former includes the probability of a sequence of atoms and the latter the maximum likelihood estimates of the constructions. In contrast to Morfessor 1.0, Morfessor 2.0 currently supports only an implicit exponential length prior for the constructions.

2.2 Training and decoding algorithms

A Morfessor model can be trained in multiple ways. The standard batch training uses a local search utilizing recursive splitting. The model is initialized with the compounds and the full model cost is calculated. The data structures are designed in such way that the cost is efficient compute during the training.

In one epoch of the algorithm, all compounds in the training data are processed. For each compound, all possible two-part segmentations are tested. If one of the segmentations yields the lowest cost, it is selected and the segmentation is tried recursively on the resulting segments. In each step of the algorithm, the cost can only decrease or stay the same, thus guaranteeing convergence. The algorithm is stopped when the cost decreases less than a configurable threshold value in one epoch.

An extension of the Viterbi algorithm is used for decoding, that is, finding the optimal segmentations for new compound forms without changing the model parameters.

3 New features in Morfessor 2.0

3.1 Semi-supervised extensions

One important feature that has been implemented in Morfessor 2.0 are the semi-supervised extensions as introduced by Kohonen et al. (2010)

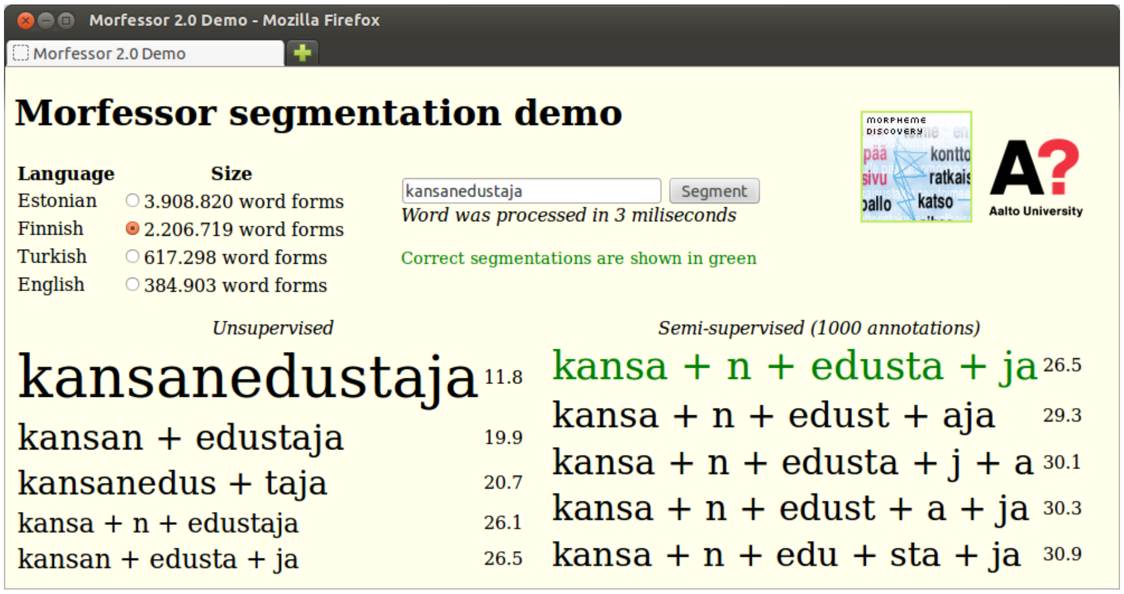
Morfessor Baseline tends to undersegment when the model is trained for morphological segmentation using a large corpus (Creutz and Lagus, 2005b). Oversegmentation or undersegmentation of the method are easy to control heuristically by including a weight parameter α for the likelihood in the cost function. A low α increases the priors influence, favoring small construction lexicons, while a high value increases the data likelihood influence, favoring longer constructions.

In semi-supervised Morfessor, the likelihood of an annotated data set is added to the cost function. As the amount of annotated data is typically much lower than the amount of unannotated data, its effect on the cost function may be very small compared to the likelihood of the unannotated data. To control the effect of the annotations, a separate weight parameter β can be included for the annotated data likelihood.

If separate development data set is available for automatic evaluation of the model, the likelihoods weights can be optimized to give the best output. This can be done by brute force using a grid search. However, Morfessor 2.0 implementation includes a simple heuristic for automatically tuning the value of α during the training, trying to balance precision and recall. A simple heuristic, which gives an equivalent contribution to the annotated data, is used for β.

3.2 On-line training

In addition to the batch training mode, Morfessor 2.0 supports on-line training mode, in which unannotated text is processed one compound at a time. This makes it simple to, for example, adapt pre-trained models for new type of data. As frequent compounds are encountered many times in running text, Morfessor 2.0 includes an option for randomly skipping compounds and constructions that have been recently analyzed. The random



Morfessor segmentation demo

Language	Size
Estonian	3.908.820 word forms
Finnish	[✓ 2.206.719 word forms]
Turkish	617.298 word forms
English	384.903 word forms

kansanedustaja [Segment]

Word was processed in 3 miliseconds

Correct segmentations are shown in green

Unsupervised		*Semi-supervised (1000 annotations)*	
kansanedustaja	11.8	kansa + n + edusta + ja	26.5
kansan + edustaja	19.9	kansa + n + edust + aja	29.3
kansanedus + taja	20.7	kansa + n + edusta + j + a	30.1
kansa + n + edustaja	26.1	kansa + n + edust + a + ja	30.3
kansan + edusta + ja	26.5	kansa + n + edu + sta + ja	30.9

Figure 1: Screenshot from the Morfessor 2.0 demo.

skips can also be used to speed up the batch training.

3.3 Integrated evaluation code

One common method for evaluating the performance of a Morfessor model is to compare it against a gold standard segmentation using segmentation boundary precision and recall. To make the evaluation easy, the necessary tools for calculating the BPR metric by (Virpioja et al., 2011) are included in Morfessor 2.0. For significance testing when comparing multiple models, we have included the Wilcoxon signed-rank test. Both the evaluation code and statistical testing code are accessible from both the command line and the library interface.

3.4 N-best segmentation

In order to generate multiple segmentations for a single compound, Morfessor 2.0 includes a n-best Viterbi algorithm. It allows extraction of all possible segmentations for a compound and the probabilities of the segmentations.

4 Demonstration

In the demonstration session, multiple features and usages of Morfessor will be shown.

4.1 Web-based demonstration

A live demonstration will be given of segmenting text with Morfessor 2.0 for different language and training data options. In a web interface, the user can choose a language, select the size of the training corpus and other options. After that a word can be given which will be segmented using n-best Viterbi, showing the 5 best results.

A list of planned languages can be found in Table 1. A screen shot of the demo interface is shown in Figure 1.

Languages	# Words	# Word forms
English	62M	384.903
Estonian	212M	3.908.820
Finnish	36M	2.206.719
German	46M	1.266.159
Swedish	1M	92237
Turkish	12M	617.298

Table 1: List of available languages for Morfessor 2.0 demonstration.

4.2 Command line interface

The new command line interface will be demonstrated to train and evaluate Morfessor models from texts in different languages. A diagram of the tools is shown in Figure 2

4.3 Library interface

Interfacing with the Morfessor 2.0 Python library will be demonstrated for building own scientific experiments, as well as integrating Morfessor in

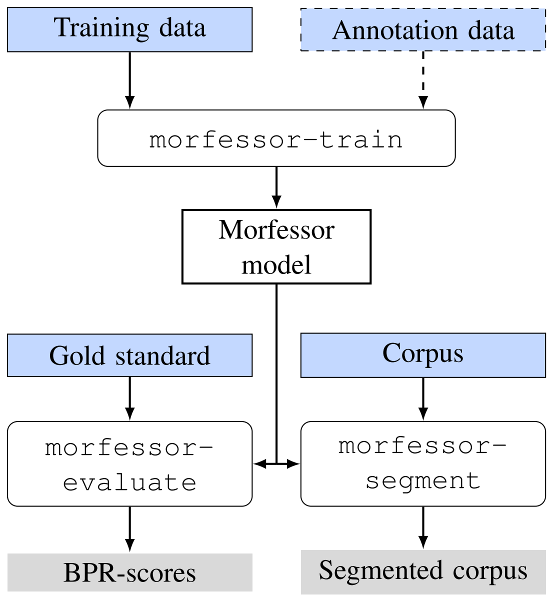


Figure 2: The standard workflow for Morfessor command line tools

bigger project. Also the code of the Web based demonstration will be shown as an example.

Acknowledgements

The authors have received funding from the EC's 7th Framework Programme (FP7/2007–2013) under grant agreement n°287678 and the Academy of Finland under the Finnish Centre of Excellence Program 2012–2017 (grant n°251170) and the LASTU Programme (grants n°256887 and 259934). The experiments were performed using computer resources within the Aalto University School of Science "Science-IT" project.

References

E. Arisoy, D. Can, S. Parlak, H. Sak, and M. Saraclar. 2009. Turkish broadcast news transcription and retrieval. *Audio, Speech, and Language Processing, IEEE Transactions on*, 17(5):874–883.

M. Creutz and K. Lagus. 2002. Unsupervised discovery of morphemes. In Mike Maxwell, editor, *Proceedings of the ACL-02 Workshop on Morphological and Phonological Learning*, pages 21–30. Association for Computational Linguistics, July.

M. Creutz and K. Lagus. 2005a. Inducing the morphological lexicon of a natural language from unannotated text. In *Proceedings of AKRR'05, International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning*, pages 106–113, Espoo, Finland, June. Helsinki University of Technology.

M. Creutz and K. Lagus. 2005b. Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0. Technical Report A81, Publications in Computer and Information Science, Helsinki University of Technology.

A. El-Desoky Mousa, M. Ali Basha Shaik, R. Schluter, and H. Ney. 2010. Sub-lexical language models for German LVCSR. In *Spoken Language Technology Workshop (SLT), 2010 IEEE*, pages 171–176. IEEE.

T. Hirsimäki, M. Creutz, V. Siivola, M. Kurimo, S. Virpioja, and J. Pylkkönen. 2006. Unlimited vocabulary speech recognition with morph language models applied to Finnish. *Computer Speech & Language*, 20(4):515–541.

T. Hirsimäki, J. Pylkkönen, and M. Kurimo. 2009. Importance of high-order n-gram models in morph-based speech recognition. *Audio, Speech, and Language Processing, IEEE Transactions on*, 17(4):724–732.

O. Kohonen, S. Virpioja, and K. Lagus. 2010. Semi-supervised learning of concatenative morphology. In *Proceedings of the 11th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology*, pages 78–86, Uppsala, Sweden, July. Association for Computational Linguistics.

C. Monson, K. Hollingshead, and B. Roark. 2010. Simulating morphological analyzers with stochastic taggers for confidence estimation. In *Multilingual Information Access Evaluation I. Text Retrieval Experiments*, pages 649–657. Springer.

S. Spiegler, B. Golénia, K. Shalonova, P. Flach, and R. Tucker. 2008. Learning the morphology of zulu with different degrees of supervision. In *Spoken Language Technology Workshop, 2008. SLT 2008. IEEE*, pages 9–12. IEEE.

S. Virpioja, J. Väyrynen, M. Creutz, and M. Sadeniemi. 2007. Morphology-aware statistical machine translation based on morphs induced in an unsupervised manner. In *Proceedings of the Machine Translation Summit XI*, pages 491–498, Copenhagen, Denmark, September.

S. Virpioja, O. Kohonen, and K. Lagus. 2010. Unsupervised morpheme analysis with Allomorfessor. In *Multilingual Information Access Evaluation I. Text Retrieval Experiments*, volume 6241 of *LNCS*, pages 609–616. Springer Berlin / Heidelberg.

S. Virpioja, V. Turunen, S. Spiegler, O. Kohonen, and M. Kurimo. 2011. Empirical comparison of evaluation methods for unsupervised learning of morphology. *TAL*, 52(2):45–90.

S. Virpioja, P. Smit, S. Grönroos, and M. Kurimo. 2013. Morfessor 2.0: Python implementation and extensions for Morfessor Baseline. Report 25/2013 in Aalto University publication series SCIENCE + TECHNOLOGY, Aalto University, Finland.

Appendix: Submitted Conference Paper (Interspeech 2014)

Combining lightly-supervised learning and user feedback to construct and improve a statistical parametric speech synthesiser for Malay

Lau Chee Yong,[1] *Oliver Watts*[2], *Simon King*[2]

[1]Faculty of Biosciences and Medical Engineering, Universiti Teknologi Malaysia, Malaysia
[2] Centre for Speech Technology Research, University of Edinburgh, UK
cylau2@live.utm.my, owatts@staffmail.ed.ac.uk, Simon.King@ed.ac.uk

Abstract

In spite of the learning-from-data used to train the statistical models, the construction of a statistical parametric speech synthesiser involves substantial human effort, especially when using imperfect data or working on a new language. Here, we use lightly-supervised methods for preparing the data and constructing the text-processing front end. This initial system is then iteratively improved using active learning in which feedback from users is used to disambiguate the pronunciation system in our chosen language, Malay. The data are prepared using speaker diarisation and lightly-supervised text-speech alignment. In the front end, grapheme-based units are used. The active learning used small amounts of feedback from a listener to train a classifier. We report evaluations of two systems built from high-quality studio data and lower-quality 'found' data respectively, and show that the intelligibility of each can be improved using active learning.

Index Terms: statistical parametric speech synthesis; active learning; lightly-supervised methods

1. Introduction

Conventional speech synthesis depends heavily on the availability of data: both the transcribed speech data required for waveform generation – whether that is through concatenation or statistical parametric models – as well as additional expert-annotated speech and text data. These additional data are required to train numerous predictors of linguistic features, prosodic structures, and so on (e.g. a pronunciation lexicon and a letter-to-sound module to predict the sequence of phonemes in an utterance). Our previous work has focused on finding ways to speed up the development of systems in new languages by developing techniques for transcribing 'found' data with only light supervision, and by using unsupervised learning to reduce reliance on expert annotated data.

In this paper, we describe the application of these techniques to the training of text-to-speech (TTS) systems for Malay. The existing techniques we have developed in previous work [1] and which are described in Section 2 are used to build a baseline voice. We then go on to describe and evaluate two new innovations to our existing toolset. All systems are based on the statistical parametric ("HMM-based") technique in which the speech corpus is used to estimate acoustic models of context-dependent sub-word units, which are then used to generate novel speech utterances. The contributions of this paper are in the preparation of the data and the construction of the text-processing front end. The estimation of the HMMs from the speech corpus is done in a standard way as described in [2], for example.

First of all, in Section 3 we describe the use of speaker diarisation techniques for extracting a homogenous subset of speech data from a public source of data featuring considerable variation. This technique supplements our existing methods for the lightly-supervised harvesting of data for acoustic model training [3] and is designed to further ease such data collection. The extension of our methods was found to be necessary because in the source of speech data used for the experimental systems of the present work, there is a much looser relation between speech and text than in the audiobooks we have used in previous work.

Then, in Section 4 we describe a method for improving the letter-based synthesis used in the baseline system by exploiting some limited interaction with a native speaker to disambiguate instances of an ambiguous grapheme, but without incurring the cost of compiling a full lexicon. This is an attractive approach for languages with generally regular letter-to-sound correspondences but with a handful of ambiguous letters: Malay is one example of such a language, but we expect the technique would be more widely applicable.

A baseline system built using our existing tools was evaluated for intelligibility and naturalness against systems built with 3 permutations of the innovative techniques. The experiment and results are presented in Section 5.

2. Baseline voice construction

A baseline voice was built from purpose-collected studio data using the procedures described in [1]. The data and techniques used to build this voice will be briefly outlined in this section.

2.1. Speech data collection

The studio data was obtained as described in [4]. The recording script was handcrafted: starting from a word list (from a pre-existing lexicon), text covering this vocabulary was gathered from various sources such as online news in Malay and a Malay language textbook. A recording script consisting of 900 sentences was composed using this text as a starting point. The sentences were not obtained by directly taking sentences from the text; instead, some existing sentences were edited to cover more words in the lexicon, and 250 of the sentences in the script were composed from scratch to further improve coverage of words in the lexicon. A Malay native male speaker was hired to record the resulting script, resulting in around 1 hour of speech material.

2.2. Front End Processing

We adopted the approach to constructing a front end processor described in [5]. The advantage of this method is that it requires no expert knowledge about the target language, such as a pronunciation dictionary, letter to sound rules or a part of speech tagger. Instead, an automatic distributional analysis, involving the construction of Vector Space Models (VSMs) is used to infer linguistic representations from a text corpus. For example, a VSM of letter distributions (i.e., simple counts in various contexts) provides a representation which can take the place of phonetic categories such as vowels, consonants, nasals, etc.

This is a reasonable choice because Malay has a highly regular alphabetic orthography, transparent affixation, simple syllable structure and a straightforward letter-to-sound relationship (with a small number of ambiguities, to which we offer a solution that does not rely on expert knowledge) [6], all of which make it eminently well matched to this approach to front end text processing. For both studio data and found data, the text corpus was manually transcribed and normalised. After normalisation, our text data consisted only of words, whitespace and punctuation symbols, and no abbreviations, numerals and other symbols remain.

2.3. Acoustic model training

Acoustic feature extraction and model training was done in the same way as described in e.g. [2]. Briefly, frames of 60 bark cepstral coefficients extracted from smoothed STRAIGHT [7] spectra were used, together with mel-warped F_0 and STRAIGHT aperiodicity measures in 25 frequency bands with non-linearly varying bandwidths. Dynamic (*delta* and *delta-delta*) coefficients were appended to the static parameters in the normal way.

The model was initialised by training context-independent grapheme models. These monographeme models were then converted into full-context models and retrained. The names of these full-context models encode the textual features described in Section 2.2. To handle problems of data sparsity and to be able to handle unseen contexts at synthesis time, decision trees were used to cluster acoustically similar states which are then tied, after which the tied models are retrained. Two iterations of clustering and retraining were used.

2.4. Synthesis of speech

At run-time, sentences to be synthesised are processed by the front end, resulting in a sequence of context-dependent grapheme labels for each utterance. According to this sequence, the decision trees built during model training are descended and the corresponding acoustic states are concatenated into an HMM of the utterance. A speech parameter generation algorithm [8] is then used to generate spectral and excitation parameters. The STRAIGHT vocoder is then used to generate a speech waveform from these generated parameters.

The two new techniques proposed for incorporation into the basic recipe outlined above are now described.

3. Innovation 1: Speaker diarisation to extract speech from found data

3.1. 'Found' data

The found data for this study is from the website `http://free-islamic-lectures.com` which is a free resource of Islamic teaching including audio recordings. It offers a free download of an audio recording of Al-quran read in intoned Arabic interspersed with a sentence-by-sentence translation into Malay. A total of 60 hours of audio data chunked into 114 files can be found on that website. Excluding the Arabic part leaves approximately 30 hours of Malay speech data. The Malay speech was recorded with an adult male voice, consistent speaking tone and using standard Malay speaking accent suitable for training data. However, the given text does not correspond exactly to the speech with a word accuracy of only approximately 30%. A subset of the text was therefore manually corrected: this is an expensive process and we were only able to obtain 3 hours of Malay speech with corrected transcription.

3.2. Diarisation

To extract Malay speech from the found data, we applied a speaker diarisation technique [9] as it is able to identify speaker homogenous regions throughout the speech data. First, feature extraction is performed on the data using HTK with standard ASR features. A GMM-HMM framework was applied whereby 16 clusters are initialized by dividing the speech frames into 32 uniform parts and using 2 parts (from different points in the data) to initialize each of the 16 GMMs. With these models, segmentation of the entire speech is done using a Viterbi algorithm with a forced minimum constraint of 250 ms. The models are retrained after segmentation followed by a clustering step to merge similar clusters based on the Bayesian Information Criterion (BIC). A penalty factor parameter is not required as the merged models have a complexity equal to sum of the complexity of the models being merged. The iteration terminates at the stopping criterion which is when the BIC scores for all clusters are below 0. Four wav files consist of 7 hours of raw data in total were chunked using diarisation. After the stopping criterion was met, there are 5 to 10 clusters remain in each file. Only the first cluster is the Malay speech that we want. All the other clusters are the intoned Arabic part. As result, 3 hours of Malay speech data were extracted from 7 hours of raw data and were chunked into 564 pieces of smaller data.

3.3. VAD and alignment

To extract transcribed segments of speech suitable for speech synthesizer training, we applied lightly supervised GMM Voice Activity Detection (VAD) [10] and grapheme-based automatic alignment [11] to our speech and text data. The aim of using these techniques is to segment the audio data and confidently match a subset of the resulting chunks with the corresponding text transcription. Our techniques require no prior language-specific knowledge and can be done in a very lightly-supervised manner: the only user input required is to match an initial few utterances to their transcription, which can be done with only some knowledge of the script used. The technique confidently aligned approximately half of the data, resulting in a total of 90 minutes of speech data which were then used for training our TTS systems.

4. Innovation 2: Interactive construction of letter-to-sound rules using active learning

The approach described in [5] and outlined in Section 2 has been shown to be effective for a variety of languages [1]. Because context-dependent models of grapheme-based acoustic units are used, it is at least theoretically the case that any predictable ambiguities in the letter-to-sound mapping (such as single letters that can be pronounced as two or more sounds) can be learned from the examples in the training data. Whilst this is an apparently elegant solution that does not require expert intervention, its performance is limited in practice by two factors. First, the training data are from the speech corpus being used for acoustic model estimation, which limits the number of types and tokens seen. Second, the letter-to-sound model is an integral part of the decision trees used for acoustic model parameter clustering, and these trees may not be the most efficient classifiers for resolving pronunciation ambiguities. Therefore, it may be be necessary to identify and resolve ambiguities using additional measures. Here, we focus on the resolution of one example ambiguity, which we identified by expert listening. Ongoing work is investigating how non-expert listeners could also be used to identify such problems, as well as to resolve them. The method we employ exploits some limited interaction with a native listener, but without incurring the cost – or requiring the expertise – involved in compiling a full pronunciation dictionary.

The relation between graphemes and phonemes in Malay is generally straightforward, but there are a limited number of graphemes with ambiguous pronunciations. One example is the letter <e>, which can correspond either to the phoneme /e/ or to schwa.

The uncertainty sampling approach to active learning was used [12]: several hundred examples of /e/ in different words were collected, and as predictor features neighbouring letters in a 7-letter window were collected. 150 randomly picked examples were initially hand-labelled (using the arbitrary symbols <e> and <é> to disambiguate the pronunciation). A further 200 examples were selected (one at a time) by active learning and presented to the user for labelling. The classifier resulting from the final iteration of active learning was then used to predict the pronunciation of new examples, including in the transcript of speech data for acoustic model training.

The learned classifier achieved an accuracy of 89.83% on the held-out, semantically unpredictable sentences described in Section 5.2.

Note that active learning has been used previously for learning letter-to-sound rules [13]. The difference in the current work is that instead of being asked to supply phonemic transcriptions of lexical items, a possibly technically naive user is instead asked to make a binary choice for each example of a small subset of letter types.

5. Experiment

5.1. Systems built

Four TTS systems were built, covering all combinations of data type (purpose-recorded vs. web-harvested) and letter-to-sound rules type (graphemes with actively learned disambiguation vs. plain graphemes). The four systems are summarised in Table 1. Systems B and D were trained on the 882 utterances of studio data described in Section 2.1, and systems C and E on the 435 utterances of web-harvested data described in Section 3.1. The quantity of data for all systems is comparable (approximately 1 hour). All systems are letter-based: systems B and C are purely letter-based, and systems D and E use the actively learned classifier for predicting the correct pronunciation of <e> described in Section 4.

As a reference 'system', utterances of the natural speech drawn equally from the purpose-recorded and found data and held out of the training set were used. These samples constitute 'system' A.

Table 1: Description of systems built

System	Description
A	Natural Speech
B	Studio data without active learning
C	Found data without active learning
D	Studio data with active learning
E	Found data with active learning

5.2. Evaluation

The synthetic speech from all systems was evaluated in a perceptual test performed by listeners. Two aspects of the speech were evaluated in this study: naturalness and intelligibility. For naturalness, Mean Opinion Score (MOS) was used: listeners were asked to rate the synthetic speech using a 5-point scale (1 for 'completely unnatural' and 5 for 'completely natural'). Natural speech (system A) was included in the naturalness section of the test. To test the intelligibility of the synthetic voices, listeners were asked to transcribe the synthetic speech they heard. In this section, semantically unpredictable sentences (SUS) were generated based on the criteria described in [14]. As naturally spoken SUS were not available, system A was excluded from this part of the evaluation.

5.2.1. *Naturalness*

For the naturalness part of the evaluation 10 sentence-texts were used; these were provided by the held-out natural utterances of system A. These 10 texts were synthesised with each TTS system, resulting in 50 experimental stimuli.

5.2.2. *Intelligibility*

10 utterance texts for the intelligibility part of the evaluation were generated for each system from the templates shown in Table 2. These sentences were synthesised by each of systems B–D, resulting in a set of 40 experimental stimuli. The SUS were generated in such a way that they consist mostly of out-of-training-corpus words. Only a few function words appear in both training data and the SUS script (e.g. *dengan*, *dan*, *ini*, *itu*, *ke* and *yang*). SUS were balanced over listeners and systems using a Latin square design. This meant that each listener heard 10 sentences spoken by each system, and heard each sentence text only once.

In short, the complete listening test consisted of 50 sentences to evaluate naturalness and 40 to evaluate intelligibility. In the intelligibility test, listeners were not permitted to listen to a sentence twice; they were exposed to speech synthesized by each system.

28 listeners were hired to listen to the stimuli and rate or transcribe them. The test was conducted via the web: listeners were asked to use headphones to listen to the speech. A subset of 8 listeners were invited into a quiet room to listen to the synthetic speech.

5.3. Result

The result of the naturalness test is shown in Figure 1.

Listeners' transcriptions were aligned with the known reference text of the stimuli, and word error rates (WER) of their transcriptions were computed per system. Figure 2 shows the word error rates (WER) of the systems, illustrating that the active learning improves intelligibility in both the high- and low-quality data scenarios. We performed a Wilcoxon Signed-Rank

Table 2: SUS templates with examples

Sentence type	Template	Example
Intransitive	Noun Det. Verb (intr.) Preposition Adjective Relat. Pronoun Noun	Sistem ini bekerjasama akan penglihatan yang lucu.
Transitive	Noun Adjective Verb (trans) Noun Adjective	Bola sepak tinggi menjamin lumrah keji.
Imperative	Verb (trans.) Noun Conjunction Noun	Dirikan rumah dan rambutan.
Interrogative	Quest. Adv Noun Verb(trans.) Noun Relat. Pronoun Adjective	Bilakah pendingin hawa ambil generasi yang senyap.
Relative	Noun Det. Verb (trans.) Det Noun Relat. Pronoun Preposition Verb(intr.)	Gunung ini menyapu balai bomba yang sungguh pahit.

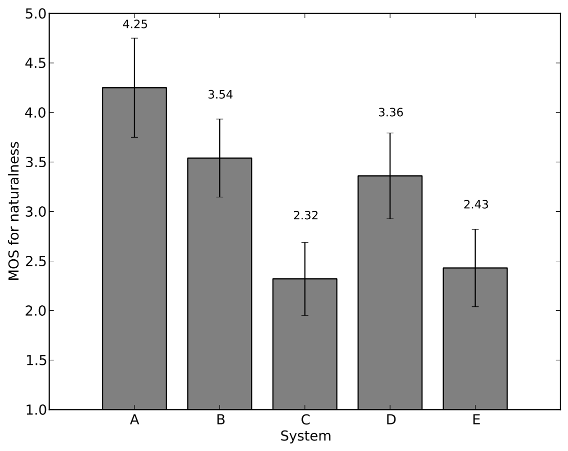


Figure 1: Naturalness of the systems. The use of found data leads to less natural voices, as expected. Active learning, which resolves a letter-to-sound ambiguity, has no effect on naturalness.

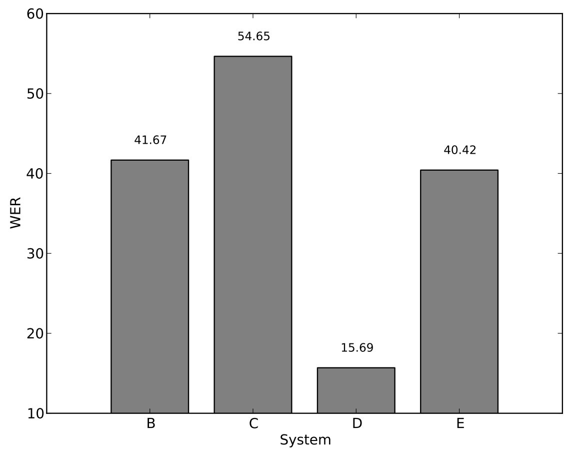


Figure 2: Intelligibility of the systems, expressed as Word Error Rate (lower is better). Active learning, which resolves a letter-to-sound ambiguity has a substantial effect, particularly on the system built from a better quality speech corpus (system D vs system B).

Test ($\alpha = 0.05$) to test the significance of the differences between all pairs of systems. All differences in WER were found to be significant except that between systems B and E, showing that active learning is highly effective in improving intelligibility.

6. Conclusion

We have presented four configurations of a Malay statistical parametric speech synthesizer in this study, which allowed us to evaluate the effect of replacing the standard high-quality studio data with lower-quality 'found' data, and to test a novel active learning technique for pronunciation disambiguation in both data conditions. The Malay language is straightforward in terms of orthography and language structure and in general it poses no major problems in building TTS systems. However, one major difficulty is predicting correct pronunciations for the ambiguous grapheme <e>. In this study, when classifiers trained using the active learning technique were used to predict the correct pronunciation of this letter in words encountered at run-time, we found significant improvements to intelligibility for systems trained on both types of data.

7. Acknowledgements

We thank Mark Sinclair for helping us apply his speaker diarisation tools to our data. The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement N^{o} 287678 (Simple4All).

8. References

[1] O. Watts, A. Stan, R. Clark, Y. Mamiya, M. Giurgiu, J. Yamagishi, and S. King, "Unsupervised and lightly-supervised learning for rapid construction of TTS systems in multiple languages from 'found' data: evaluation and analysis," in *8th ISCA Workshop on Speech Synthesis*, Barcelona, Spain, August 2013, pp. 121–126.

[2] J. Yamagishi and O. Watts, "The CSTR/EMIME HTS System for Blizzard Challenge," in *Proc. Blizzard Challenge 2010*, Sep. 2010.

[3] A. Stan, O. Watts, Y. Mamiya, M. Giurgiu, R. Clark, J. Yamagishi, and S. King, "TUNDRA: A Multilingual Corpus of Found Data for TTS Research Created with Light Supervision," in *Proc. Interspeech*, Lyon, France, August 2013.

[4] L. C. Yong and T. T. Swee, "Low footprint high intelligibility malay speech synthesizer based on statistical data," *Journal of Computer Science*, vol. 10, pp. 316–324, 2014.

[5] J. Lorenzo-Trueba, O. Watts, R. Barra-Chicote, J. Yamagishi, S. King, and J. M. Montero, "Simple4all proposals for the albayzin evaluations in speech synthesis," in *In Proc. Iberspeech*, 2012.

[6] M. J. Yap, S. J. R. Liow, S. B. Jalil, and S. S. B. Faizal, "The malay lexicon project: A database of lexical statistics for 9,592 words," *Behavior research methods*, vol. 42, no. 4, pp. 992–1003, 2010.

[7] H. Kawahara, I. Masuda-Katsuse, and A. Cheveigné, "Restructuring speech representations using a pitch-adaptive time-frequency smoothing and an instantaneous-frequency-based F0 extraction: possible role of a repetitive structure in sounds," *Speech Communication*, vol. 27, pp. 187–207, 1999.

[8] K. Tokuda, T. Yoshimura, T. Masuko, T. Kobayashi, and T. Kitamura, "Speech parameter generation algorithms for hmm-based speech synthesis," in *In: Proc. ICASSP*, 2000, pp. 1315–1318.

[9] M. Sinclair and S. King, "What are the challenges in speaker diarization?" in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013.

[10] Y. Mamiya, J. Yamagishi, O. Watts, R. A. Clark, S. King, and A. Stan, "Lightly supervised gmm vad to use audiobook for speech synthesiser," in *In Proc. ICASSP*, 2013.

[11] A. Stan, P. Bell, J. Yamagishi, and S. King, "Lightly Supervised Discriminative Training of Grapheme Models for Improved Sentence-level Alignment of Speech and Text Data," in *Proc. of Interspeech (accepted)*, 2013.

[12] D. D. Lewis and W. A. Gale, "A sequential algorithm for training text classifiers," in *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*.

[13] K. Dwyer and G. Kondrak, "Reducing the annotation effort for letter-to-phoneme conversion," in *ACL/IJCNLP*, 2009, pp. 127–135.

[14] C. Benoit, M. Grice, and V. Hazan, "The SUS test: A method for the assessment of text-to-speech synthesis intelligibility using Semantically Unpredictable Sentences," *Speech Communication*, vol. 18, no. 4, pp. 381 – 392, 1996.

Appendix: Submitted Conference Paper (Coling 2014)

Morfessor FlatCat: An HMM-Based Method for Unsupervised and Semi-Supervised Learning of Morphology

Stig-Arne Grönroos
Department of Signal Processing
and Acoustics, Aalto University
stig-arne.gronroos@aalto.fi

Sami Virpioja
Department of Information and
Computer Science, Aalto University
sami.virpioja@aalto.fi

Abstract

Morfessor is a family of methods for learning morphological segmentations of words based on unannotated data. We introduce a new variant of Morfessor, FlatCat, that applies a hidden Markov model structure. It builds on previous work on Morfessor, sharing model components with the popular Morfessor Baseline and Categories-MAP variants. Our experiments show that while unsupervised FlatCat does not reach the accuracy of Categories-MAP, with semi-supervised learning it provides state-of-the-art results in the Morpho Challenge 2010 tasks for English, Finnish, and Turkish.

1 Introduction

Morphological analysis is essential for automatic processing of compounding and highly-inflecting languages, for which the number of unique word forms may be very large. Apart from rule-based analyzers, the task has been approached by machine learning methodology. Especially unsupervised methods that require no linguistic resources have been studied widely (Hammarström and Borin, 2011). Typically these methods focus on morphological segmentation, i.e., finding *morphs*, the surface forms of the morphemes.

For language processing applications, unsupervised learning of morphology can provide decent-quality analyses without resources produced by human experts. However, while morphological analyzers and large annotated corpora may be expensive to obtain, a small amount of linguistic expertise is more easily available. A well-informed native speaker of a language can often identify the different prefixes, stems, and suffixes of words. Then the question is how many annotated words makes a difference. One answer was provided by Kohonen et al. (2010), who showed that already one hundred manually segmented words provide significant improvements to the quality of the output when comparing to a linguistic gold standard.

The semi-supervised approach by Kohonen et al. (2010) was based on Morfessor Baseline, the simplest of the Morfessor methods by Creutz and Lagus (2002; 2007). The statistical model of Morfessor Baseline is simply a categorical distribution of morphs—a unigram model in the terms of statistical language modeling. As the semi-supervised Morfessor Baseline outperformed all unsupervised and semi-supervised methods evaluated in the Morpho Challenge competitions (Kurimo et al., 2010a) so far, the next question is how the approach works for more complex models.

Another popular variant of Morfessor, Categories-MAP (CatMAP) (Creutz and Lagus, 2005), models word formation using a hidden Markov model (HMM). The context-sensitivity of the model improves the precision of the segmentation. For example, it can prevent splitting a single *s*, a common English suffix, from the beginning of a word. Moreover, it can disambiguate between identical morphs that are actually surface forms of different morphemes. Finally, separation of stems and affixes in the output makes it simple to use the method as a stemmer.

In contrast to Morfessor Baseline, the lexicon of CatMAP is *hierarchical*: a morph that is already in the lexicon may be used to encode the forms of other morphs. This has both advantages and drawbacks.

One downside is that it mixes the prior and likelihood components of the cost function, so that the semi-supervised approach presented by Kohonen et al. (2010) is not usable.

1.1 Hierarchical versus flat lexicons

From the viewpoint of data compression and following the two-part Minimum Description Length principle (Rissanen, 1978), Morfessor tries to minimize the number of bits needed to encode both the model parameters and the training data. Equivalently, the cost function L can be derived from the Maximum a Posteriori (MAP) estimate:

$$\hat{\theta} = \arg\max_{\theta} \mathrm{P}(\theta \,|\, \boldsymbol{D}) = \arg\min_{\theta} \big(-\log \mathrm{P}(\theta) - \log \mathrm{P}(\boldsymbol{D} \,|\, \theta) \big) = \arg\min_{\theta} L(\theta, \boldsymbol{D}), \tag{1}$$

where θ are the model parameters, $\boldsymbol{D}$ is the training corpus, $\mathrm{P}(\theta)$ is the prior of the parameters and $\mathrm{P}(\boldsymbol{D} \,|\, \theta)$ is the data likelihood.

In context-independent models such as Morfessor Baseline, the parameters include only the forms and probabilities of the morphs in the lexicon of the model. Morfessor Baseline and Categories-ML (CatML) (Creutz and Lagus, 2004) use a flat lexicon, in which the forms of the morphs are encoded directly as strings: each letter requires a certain number of bits to encode. Thus longer morphs are more expensive. Encoding a long morph is worthwhile only if the morph is referred to frequently enough from the words in the training data. If a certain string, let us say *segmentation*, is common enough in the training data, it is cost-effective to have it as a whole in the lexicon. Splitting it into two items, *segment* and *ation*, would double the number of pointers from the data, even if those morphs were already in the lexicon. The undersegmentation of frequent words becomes evident especially if the training data is a corpus instead of a list of unique word forms.

In contrast, Morfessor CatMAP applies a hierarchical lexicon, which makes use of the morphs that are already in the lexicon. Instead of encoding the form of *segmentation* by its 12 letters, we could just encode the form with two references to the forms of the morphs *segment* and *ation*. This may also cause errors, for example encoding *station* with *st* and *ation*.

The lexicon of Morfessor CatMAP allows but does not force hierarchical encoding for the forms: each morph has an extra parameter that indicates whether it has a hierarchical representation or not. The problem of oversegmentation, as in *st + ation*, is solved using the morph categories. The categories, which are states of the HMM, include stem, prefix, suffix, and a special non-morpheme category. The non-morpheme category is intended to catch segments that do not fit well into the three proper morph categories because they are fragments of a larger morph. In our example, the morph *st* cannot be a suffix as it starts the word, it is unlikely to be a prefix as it directly precedes a common suffix *ation*, and it is unlikely to be a stem as it is very short. Thus the algorithm is likely to use the non-morpheme state. The hierarchy is expanded only up to the level in which there are no non-morphemes, so the final analysis is still *station*. Without the hierarchy, the non-morphemes have to be removed heuristically, as in CatML (Creutz and Lagus, 2004).

A hierarchical lexicon presents some challenges to model training. For a standard unigram or HMM model, if you know the state and emission sequence of the training data, you can directly derive the maximum likelihood (ML) parameters of the model: a probability of a morph is proportional to the number of times it is referred to, conditional on the state in the HMM. But if the lexicon is partly hierarchical, also the references *within* the lexicon add to the reference counts, and there is no direct way to find the ML parameters even if the encoding of the training data is known. Similarly, semi-supervised learning cannot be accomplished simply by adding the counts from an annotated data set, as it is not clear when to use hierarchy instead of segmenting a word directly in the data.

Moreover, for a flat lexicon, the cost function divides into two parts that have opposing optima: the cost of the data (likelihood) is optimal when there is minimal splitting and the lexicon consists of the words in the training data, whereas the cost of the model (prior) is optimal when the lexicon is minimal and consists only of the letters. In consequence, the balance of precision and recall of the segmentation boundaries can be directly controlled by setting a weight for the data likelihood. Tuning this hyper-parameter is a very simple form of supervision, but it has drastic effects on the segmentation results

(Kohonen et al., 2010). A direct control of the balance may also be useful for some applications: Virpioja et al. (2011) found that the performance of the segmentation algorithms in machine translation correlates more with the precision than the recall. The weighting approach does not work for hierarchical lexicons, for which changing the weight does not directly affect the decision whether to encode the morph with hierarchy or not.

1.2 Morfessor FlatCat

In this paper, we introduce a new member to the Morfessor family, Morfessor FlatCat. As indicated by its name, FlatCat uses a flat lexicon. Our hypothesis is that enabling semi-supervised learning is effective in compensating for the undersegmentation caused by the lack of hierarchy. In particular, semi-supervised learning can improve modeling of suffixation. In the examined languages, suffixes tend to serve syntactic purposes, such as marking case, tense, person or number. Examples are the suffix *s* marking tense and person in *she writes* and number in *stations*. Thus the suffix class is closed and has only a small number of morphemes compared to the prefix and stem categories. As a consequence, a large coverage of suffixes can be achieved already with a relatively small annotated data set.

The basic model of morphotactics in FlatCat is the same as in the CatML and CatMAP variants: a hidden Markov model with states that correspond to a word boundary and four morph categories: stem, prefix, suffix, and non-morpheme. As in CatML, we apply heuristics for removal of non-morphemes from the final segmentation. However, due to the MAP estimation of the parameters, these heuristics are not necessary for controlling the model complexity, but merely a post-processing step to get useful categories.

Modeling of morphotactics improves the segmentation of compound words, by allowing the overall level of segmentation to be increased without increasing the number of correct morphs used in incorrect positions. As the benefits of semi-supervised learning and improved morphotactics are likely to complement each other, we can expect improved performance over the semi-supervised Morfessor Baseline method.

2 FlatCat model and algorithms

Morfessor FlatCat uses components from the older Morfessor variants. Instead of going through all the details, we refer to the previous work and highlight only the differences.

As a generative model, Morfessor FlatCat describes the joint distribution $\mathrm{P}(\boldsymbol{A}, \boldsymbol{W} \mid \theta)$ of words and their analyses. The words $\boldsymbol{W}$ are observed, but their analyses, $\boldsymbol{A}$, is a latent variable in the model. An analysis of a word contains its morphs and morph categories: prefix, stem, suffix, and non-morpheme. As marginalizing over all possible analyses is generally infeasible, point estimates are used during the training. The likelihood conditioned on the current analyses is

$$\mathrm{P}(\boldsymbol{D} \mid \boldsymbol{A}, \theta) = \prod_{j=1}^{|\boldsymbol{D}|} \mathrm{P}(\boldsymbol{A}_j \mid \theta). \tag{2}$$

If m_i are the morphs in $\boldsymbol{A}_j$, c_i are the hidden states of the HMM corresponding to the categories of the morphs, and $\#$ is the word boundary, $\mathrm{P}(\boldsymbol{A}_j \mid \theta)$ is

$$\mathrm{P}(c_1 \mid \#) \prod_{i=1}^{|\boldsymbol{A}_j|} \left[\mathrm{P}(m_i \mid c_i)\, \mathrm{P}(c_{i+1} \mid c_i) \right] \mathrm{P}(\# \mid c_{|\boldsymbol{A}_j|}). \tag{3}$$

Morfessor FlatCat applies an MDL-derived prior designed to control the number of non-zero parameters. The prior is otherwise the same as in Morfessor Baseline, but it includes the usage properties from Morfessor CatMAP: the length of the morph and its right and left perplexity. The perplexity measures describe the predictability of the context in which the morph occur. The emission probability of a morph conditioned on the morph category, $\mathrm{P}(m \mid c)$, is calculated from the properties of the morphs similarly as in CatMAP.

(a) English.

Method	α	β	Pre	Rec	F
U Baseline	1.0	–	.88	.59	.71
U CatMAP	–	–	.89	.51	.65
U FlatCat	1.0	–	.90	.57	.69
W Baseline	0.7	–	.83	.62	.71
W FlatCat	0.5	–	.84	.60	.70
SS Baseline	1.0	3000	.83	**.77**	.80
SS FlatCat	0.9	2000	.86	.76	.81
SS CRF+FlatCat	0.9	2000	.87	**.77**	**.82**
S CRF	–	–	**.92**	.73	.81

(b) Finnish.

Method	α	β	Pre	Rec	F
U Baseline	1.0	–	.84	.38	.53
U CatMAP	–	–	.76	.51	.61
U FlatCat	1.0	–	.84	.38	.52
W Baseline	.02	–	.62	.54	.58
W FlatCat	.015	–	.66	.52	.58
SS Baseline	.1	15000	.75	.72	.73
SS FlatCat	.2	1500	.79	.71	.75
SS CRF+FlatCat	.2	2500	.82	**.76**	.79
S CRF	–	–	**.88**	.74	**.80**

Table 1: Boundary Precision and Recall results in comparison to gold standard segmentation. Abbreviations have been used for Unsupervised (U), likelihood weighted (W), semi-supervised (SS) and fully supervised (S) methods. Best results for each measure have been hilighted using boldface.

2.1 Training algorithms

The parameters are optimized using a local search. Only a part of the parameters are optimized in each step: the parameters that are used in calculating the likelihood of a certain part, *unit*, of the corpus. Units vary in complexity, from all occurrences of a certain morph to the occurrences of a morph bigram whose context fits to certain criteria.

The algorithm tries to simultaneously find the optimal segmentation for the unit and the optimal parameters consistent with that segmentation:

$$(\boldsymbol{A}, \theta) = \underset{\textsc{Op}(\boldsymbol{A},\theta)}{\arg\min} \big\{ L(\theta, \boldsymbol{A}, \boldsymbol{D}) \big\}. \tag{4}$$

The training operators Op define the units changed by the local search and the alternative segmentations tried for each unit. There are three training operators: *split, join* and *resegment*, analogous to the similarly named stages in CatMAP.

The split operator is applied first. It targets all occurrences of a specific morph in the corpus simultaneously, attempting to split it into two parts. The whole corpus is processed by sorting the current morphs by length from shortest to longest.

The second operator attempts to join morph bigrams, grouped by the position of the bigram in the word. The position grouped bigram counts are sorted by frequency, from most to least common.

Finally, resegmenting uses the generalized Viterbi algorithm to find the currently optimal segmentation for one whole word at a time. This operator targets each word in the corpus in increasing order of frequency.

The heuristics used in FlatCat to remove non-morphemes from the final segmentation are the following: All consequent non-morphemes are joined together. If the resulting morph is longer than 4 characters, it is accepted as a stem. All non-morphemes preceded by a suffix and followed by only suffixes or other non-morphemes are recategorized as suffixes without joining with their neighbors. If any short non-morphemes remain, they are joined either to the preceding or following morphs (the latter only for those in the initial position).

2.2 Semi-supervised learning

Kohonen et al. (2010) found that semi-supervised learning of Morfessor models was not effective by only fixing the values of the analysis $\boldsymbol{A}$ for the annotated samples $\boldsymbol{D}_A$. Their solution was to introduce corpus likelihood weights α and β, one for the unannotated data set and one for the annotated data set.

Method	α	β	Pre	Rec	F
U Baseline	1.0	–	.85	.36	.51
U CatMAP	–	–	.83	.50	.62
U FlatCat	1.0	–	.87	.36	.51
W Baseline	0.1	–	.71	.41	.52
W FlatCat	0.3	–	.88	.38	.53
SS Baseline	0.4	2000	.86	.60	.71
SS FlatCat	0.8	2666	.87	.59	.70
SS CRF+FlatCat	1.0	3000	.87	**.61**	**.72**
S CRF	–	–	**.89**	.58	.70

Table 2: BPR results for Turkish

Thus, instead of optimizing the MAP estimate, Kohonen et al. (2010) minimize the cost

$$L(\theta, \boldsymbol{A}, \boldsymbol{D}, \boldsymbol{D}_A) = -\log \mathrm{P}(\theta) - \alpha \log \mathrm{P}(\boldsymbol{D} \mid \boldsymbol{A}, \theta) - \beta \log \mathrm{P}(\boldsymbol{D}_A \mid \boldsymbol{A}, \theta). \tag{5}$$

The weights can be tuned on a development set. We use the same scheme for FlatCat.

The likelihood of the annotated data is calculated using the same HMM that is used for the unannotated data. The morph properties are estimated only from the unannotated data. To ensure that the morphs required for the annotated data can be emitted, a copy of each word in the annotations is added to the unannotated data. This unannotated copy is loosely linked to the annotated word: operations that would result in the removal of a morph required for the annotations from the lexicon cannot be selected, as such an operation would have infinite cost.

3 Experiments

We compare Morfessor FlatCat to two previous Morfessor methods and a fully supervised discriminative segmentation method. The Morfessor methods used as references are the CatMAP and Baseline implementations by Creutz and Lagus (2005) and Virpioja et al. (2013), respectively. Also the FlatCat implementation is based on the latter.[1] For a supervised discriminative model, we use a character-level conditional random field (CRF) implementation by Ruokolainen et al. (2013).

We use the English, Finnish and Turkish data sets from Morpho Challenge 2010 (Kurimo et al., 2010b). They include large unannotated word lists, one thousand annotated words for training, 700–800 annotated words for parameter tuning, and 10×1000 annotated words for testing.

For evalution, we use the BPR score by Virpioja et al. (2011). The score calculates the precision (Pre), recall (Rec), and F_1-score (F) of the predicted morph boundaries compared to a linguistic gold standard. In the presence of alternative gold standard analyses, we weight each alternative equally. We also report the mean average precision from the English and Finnish information retrieval (IR) tasks of the Challenge.

Morfessor FlatCat is a pipeline method that refines an initial segmentation given as input. We try two different initializations for the semi-supervised setting: initializing with the segmentation produced by semi-supervised Morfessor Baseline, and initializing with the CRF segmentation. All unsupervised and likelihood-weighted results are initialized with the corresponding Baseline output.

All methods were trained using word types. The weight and perplexity threshold parameters were optimized separately for each method, using a grid search with the held-out data set.

3.1 Comparison to linguistic gold standards

The results of the BPR evaluations are shown in Tables 1 (English, Finnish) and 2 (Turkish). Semi-supervised FlatCat initialized using CRF achieves the highest F-score for both the English and Turkish

[1]A link to our implementation will be published in the camera-ready version.

(a) English.

	STM	STM + STM			PRE + STM			STM + SUF			STM + SUF + SUF		
Method	Pre	Pre	Rec	F	Pre	Rec	F	Pre	Rec	F	Pre	Rec	F
U CatMAP	**.90**	.94	.63	.75	**.91**	.64	.75	.87	.45	.59	.90	.51	.65
SS Baseline	.64	.93	.77	.84	.82	**.74**	**.77**	.83	.86	.84	.91	.79	.85
SS FlatCat	.68	.94	.65	.77	.78	.62	.69	.86	.88	.87	.94	.79	.86
SS CRF+FlatCat	.68	**.95**	**.78**	**.86**	.78	.66	.72	.87	.89	.88	.94	.80	.87
S CRF	.78	.94	.72	.81	.85	.59	.69	**.92**	**.91**	**.91**	**.95**	**.82**	**.88**

(b) Finnish.

	STM	STM + STM			PRE + STM			STM + SUF			STM + SUF + SUF		
Method	Pre	Pre	Rec	F	Pre	Rec	F	Pre	Rec	F	Pre	Rec	F
U CatMAP	**.77**	.90	**.97**	**.94**	.88	**.96**	**.92**	.67	.46	.54	.68	.38	.49
SS Baseline	.50	.82	.88	.85	.73	.83	.78	.64	.85	.73	.76	.78	.77
SS FlatCat	.49	**.91**	.95	.93	.80	.89	.85	.67	.84	.75	.77	.75	.76
SS CRF+FlatCat	.53	**.91**	.96	**.94**	.84	.94	.88	.71	.88	.79	.80	.79	.79
S CRF	.68	.88	.91	.89	**.90**	.91	.91	**.83**	**.91**	**.87**	**.91**	**.85**	**.88**

Table 3: Results of BPR experiments with different morph category patterns. Best results for each measure have been hilighted using boldface.

data sets. The difference between the highest and second-highest scoring methods is statistically significant for Finnish and Turkish, but not for English (Wilcoxon signed-rank test, $p < 0.01$).

Table 3 shows BPR for subsets of words consisting of different morph category patterns. Each subset consists of 500 words from the English or Finnish gold standard, with one of five selected morph patterns as the only valid analysis. The subsets consist of words with the following morph patterns: words that should not be segmented (STM), compound words consisting of exactly two stems (STM + STM), a prefix followed by a stem (PRE + STM), a stem followed by a single suffix (STM + SUF) and a stem and exactly two suffixes (STM + SUF + SUF). For the STM pattern only precision is reported, as recall is not defined for an empty set of true boundaries.

The fact that semi-supervised FlatCat compares well against CatMAP in recall, for all morph patterns and for the test set as a whole, indicates that supervision indeed is effective in compensating for the undersegmentation caused by the lack of hierarchy in the lexicon. The benefit of modeling morphotactics can be seen in improved precision for the STM + STM (for English and Finnish) and PRE + STM (for Finnish) patterns when comparing against semi-supervised Baseline. While not directly observable in Table 3, a large part of the improvement over semi-supervised Baseline is explained by that FlatCat does not use suffix-like morphs in incorrect positions.

Initializing the FlatCat model with CRF segmentation improves the F-scores in all subsets compared to the initialization with Morfessor Baseline. While FlatCat cannot keep the accuracy of the suffix boundaries at as high level as CRF, it clearly improves the stem splitting.

3.2 Information retrieval

Stemming has been shown to improve IR results (Kurimo et al., 2009), by removing inflection that is often not relevant to the query. The morph categories make it possible to simulate stemming by removing morphs categorized as prefixes or suffixes. As longer affixes are more likely to be meaningful, we limited the affix removal to morphs of at most 3 letters. For methods that use morph categories, we report two IR results: the first using all the data and the second with short affix removal (SAR) applied.

In the IR results, we include the topline methods from Morpho Challenge: Snowball Porter stemmer (Porter, 1980) for English and "TWOL first" for Finnish. The latter selects the lemma from the first

(a) English.

Rank		Method	SAR	MAP
1	–	Snowball Porter	–	0.4092
2	SS	Baseline	–	0.3855
3	**SS**	**FlatCat**	**No**	**0.3837**
4	**SS**	**FlatCat**	**Yes**	**0.3821**
5	**SS**	**CRF+FlatCat**	**Yes**	**0.3810**
6	**SS**	**CRF+FlatCat**	**No**	**0.3788**
7	S	CRF	–	0.3771
8	W	Baseline	–	0.3761
9	U	Baseline	–	0.3695
10	U	CatMAP	No	0.3682
11	U	CatMAP	Yes	0.3653
12	**W**	**FlatCat**	**No**	**0.3651**
13	–	(First 5)	–	0.3648
14	**W**	**FlatCat**	**Yes**	**0.3606**
15	**U**	**FlatCat**	**No**	**0.3486**
16	**U**	**FlatCat**	**Yes**	**0.3451**
17	–	(Words)	–	0.3303

(b) Finnish.

Rank		Method	SAR	MAP
1	**W**	**FlatCat**	**No**	**0.5057**
2	**W**	**FlatCat**	**Yes**	**0.5029**
3	**SS**	**FlatCat**	**Yes**	**0.4987**
4	–	TWOL first	–	0.4973
5	**SS**	**CRF+FlatCat**	**Yes**	**0.4912**
6	U	CatMAP	Yes	0.4884
7	U	CatMAP	No	0.4865
8	**SS**	**CRF+FlatCat**	**No**	**0.4826**
9	**SS**	**FlatCat**	**No**	**0.4821**
10	–	(First 5)	–	0.4757
11	SS	Baseline	–	0.4722
12	S	CRF	–	0.4660
13	W	Baseline	–	0.4582
14	U	Baseline	–	0.4378
15	**U**	**FlatCat**	**Yes**	**0.4349**
16	**U**	**FlatCat**	**No**	**0.4334**
17	–	(Words)	–	0.3483

Table 4: Information Retrieval results. Results of the method presented in this paper are hilighted using boldface. Mean Average Precision is abbreviated as MAP. Short affix removal is abbreviated as SAR.

of the possible analyses given by the morphological analyzer FINTWOL (Lingsoft, Inc.) based on the two-level model by Koskenniemi (1983). As baseline results we also include unsegmented word forms and truncating each word after the first five letters (First 5).

The results of the IR experiment are shown in Table 4. FlatCat provides the highest score for Finnish. The English scores are similar to those of the semi-supervised Baseline. FlatCat performs better than CRF for both languages. This is explained by the higher level of consistency in the segmentations produced by FlatCat, which makes the resulting morphs more useful as query terms. The number of words in the lexicons of FlatCat initialized using CRF are 108 391 (English), 46 123 (Finnish) and 74 193 (Turkish), which is much smaller than the respective lexicon sizes counted from the CRF segmentation: 339 682 (English), 396 869 (Finnish) and 182 356 (Turkish). This decrease in lexicon size indicates a more structured segmentation.

The IR performance of semi-supervised FlatCat benefits from the removal of short affixes for English when initialized by CRF, and Finnish for both initializations. It also improves the results of unsupervised FlatCat and CatMAP for Finnish, but lowers the precision for English. A possible explanation is that the unsupervised methods do not analyze the suffixes with a high enough accuracy.

4 Conclusions

We have introduced a new variant of the Morfessor method, Morfessor FlatCat. It predicts both morphs and their categories based on unannotated data, but also annotated training data can be provided. It was shown to outperform earlier Morfessor methods in the semi-supervised learning task for English, Finnish and Turkish.

The purely supervised CRF-based segmentation method proposed by Ruokolainen et al. (2013) outperforms FlatCat for Finnish and reaches the same level for English. However, we show that a discriminative model such as CRF gives inconsistent segmentations that do not work as well in a practical application: In English and Finnish information retrieval tasks, FlatCat clearly outperformed the CRF-based segmentation.

We see two major directions for future work. Currently Morfessor FlatCat, like most Morfessor methods, assumes that words in a sentence occur independently. Making use of the sentence context in which words occur would, however, allow making Part-Of-Speech -like distinctions. These distinctions could help disambiguate inflections of different lexemes that have the same surface form but should be analyzed differently (Can and Manandhar, 2013).

The second direction is removal of the assumption that a morphology consists only of concatenative processes. Introducing transformations to model allomorphy in a similar manner as Kohonen et al. (2009) would allow finding the shared abstract morphemes underlying different allomorphs. This could be especially beneficial in information retrieval and machine translation applications.

References

Burcu Can and Suresh Manandhar. 2013. Dirichlet processes for joint learning of morphology and pos tags. In *Proceedings of the International Joint Conference on Natural Language Processing*, pages 1087–1091, Nagoya, Japan, October.

Mathias Creutz and Krista Lagus. 2002. Unsupervised discovery of morphemes. In Mike Maxwell, editor, *Proceedings of the ACL-02 Workshop on Morphological and Phonological Learning*, pages 21–30, Philadelphia, PA, USA, July. Association for Computational Linguistics.

Mathias Creutz and Krista Lagus. 2004. Induction of a simple morphology for highly-inflecting languages. In *Proceedings of the Seventh Meeting of the ACL Special Interest Group in Computational Phonology*, pages 43–51, Barcelona, Spain, July. Association for Computational Linguistics.

Mathias Creutz and Krista Lagus. 2005. Inducing the morphological lexicon of a natural language from unannotated text. In Timo Honkela, Ville Könönen, Matti Pöllä, and Olli Simula, editors, *Proceedings of AKRR'05, International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning*, pages 106–113, Espoo, Finland, June. Helsinki University of Technology, Laboratory of Computer and Information Science.

Mathias Creutz and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing*, 4(1):3:1–3:34, January.

Harald Hammarström and Lars Borin. 2011. Unsupervised learning of morphology. *Computational Linguistics*, 37(2):309–350, June.

Oskar Kohonen, Sami Virpioja, and Mikaela Klami. 2009. Allomorfessor: Towards unsupervised morpheme analysis. In *Evaluating Systems for Multilingual and Multimodal Information Access: 9th Workshop of the Cross-Language Evaluation Forum, CLEF 2008, Aarhus, Denmark, September 17–19, 2008, Revised Selected Papers*, volume 5706 of *Lecture Notes in Computer Science*, pages 975–982. Springer Berlin / Heidelberg, September.

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

Kimmo Koskenniemi. 1983. *Two-level morphology: A general computational model for word-form recognition and production*. Ph.D. thesis, University of Helsinki.

Mikko Kurimo, Sami Virpioja, Ville T. Turunen, Graeme W. Blackwood, and William Byrne. 2009. Overview and results of Morpho Challenge 2009. In *Working Notes for the CLEF 2009 Workshop*, Corfu, Greece, September.

Mikko Kurimo, Sami Virpioja, Ville Turunen, and Krista Lagus. 2010a. Morpho challenge 2005-2010: Evaluations and results. In Jeffrey Heinz, Lynne Cahill, and Richard Wicentowski, editors, *Proceedings of the 11th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology*, pages 87–95, Uppsala, Sweden, July. Association for Computational Linguistics.

Mikko Kurimo, Sami Virpioja, and Ville T. Turunen. 2010b. Overview and results of Morpho Challenge 2010. In *Proceedings of the Morpho Challenge 2010 Workshop*, pages 7–24, Espoo, Finland, September. Aalto University School of Science and Technology, Department of Information and Computer Science. Technical Report TKK-ICS-R37.

Martin F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

Jorma Rissanen. 1978. Modeling by shortest data description. *Automatica*, 14:465–471.

Teemu Ruokolainen, Oskar Kohonen, Sami Virpioja, and Mikko Kurimo. 2013. Supervised morphological segmentation in a low-resource learning setting using conditional random fields. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 29–37, Sofia, Bulgaria, August. Association for Computational Linguistics.

Sami Virpioja, Ville T Turunen, Sebastian Spiegler, Oskar Kohonen, and Mikko Kurimo. 2011. Empirical comparison of evaluation methods for unsupervised learning of morphology. *Traitement Automatique des Langues*, 52(2):45–90.

Sami Virpioja, Peter Smit, Stig-Arne Grönroos, and Mikko Kurimo. 2013. Morfessor 2.0: Python implementation and extensions for Morfessor Baseline. Report 25/2013 in Aalto University publication series SCIENCE + TECHNOLOGY, Department of Signal Processing and Acoustics, Aalto University.