User Manual for Normalisation of Noisy Input Data using HistNorm

Eva Pettersson
Department of Linguistics and Philology
Uppsala University
eva.pettersson@lingfil.uu.se

October 9, 2015

1 Introduction

This user manual describes how to use the HistNorm package for normalisation of noisy input data. The original usage of the script was for normalising historical spelling to a modern spelling, to enable the application of taggers and parsers trained on modern language to the historical input text. The program may however also be used for normalisation of other types of non-standard or noisy input text, such as SMS and Twitter text. If you publish work based on this program, kindly refer to any of the following papers:

- Eva Pettersson, Beáta Megyesi and Joakim Nivre (2013)
  Normalisation of Historical Text Using Context-Sensitive Weighted Levenshtein Distance and Compound Splitting.
  In: Proceedings of the 19th Nordic Conference of Computational Linguistics (NODALIDA 2013); Linköping Electronic Conference Proceedings 85. Oslo, Norway, 2013.
- Eva Pettersson, Beáta Megyesi and Jörg Tiedemann (2013)
  An SMT Approach to Automatic Annotation of Historical Text.
  In: Proceedings of the Workshop on Computational Historical Linguistics at NoDaLiDa 2013. NEALT Proceedings Series 18; Linköping Electronic Conference Proceedings 87:54-69. Oslo, Norway, 2013.
- Eva Pettersson, Beáta Megyesi and Joakim Nivre (2014)
  A Multilingual Evaluation of Three Spelling Normalisation Methods for Historical Text.
  In: Proceedings of the 8th Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities (LaTeCH) @ EACL 2014, pages 32—41. Gothenburg, Sweden, 2014.

There are currently two normalisation methods to choose from in the script: *Levenshtein-based* or *SMT-based* normalisation. For more information on the performance of these two methods for different languages, see further Pettersson et al (2014) [3].

2 Installation

The normalisation scripts are stored in the tgz-file `HistNorm.tgz`. The command `tar xvzf HistNorm.gz` will generate the following:

- `usersguide-histnorm.pdf`, in other words this user manual :-)
- The directory `resources`, containing language resources and pre-trained models for Levenshtein-based and SMT-based normalisation of specific languages.
- The directory `scripts`, containing scripts needed for Levenshtein-based normalisation. (SMT-based normalisation is performed using the Moses system, downloadable from `http://www.statmt.org/moses/`)

3 Levenshtein-Based Normalisation

In the Levenshtein-based approach, the input strings are normalised by calculating the Levenshtein edit distance between the original word form and any tokens present in a dictionary of modern standard language. The dictionary entry/entries with the smallest edit distance as compared to the original word form are stored as possible normalisation candidates, given that the edit distance is below a preset threshold. See further Pettersson et al (2013) [2] for a more detailed description of the Levenshtein-based normalisation approach.

3.1 Running the Levenshtein Normalisation Script

To normalise a text using the Levenshtein method, run the command `normalise_levenshtein.perl`, located in the `scripts` directory. The normalisation script takes six arguments:

1. The input file to be normalised. The format should be one token per line. An example file for German is given in: `resources/german/levenshtein/german.de-hs.test.hs`.
2. A valid path to a modern language dictionary to be used for Levenshtein comparisons. If no ordinary dictionary is available, a modern language corpus of reasonable size may be used instead of a dictionary. The format should be one token per line. An example file for German is given in: `resources/german/levenshtein/parole.mono.de.uniq`.
3. A valid path to a corpus of original word forms mapped to manually normalised spellings. In the normalisation process, whenever a token is encountered that also occurs in this corpus, the manually modernised word form will be chosen from this corpus. Only tokens not occurring in the corpus will be normalised by means of Levenshtein calculations. Each line of the file should contain one original word form mapped to its corresponding modern spelling by a separating tab. An example file for German is given in: `resources/german/levenshtein/german.de-hs.train.hsde`. If no such file is available, instead enter the string `nostop` as the third argument.

4. A valid path to a corpus of modern language, used for choosing the most frequent normalisation candidate in cases where several candidates are equally close in spelling to the source word. The format should be one token per line. An example file for German is given in:
   `resources/german/levenshtein/parole.mono.de`,
   and an example file for Swedish is given in:
   `resources/swedish/levenshtein/saldo-total_wordforms.txt`.

5. A valid path to a file containing weights lower than 1 for commonly occurring edits observed in training data. An example file for German is given in: `resources/german/levenshtein/training-weights.german.txt`. (See further Section 3.2 for information on how to generate weights based on your own training data.) If no such file is available, instead enter the string `noweights` as the fifth argument. All edits will then be given a weight of 1.

6. A valid path to a file containing the edit distance threshold value, for a dictionary entry to be regarded as a normalisation candidate. An example file for German is given in: `resources/german/levenshtein/threshold.german.txt`. (See further Section 3.2 for information on how to generate a threshold based on your own training data.)

If you want to run Levenshtein-based normalisation with the pre-trained German models that come with the installation of `HistNorm`, the full command would be:

```
perl -CSAD scripts/normalise_levenshtein.perl
[inputfile]
resources/german/levenshtein/parole.mono.de.uniq
resources/german/levenshtein/german.de-hs.train.hsde
resources/german/levenshtein/parole.mono.de
resources/german/levenshtein/weights.german.txt
resources/german/levenshtein/threshold.german.txt
> [outputfile]
```

3.2 Set Your Own Weights and Edit Distance Threshold

Provided that you have access to a corpus of original word forms mapped to manually normalised spellings, you may set your own weights and edit distance threshold based on this corpus. To do so, start by splitting your corpus into approximately 90% training and 10% tuning. Each line of your training and tuning corpora should contain one original word form mapped to its corresponding normalised spelling by a separating tab. Example files for German are given in `resources/german/levenshtein/german.de-hs.train.hsde` (training) and `resources/german/levenshtein/german.de-hs.dev.hsde` (tuning). The script `setWeightsAndThreshold.sh`, located in the `scripts` directory, may then be run with the following five arguments:

1. A valid path to the working directory, where the scripts that come with the installation are located.

2. Your input training corpus. An example file for German is given in: `resources/german/levenshtein/german.de-hs.train.hsde`.
3. Your input tuning corpus. An example file for German is given in: `resources/german/levenshtein/german.de-hs.dev.hsde`.
4. A valid path to the output file where the resulting weights should be stored.
5. A valid path to the output file where the resulting edit distance threshold should be stored.

If you want to run the script with the German example files that come with the installation of `HistNorm`, the full command would be:

```
sh scripts/setWeightsAndThreshold.sh
[workingdir]
resources/german/levenshtein/german.de-hs.train.hsde
resources/german/levenshtein/german.de-hs.dev.hsde
weights.german.txt
threshold.german.txt
```

4 SMT-based Normalisation

In the SMT-based approach, spelling normalisation is treated as a translation task, where the original spelling is translated to a normalised spelling using the Moses statistical machine translation system, with the GIZA++ toolkit for character alignment [1]. To address changes in spelling rather than full translation of words and phrases, character-based machine translation is performed, as opposed to the traditional word-based and phrase-based models. In character-level SMT, phrases are modeled as character sequences instead of word sequences, and translation models are trained on character-aligned parallel corpora, whereas language models are trained on character N-grams.

4.1 Running SMT-based Normalisation

To perform SMT-based normalisation as proposed in this guide, start by installing the Moses statistical machine translation system, downloadable from the following website: `http://www.statmt.org/moses/`. Normalisation is then run in the same way as you would normally run any Moses translation task, the only difference being the format of the input files. Training and tuning corpora (i.e. tokens in their historical and modernised spelling), as well as the modern language corpus used as a language model, have to be in the format of one token per line, with blank lines separating sentences, and with space separating the characters within each token. This will make the SMT system regard each character as a word, the full token as a sentence and the entire sentence as a paragraph. Accordingly, the file to be normalised also has to comply with this format. Example files for German are given in the directory `scripts/german/smt/`, see further Section 5.1.2.

In my normalisation experiments, I have run the Moses toolkit in its standard settings with Giza++ unigram models. The maximum size of a phrase (sequence of characters) was set to 10, and language models (10-gram models) were estimated using SRILM (interpolated and smoothed). Reordering was switched off during decoding (and tuning), as monotonic alignments are assumed. See further Pettersson et al (2013) [4] for detailed information on the settings used in the SMT-based normalisation approach.

5 Pre-Trained Models

The installation of HistNorm currently comes with pre-trained models for normalising Early New High German into Modern German. Pre-trained models for more languages are planned to be included continuously.

5.1 German

The pre-trained German normalisation models are based on two sources:

- Token pairs of historical word forms mapped to modern spellings are compiled from a manually normalised subset of the *GerManC* corpus of German texts from the period 1650–1800 [5]. This subset contains 22 texts from the period 1659–1780, within the genres of drama, newspaper text, letters, sermons, narrative prose, humanities, science och legal documents.
- The German *Parole* corpus is used as the single modern language resource [6].

Both these corpora are distributed by the University of Oxford under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License: (http://creativecommons.org/licenses/by-nc-sa/3.0/).

5.1.1 Levenshtein-based Normalisation

Files needed for Levenshtein-based normalisation using the pre-trained German model are stored in the `resources/german/levenshtein` directory:

- Training corpus, historical spelling mapped to modern spelling: `german.de-hs.train.hsde`
- Tuning corpus, historical spelling mapped to modern spelling: `german.de-hs.dev.hsde`
- Modern language dictionary for Levenshtein comparisons: `parole.mono.de.uniq`
- Modern language corpus for frequency-based choice of a final normalisation candidate: `parole.mono.de`
- Weights lower than 1 for frequently occurring edits: `weights.german.txt`
- Edit distance threshold: `threshold.german.txt`

5.1.2 SMT-based Normalisation

Files needed for SMT-based normalisation using the pre-trained German model are stored in the `resources/german/smt` directory:

- Training corpus, historical spelling: `german.de-hs.train.hs`
- Training corpus, modern spelling: `german.de-hs.train.de`
- Tuning corpus, historical spelling: `german.de-hs.dev.hs`
- Tuning corpus, modern spelling: `german.de-hs.dev.de`
- Language model, modern spelling: `parole.mono.de`
- Configuration file: `de-hs.giza.1-gram.ini`. If you use this file, be sure to change the search paths to the phrase-table and the language model, to comply with your directories.
- Phrase table: `phrase-table.gz`
- Compiled language model: `parole.mono.1-gram.de.10.kenlm`

6 License

The HistNorm script is released under a Creative Commons Attribution-ShareAlike 3.0 Unported license (http://creativecommons.org/licenses/by-sa/3.0/).

7 Contact

If any questions arise, do not hesitate to contact Eva Pettersson, by e-mail: eva.pettersson@lingfil.uu.se.

References

[1] F. J. Och and H. Ney. Improved statistical alignment models. In *ACL00*, pages 440–447, Hongkong, China, October 2000.

[2] Eva Pettersson, Beata Megyesi, and Joakim Nivre. Normalisation of historical text using context-sensitive weighted Levenshtein distance and compound splitting. In *Proceedings of the 19th Nordic Conference on Computational Linguistics*, 2013.

[3] Eva Pettersson, Beáta Megyesi, and Joakim Nivre. A multilingual evaluation of three spelling normalisation methods for historical text. In *Proceedings of the 8th Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities (LaTeCH)*, pages 32–41, Gothenburg, Sweden, April 2014. Association for Computational Linguistics.

[4] Eva Pettersson, Beáta Megyesi, and Jörg Tiedemann. An SMT approach to automatic annotation of historical texts. In *Workshop on Computational Historical Linguistics, NoDaLiDa*, 2013.

[5] Silke Scheible, Richard J. Whitt, Martin Durrell, and Paul Bennett. A Gold Standard Corpus of Early Modern German. In *Proceedings of the 5th Linguistic Annotation Workshop*, pages 124–128, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[6] Wolfgang Teubert (ed.). German parole corpus. Electronic resource, Oxford Text Archive, 2003.