IMPACT Lexicon cookbook

How to build and deploy a lexicon using tools developed in work package EE2: Lexicon structure and tools

- D-EE2.4 Practical guidelines and toolbox for building lexicon content
- D-EE2.5 Toolbox for lexicon deployment in enrichment

Document history

Revisions

Version	Status	Author	Date	Changes
1.0	Final	INL Impact team, UIBK (appendix keying instructions)	22 Feb 2010	Created and submitted to EC
1.1	Draft	Tom Kenter	31 May 2010	Updated with new features for the Lexicon Tool. Plus some notes on installing the tool on Ubuntu Linux.
1.2	Draft	Tom Kenter	6 July 2010	Update in documentation for Attestation Tool matching software.
1.3	Draft	Tom Kenter, Adrienne Bruyn	20 Sept 2010	Update in documentation for Lexicon Tool.
1.4	Draft	“	18 Oct 2010	Update in documentation for Lexicon Tool.
2.0	Final	Frank Landsbergen, Katrien Depuydt, Jesse de Does	18-22 Feb 2011	Update in documentation for Lexicon Tool Part II added on NE lexicon + tools
2.1	Draft	Frank Landsbergen, Tom Kenter	Nov 2011	Update tool description + evaluation NE tools
3.0	Final	Katrien Depuydt, Jesse de Does	Dec 2011	Review and adaptations, added section on morphogical analysis

Approvals

Version	Date of approval	Name	Role in project	Signature
1.0	1 March 2010	Clemens Neudecker, Max Kaiser, Hildelies Balk	Interoperability Manager, SP EE leader, Project Director	OK
2.0	22 February 2011	WP EE3 members	Internal review	OK
2.0	4 March 2011	Max Kaiser, Hildelies Balk	SP EE leader, Project Director	OK
3.0	December 2011	WP EE3 members	Internal review	OK
3.0	23 March 2012	Max Kaiser, Hildelies Balk	SP EE leader, Project Director	OK

Distribution

Version	Date of sending	Name	Role in project
1.0	23 February 2010	Clemens Neudecker, Max Kaiser, Hildelies Balk	Interoperability Manager, SP EE leader, Project Director
1.0	1 March 2010	Liina Munari	EC Project Officer
1.1 – 2.0	31 May – 22 Feb 2011	WP EE3 members	Internal reviewers
2.0	1 March 2011	Max Kaiser, Hildelies Balk	SP EE leader, Project Director
2.0	7 March 2011	Liina Munari	EC Project Officer
2.1 – 3.0	Nov-Dec 2011	WP EE3 members	Internal reviewers
2.0	23 January 2012	Max Kaiser, Hildelies Balk	SP EE leader, Project Director
2.0	6 April 2012	Liina Munari	EC Project Officer

IMPACT is supported by the European Community under the FP7 ICT Work Programme. The project is coordinated by the National Library of the Netherlands

Table of contents

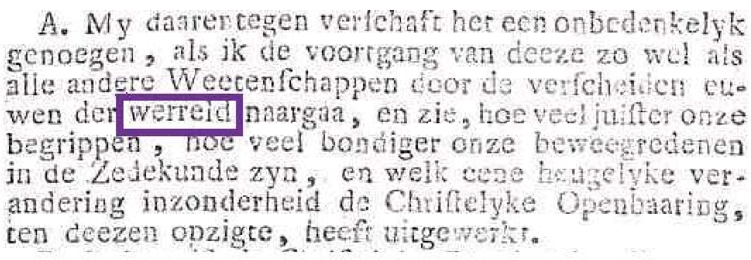
Introduction

1. Introduction: improving access to historical documents

IMPACT is a project funded by the European Commission. It aims to significantly improve access to historical text and to take away the barriers that stand in the way of the mass digitization of the European cultural heritage. For that IMPACT wants to improve the quality of OCR (Optical Character Recognition) for historical documents and to enhance their accessibility. There are many aspects involved in dealing with this problem which are addressed by IMPACT. Image processing, which tries to remedy typical problems like skewed, warped or otherwise noisy data; better segmentation procedures and adaptive OCR aim to overcome the irregularities of historical typography.

Full-text accessibility for historical text documents is also hindered by the historical language barrier. Historical language is not only a problem for text recognition, but also for users wanting to access the texts. How are they to find the necessary information, without having to take into account all possible spellings and inflections of words?

The following picture exemplifies the problem:[1]



A. My daarentegen verſchaft het een onbedenkelyk genoegen, als ik de voortgang van deeze zo wel als alle andere Weetenſchappen door de verſcheiden eu-wen der werreld naargaa, en zie, hoe veel juiſter onze begrippen, hoe veel bondiger onze beweegredenen in de Zedekunde zyn, en welk eene heugelyke ver-andering inzonderheid de Chriſtelyke Openbaaring, ten deezen opzigte, heeft uitgewerkt.

The variant form '*werreld*' poses a problem for text recognition (the recognition process will have to recognize this as a valid word; in fact Abbyy FineReader Engine 9.0 recognizes 'werreid') and retrieval: the user should be able to key in 'wereld' and find 'werreld' and other variants of this word.

> werelt weerelt wereld weerelds wereldt werelden weereld werrelts waerelds weerlyt
> wereldts vveerelts waereld weerelden waerelden weerlt werlt werelds sweerels zwerlys
> swarels swerelts werelts swerrels weirelts tsweerelds werret vverelt werlts werrelt
> worreld werlden wareld weirelt weireld waerelt werreld werld vvereld weerelts werlde
> tswerels werreldts weereldt wereldje waereldje weurlt wald weëled

To improve OCR, the OCR engine needs wellsuited historical lexica, with vocabulary and spelling corresponding to the language of the text that is to be digitized, i.e. an OCR **lexicon**. The OCR lexicon is also used in OCR post correction. To improve retrieval, a solution is to use a computational historical lexicon, supplemented by computational tools and linguistic models of variation. This type of lexicon, which we call IR **lexicon**, lists historical variants (orthographical variants, inflected forms) and links them to a corresponding dictionary form in modern spelling ('*modern lemma*'). In IMPACT, the OCR lexicon can be an extraction of data from the IR lexicon.

[1] De Denker 1, 1763 <www.dbnl.org>

2. Using historical lexica and linguistic models to improve text recognition and accessibility

Two simple examples from the WNT[2] give an indication of the kind of historical language variation we are up against.

Lemma UITERLIJK ('exterior')

uytterlijcste uyterlijkste d'uyterlijke uiterlyke uyterlijcke uiterlijke uyterlijck uiterlyken uiterlijkste uiterlicke wterlicke wterlijcke ulterlijk uiterlyk uiterlijk uyterlick wterlicken d'uyterlijcke uiterlijken uiterlijks wterlijck uytterlicke uitterlijke ujterlijke uytterlijk uyterlycke uyterlicken uijterlicke d'uiterlijcke wtterlijcke wterlyke wtterlijk (uiterlijke uuterlick uuterlic uyterlijke uyterlijcken uyterlicke d'uiterlyke wterlijke vuyterlijcke uuterlycke uuterlicke wterlijken uyterlijcksten uuyterlicke uuyterlick uuyterlycke uytterl uytterlijcke uytterlycke uytterlick vuytterlicke uiterlijker uyterlyck uterliek wterlijcken uiterlijkst uitterlijk uytterlijcken uyterlyk uiterlijk-net wterlick uutterlijck uuyterlicken uyttelijck uijterlijk uytterlijck uuterlijck uiterlick uitterlyk uuyterlic uuyterlyck uuyterlijck uiterlijck uytterlyck uterlyc wterlijk

Lemma WERELD ('world'):

werelt weerelt wereld weerelds wereldt werelden weereld werrelts waerelds weerlyt wereldts vveerelts waereld weerelden waerelden weerlt werlt wereids sweerels zwerlys swarels swerelts werelts swerrels weirelts tsweerelds werret vverelt werlts werrelt worreld werlden wareld weirelt weireld waerelt werreld werld vvereld weerelts werlde tswerels werreldts weereldt wereldje waereldje weurlt wald weëled

A few orthographical rules would obviously suffice to account for a large part of the variation encountered in the first example. This example also makes clear that for longer words, we can hardly hope to list all variants extensively in the lexicon with reasonable effort. Accounting for the variants in terms of orthographical rules is less obvious for the second example: many variants are largely unpredictable and can only be dealt with by listing them in the lexicon. This is why both linguistic modeling and extensive data development are essential to deal with historical language.

3. General lexicon vs. Named Entity Lexicon.

In IMPACT, we focus on two different types of lexica: the general lexicon and the Named Entities Lexicon. Named entities (NE) are specific words or word groups that refer to a single particular item in the real world, eg. *Amsterdam* is a location, *Silvio Berlusconi* a person name, *United Nations* an organisation. Named entities behave differently in terms of variation, hence the need to work on an attested lexicon, to be able to get a good view on this aspect.

4. Structure of this document

This document has two main parts. Part I deals with building 'general lexica', i.e. lexica of common words. Part II deals with building NE lexica. There are two appendixes.

[2] *Woordenboek der Nederlandsche Taal* (cf. <http://gtb.inl.nl>).

Part I: Building ‘general lexica’, i.e. lexica of common words (D-EE2.4 and D-EE2.5)

I. Data selection for demonstration, testing and evaluation and lexicon building

1. Selection of the text material the IMPACT lexica and language tools will be applied to

Lexicon building makes no sense when there is no correlation between the built lexica and the datasets the lexica and tools will be applied to. Within the IMPACT project, the result of the efforts put into lexicon building for OCR and retrieval will have to be demonstrated on a text collection coming from the different libraries involved. Each involved library has created an institutional dataset with text collections representative of their own library collection. From the point of view of language tools and lexica, the general requirements for the dataset are:

- The material should be challenging from a linguistic point of view: it makes no sense to build historical lexica if the language and spelling in the historical documents is not or only slightly different from modern language since in that case modern lexica can be used.
- The material should not be too challenging from an OCR point of view. If the current OCR is too bad, little or no use is to be expected from the application of lexica and language tools. To test this hypothesis, in the evaluation of the OCR, a random set from the entire GT collection has been used.
- There should be a substantial set of pages per text collection available.The material should be delivered in the form of images (i.e. the scanned pages), OCR’ed output on the scanned pages and metadata (author, data, editor, image quality, library data, etc.).

Important for any new library and linguistic partner within the IMPACT project building a lexicon: check also the amount of available data for lexicon building. This might also be a - though very practical - criterium to go for one particular time period or text collection and not for the other.

2. The creation of ground truth data

2.1 Background

In the IMPACT project, huge amounts of historical texts are being digitized in order to preserve them and to make them searchable. Linguistic tools, such as information retrieval software, named entity recognizers, lemmatisers etc. are needed to process these text data. Many of these tools need to be adapted to historical language use. Also, they need to be robust in order to deal with, among other things, OCR errors.

In order to do this, we need ground truth text data. Ground truth text data is text data without OCR errors. Having both an OCR’ed text, and its ground truth version, we can model the behaviour of OCR errors. Furthermore, we can determine the upper performance limit for the linguistic tools mentioned above.

Besides serving as a source for OCR error models, and acting as a means to determine an upper performance threshold for linguistic tools, ground truth data has more uses:

- investigating differences in the output of linguistic tools (the perfect GT data vs imperfect OCR’ed data) in order to determine the causes of a lower performance

- as benchmark data to reliably measure and evaluate OCR performance
- as a source for increasing robustness of linguistic tools
- as a source for 'perfect' wordlists, word n-gram lists and named entity lists. The lists, OCR lexica, are planned to be plugged into the OCR engine later on in the project.

2.2 Ground data selection

2.2.1 Available text material, illustrated by the use case for Dutch

The ground truth data has to be a selection of the institutional dataset, as mentioned in II.1. For Dutch a selection was made out of the following collections:

Name	Period
Staten Generaal [3] (parliamentary proceedings)	19th century, 20th century
Newspapers (Databank Digitale Dagbladen)[4]	1777-beginning of 20th century
Literature, name lists, reports, history (Dutch Prints Online [5](DPO)	1781-1800

Apart from the list above, we had also data available from DBNL (see www.dbnl.org): OCR'ed, manually corrected literary texts. We selected a corpus of about one million words of 18th and 19th century DBNL-texts as well for internal evaluation purposes.

2.2.2 Ground truth corpus selection: criteria, illustrated by the use case for Dutch

The Dutch ground truth corpus was made out of a selection of files from the raw text corpus described above. We selected a number of images (i.e., scanned pages of text), and sent the images to a company who manually rekeyed the images for us. We used the following criteria for the ground truth data selection:

1. the ground truth corpus should contain enough data to model spelling variation and other variations such as OCR misrecognitions.
2. the ground truth corpus should contain enough data to train (primarily probabilistic) linguistic tools
3. the ground truth corpus should reflect enough diversity in the text data to measure the impact of factors such as genre, time period
4. INL focuses on the 18th and 19th century, so the ground truth corpus should only contain text files from that period

In the case of Dutch, we also decide to ground truth the gold standard Named Entity tagged corpus we made with OCR'ed material (see D-EE 2.3), since this was for this toolbox an option to start work without the necessary ground truth material available.

[3] http://www.statengeneraaldigitaal.nl/

[4] http://www.kb.nl/hrd/digi/ddd/index.html

[5] http://www.dutchprintonline.nl/en/index.php

The selection and rekeying process produced a 13.1-milion-words ground truth corpus, consisting of literary texts ("Book"), Parliamentary proceedings ("Staten Generaal"), and Newspaper texts:

Type and genre	*# words*
Gold Standard Book	300k
Random Set Book	340k
Random Set Staten Generaal	2.5M
Gold Standard Staten Generaal	500k
Gold Standard Newspapers1	3.4M
Gold Standard Newspapers2	170k
Random Set Newspapers	3.2M
Total	*13.1M*

Each of these three sub corpora contains a 'gold standard' set, which means that for each file in this set, a manually verified, named entity-tagged version is available. The rest of the sub corpus is the 'random set': files, randomly picked from the text corpus, but not already present in the gold standard set. Newspapers1 contains articles of running text, whereas Newspapers2 contains ads.

As said, the images were rekeyed manually. The selected provider delivered the 13.1-million-words corpus well within two months with an average accuracy above 99.95% (which is at most 5 typos per 10000 characters).

2.3 Keying instructions

After selection of the data, keying instructions have to be formulated for the keying company. For the Dutch dataset, instructions were written for books, newspapers and parliamentary papers by INL and UIBK, coordinator of the ground truthing within IMPACT. These instructions can be found in appendix 1 of this document. Later on in the project, the instructions were adapted by UIBK. The XML format was changed to the PAGE XML format[6] developed by the Pattern Recognition and Image Analysis (PRImA) Research Lab.

2.4 Some useful suggestions as to data delivery

- As early as possible in the project, get an overview from your text material provider of the text material that is, and/or will be available to you. Make sure that it is clear to both you and the provider how much, and what kind of text data you will receive on which dates.
- The smoother the text material delivery process, the better. Try to get batches of equal size from your provider
- Set up an administration (a metadata database, and a delivery scheme) of the text material you receive. It is not uncommon to receive overlapping text material, or updates of the same text material. Queries like these should be easy to retrieve: 'which files did we receive on <date>'? or ‘give me all newspaper texts we have received until now, issued between 1820 and 1823’.

[6] http://www.primaresearch.org/papers/ICPR2010_Pletschacher_PAGE.pdf

3. Guidelines for the selection and acquisition of resources for lexicon building

There are different sources from which lexicon building may start: A lemma lexicon (list of lemma's, for instance the entry list of a historical dictionary); a full form lexicon (list of lemma's with their paradigmatic word forms); historical text, untagged; historical text, lemmatized; historical text, with part-of-speech tags. Also needed is an existing modern full form lexicon or at least a lemma list. And obviously any already existing historical lexicon that assigns modern lemmas to historical full forms of sufficient coverage is useful. This means we have to work with historical corpora, electronic historical dictionaries and computational lexica.

Apart from the fact that it is most preferable that the materials for lexicon building are free of use at least within the IMPACT project, the data have to meet with the following specific requirements:

- *Historical corpora:*
  - have to be of ground truth quality, meaning either keyed or OCR'ed with post-correction, yielding an accuracy of above 99,95% character recognition rate
  - have to be of sufficient size (at least some millions of tokens, balanced as to content)
  - have to correspond to the document collection used in the demonstrator
  - preferably in UTF-8 since the tools built within the project assume UTF-8 input
  - if possible: lemmatized with modern lemma
- *Electronic historical dictionaries*
  - have to be of ground truth quality, meaning either keyed or OCR'ed with post-correction, yielding an accuracy of above 99,95% character recognition rate
  - have to describe the language period corresponding to the document collection used in the demonstrator
  - need to have a sufficient level of XML encoding to at least extract the headwords preferably with part of speech, and preferably also the quotations, containing example material in the original historical spelling (examples are: The *Oxford English Dictionary*[7], the *Woordenboek der Nederlandsche Taal* ('Dictionary of the Dutch Language)[8], the *Deutsches Wörterbuch* by Jacob and Wilhelm Grimm[9])
  - preferably have some structured description of the inflectional paradigm
- *Computational lexica*
  - a modern lexicon is necessary, a historical one would be of great help but not a condition sine qua non
  - have to be of sufficient coverage
  - are preferably full form lexica (lemma's + paradigmatic word forms)

Historical corpora can be used for lexicon content, as well as the electronic historical dictionaries of which not only the lemma's can be used, but also the material in the citations. Modern lexica can be used when a set of patterns describing the relation between historical and present-day orthography is available.

[7] http://www.oed.com/
[8] http://gtb.inl.nl/
[9] http://germazope.uni-trier.de/Projects/DWB

The best place to look for these materials are research institutes dealing with corpus building, working on dictionaries and/or on computational lexica (examples for France: ATILF (Nancy), Spain: Cervantes digital library (Alicante), Germany: IDS (Mannheim)). But also the internet can provide useful data. Think of projects like Gutenberg (www.gutenberg.org), or the Digital Library of Dutch Literature (DBNL; www.dbnl.org). But also national libraries, like e.g. the BNF (Bibliothèque Nationale de France), sometimes have data of ground truth quality for lexicon building, or cooperate with projects that provide for such quality data (e.g. the British Library in the Eebo-project (Early English Books online; http://quod.lib.umich.edu/e/eebo/).

All these materials can be used to build a general lexicon of a particular language. We will go into the materials for building NE lexica in the next release of this cookbook.

Special case: LMU / BSB: creation of ground truth quality data for lexicon building and evaluation[10]

In the IMPACT project, BSB chose to tackle part of its collection of 16th century German prints. Since not enough material for corpus building was available, a special procedure was followed.

For corpus based lexicon building and benchmarking, materials of the Early High German period, 16th century, were selected exclusively from a focus area of theological documents. This focus was specified by BSB based on digitization and presentation projects scheduled for the next years. The benchmarking documents were selected according to experiences with the IMPACT Random Data Set.

This was done according to the following procedure:

- First step was a random selection of more than 200 titles from the 16th century falling into the decided focus area "Theology". The documents were accessible through a BSB call number. Due to work package constraints was necessary to narrow down the scope by focusing on only one subject. This was arranged according to a joint meeting CIS/LMU, INL, BSB in Munich, 30/07/08.
- All Latin materials were excluded and 100 titles in German language were chosen
- Two pages from each of the selected 100 titles were selected and processed with ABBYY FineReader version 7.1 with options Gothic, Old German
- For keying of complete volumes the documents with an acceptable recognition performance (estimated ~ 70% or better) were chosen. Aim was to collect complete books often smaller than 50 pages for the 16th century. These books will be used for lexicon building. Altogether 1766 pages from 84 works were selected.
- Additionally, the used 200 random pages of the 100 titles were selected for ground truthing to have a benchmark set for lexicon development. These materials of the 16th century will establish the largest digitized corpus and evaluation set available for research on digitization of Early High German so far.

[10] With thanks to Clemens Neudecker (BSB, Bayerische Staatsbibliothek, Munich) and Christoph Ringlstetter (CIS (LMU), Munich).

II. On the linguistic annotation of the lexicon: lemmatization, attaching a modern lemma to historical words

In the lexicon, for the purpose of better word look up (basic level of information retrieval), to each historical word form a modern lemma is attached. This lemma, in combination with part of speech, is the key under which variants (spelling variants (*wereld/werreldl*), inflectional variants (*wereld/werelden*) and words written together or split (*swerelds; we reld*) are grouped. The lemma is not a version of the historical word form in a text in modern spelling, but an uninflected modern form corresponding to the historical word form as one would find as an entry in a traditional dictionary.

Important for this task is that these modern lemma's are written according to the current orthographical rules of the language. For Dutch the rules have been published officially and rules and a lexicon illustrating them are found on http://www.woordenlijst.org.

Specific rules for dealing with historical language

Not all historical words of a language have survived. In those cases, the modern lemma is reconstructed, meaning, that linguistic (etymological) rules are applied to create a modern lemma as it would most probably have looked like if the word would have survived in present day language. The lemmatization is based on etymological grounds, NOT on semantic grounds. Meaning that for a historical word in Dutch *mersch* ('meadow'), the modern lemma *meers* is constructed, and not the current Dutch word *weiland* or *weide*, semantically equivalent, but not from an etymological point of view.
Useful: most of these particular historical words have one or more elements of words that do have survived. This is particularly so in case of compounds and derivations. It is therefore useful to provide for a list of affixes and their modern equivalent to be used, and also to check whether parts of a particular word can already be found in the lexicon under construction.

Very regularly, words in historical documents are 'glued' together, meaning that they are attached to one another instead of written separately. This does not only apply to the traditional clitics (*swerelds* 'of the world', a combination of an article in the genitive and the noun in the genitive), but to any combination of words. When lemmatizing these particular word forms, the word form is NOT split up, but a double lemma is attached (*swerelds* lemma: DE_WERELD 'the_world'). The lexicon structure allows this.

It is also very well possible that one will find a single word, split up in parts and separated by one or more spaces. This might be the case for the traditional separate verbs (*Ik **geef** twee boeken **weg*** (I give away two books – Dutch order: I give two books away) in which ***geef … weg*** should be lemmatized like WEGGEVEN), but can also happen very randomly *(eg. febru ary* (February) – *an actual example in Dutch historical material*). When lemmatizing these particular word forms, the word form is NOT glued together, but a lemma is attached to the combination part 1#part 2 and it will be stored as such in the lexicon (*e.g. febru ary* Lemma FEBRUARY).The lexicon structure allows this.

III. Tools for lexicon building and lexicon deployment: general remarks

1. Requirements for the IMPACT lexica and linguistic tools

Our aim is to develop historical lexica combining scholarly precision with broad coverage for use in digitization (for both text recognition (TR5) and enhanced retrieval (EE2/3), and to deliver guidelines and a set of tools for the efficient production and deployment of such lexica.

This particular application imposes a few requirements:

- First, the lexica need to allow for specialization to periods (for instance, *waereld* should not be included for OCR of texts after 1850). An unstructured, ever-growing set of word forms, without information about the kind of text (in terms of period) in which we can expect the words to occur, is neither usable in text recognition nor in enrichment. Frequency information, essential in OCR, will also be added to the lexicon.
- Second, the lexica should be suitable for retrieval in applications for the general public by providing 'modern' query terms to search for historical variants (*use 'wereld' to search for all variants*).
- Lexica used for OCR and retrieval are necessarily incomplete due to the immense amount of possible orthographic variants found in the texts. Hence they need to be complemented by linguistic tools and models to deal with this problem[11].

Since the computational linguistic tools are developed within the context of a European project focusing on mass digitization of historical text, they should be language-independent (generic) whenever possible, and fit to quickly process large quantities of data.

The fact that linguistic modelling cannot account for all variants entails that the tools should part of a lexicon development workflow involving both automatic and manual processing[12].

2. Corpus-based lexicon structure

The core objects in the lexicon structure developed for IMPACT are word forms, lemma's and documents. All other objects define some kind of relation between these.

In order to enable the OCR's spellchecking mechanism to assess the plausibility of the occurrence of a word in a certain text, it is not sufficient to convert existing lexica and dictionaries into a large word list. We also need to

- keep track of the sources from which we took the words;
- list the words actually encountered in the language and record occurrences in actual texts, with frequency information (attestation);
- record in what kind of texts these words occur (document properties).

It is impossible to extract all possible word forms from the limited amount of available reliably transcribed historical text. Hence, we need mechanisms to extend the lexicon and to enable us to assess the plausibility of '*hypothetical*' words without previous

[11] The tools and models deal mainly with variation of the 'predictable' type (cf. *uiterlijk* above).

[12] In order to deal with variation of the second type (cf. *wereld* above).

attestations, i.e. words we have not seen before. Supporting data for these mechanisms have to be present in the database:

- unknown inflected forms of lemma's which already are in the database can be dealt with by means of the automatic expansion from the lemma to the full paradigm of word forms (paradigmatic expansion);
- new spellings of known words can be dealt with by developing a good model of the spelling conventions of the period at hand. The database structure provides for the storage of orthographic variant patterns;
- previously unseen compounds can be dealt with by means of a good model of word formation.

In order to effectuate word searches without having to worry about inflection and variation of word forms, enrichment will use 'modern lemma's' as variation-independent retrieval keys for the full spectrum of inflectional and orthographical variation.
The database structure is divided into a few main blocks:

- Information attached to word forms, either unlabelled (i.e. not yet lemmatized or labelled with Part of Speech) or labelled (i.e. with lemma and possibly PoS).
- Information attached to the lemma's.
- Information about documents, parts of documents, document collections.
- Auxiliary information needed for expansion and for plausibility-of-new-words prediction.
- Lexical Source.

Hence, to each labelled or unlabelled word form, we link *attestation* objects which are basically just verified occurrences of the words in documents. The attestations enable us to derive the relevant information about the domain of applicability of word forms from the properties of the documents they occur in. When a word form is taken from a lexicon or dictionary, or when it originates from automatic analysis expansion, we also keep track of its provenance. Apart from the link to the relevant word form and a location in a document, the attestation objects contain the following information:

- verification (yes/no): Whether the occurrence of a labelled word form is checked manually by an expert;
- frequency in a document or document collection.

Two distinct kinds of attestation may be relevant: we may just link a word form to a document, recording the frequency of occurrence ('attestation at text level'), or we may link to an individual occurrence of the word ('attestation at the token level')[13]. The latter kind of attestation is especially relevant to tagged corpora. In the lexicon building workflow, lemma's may first be assigned on the text level, and ambiguity is not completely resolved. At a later stage, ambiguity may be resolved by assigning lemma's on the token level.

The lexicon structure of the IMPACT lexicon is described in D-EE2.1. Two XML export formats have been defined:

1. An export in the LMF format, including a tool to convert the lexicon database output into LMF (Lexical Markup Framework)[14].
2. An export (and export tool) to a TEI p5 format defined by Tomaž Erjavec. As several of the language partners have argued for the choice of a TEI- based format, we exported all the delivered final versions of lexica to TEI.

[13] A type is a word form, a token is a particular instance (occurrence) of the type in a text.
[14] http://www.lexicalmarkupframework.org/

IV. Recipes for lexicon building and deployment

This section describes the two major recipes for lexicon building resulting in attested word forms, involving different data sources for lexicon development and the tools as described in section VI. Our purpose is, in both cases, to build a diachronic word form lexicon that contains spelling variants and morphological variants of words that have appeared in documents over a certain period.

Some important properties of the resulting word form lexicon are:

- it contains the modern lemma corresponding to the historic word form;
- it provides attestations representing genuine usage of the words in historical texts;
- the attestations have bibliographical information, including date.
-

There are several ways to build such a lexicon starting from language data like a diachronic corpus, a modern full form lexicon, and a historical dictionary. In this section several approaches (recipes for lexicon building) will be described. The tools referred to including user manual, requirements and installation guide, are listed in section VI.

1. Building an attested word form lexicon using historical dictionaries

1.1 Introduction

Historical or diachronic scholarly dictionaries tend to include numerous quotations from different periods illustrating the usage of words in historic texts. The main idea is to use these dictionary quotations and the associated bibliographical information as attestations of word forms. These quotations exemplify the usage of the head word of a dictionary item; the lemma. Usually the word form in the quotation which corresponds to the lemma is not explicitly marked in the digital versions of the dictionary. In this section we describe a method to match the lemma to the corresponding word form in each quotation. This method consists of two separate processes. First, we apply automatic preprocessing to select the most probable candidate word form in the quotations. The results are stored in a database. Secondly, the results are manually verified and corrected using a specially designed tool. Cf. figure 1. This approach is a quick way to start building a lexicon. Only simple matching procedures are needed to match the occurrences of the headword of each dictionary article into the citations. Moreover, these word forms are already grouped under a lemma.

IMPACT is supported by the European Community under the FP7 ICT Work Programme. The project is coordinated by the National Library of the Netherlands

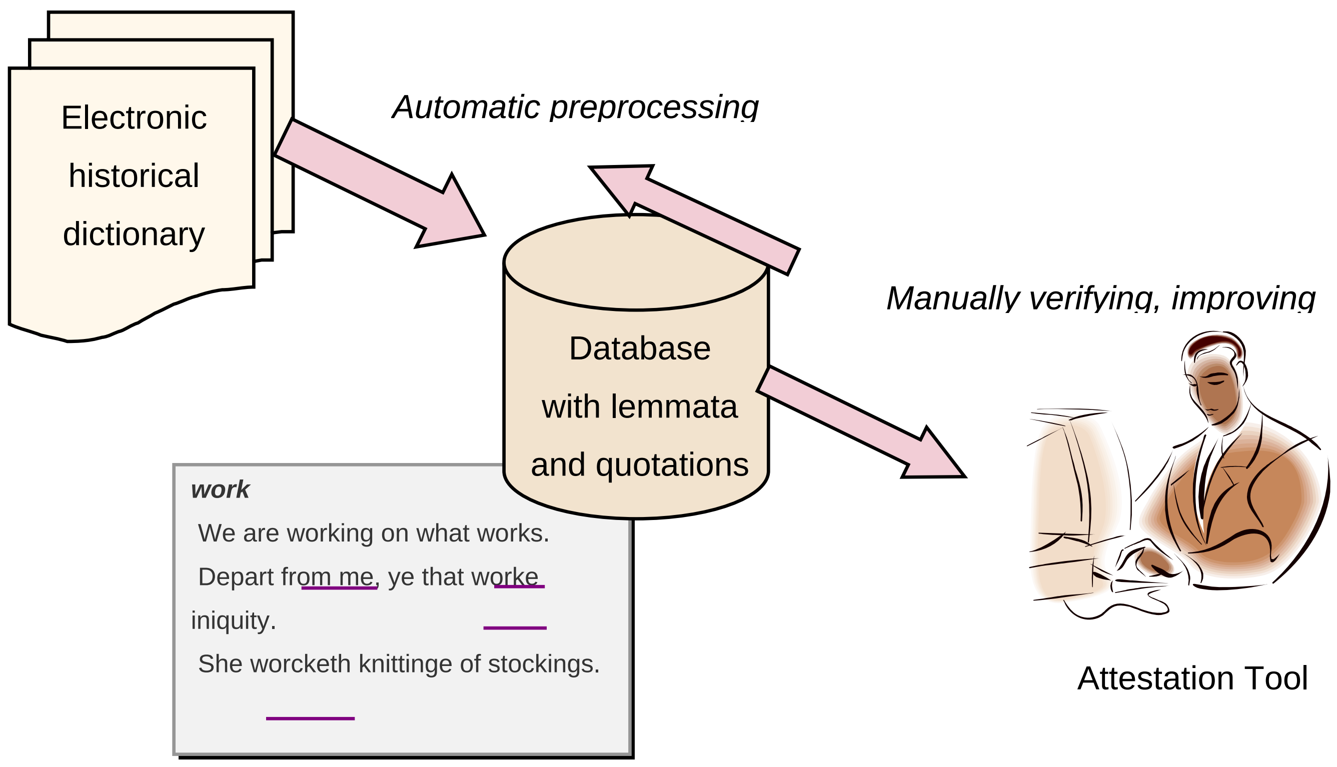

Electronic historical dictionary
Automatic preprocessing
Database with lemmata and quotations
Manually verifying, improving
work
We are working on what works.
Depart from me, ye that worke iniquity.
She worcketh knittinge of stockings.
Attestation Tool

Figure 1

First we will describe the general approach to the building of a word form lexicon from a historic dictionary, and then, a description will be given of the application of this recipe and the tools for constructing the Dutch general lexicon by processing the WNT ("Woordenboek der Nederlandse Taal" – Dictionary of the Dutch Language). The IMPACT general IR lexica for English and Polish were built in the same way using the Oxford English Dictionary and the *Dictionary of 17th and early 18th century Polish* (Słownik języka polskiego xvii i 1. połowy xviii wieku).

1.2. Recipe for a diachronic word form lexicon

In this chapter we describe the essential operations to find word forms in quotations that correspond to the lemma in a dictionary entry. Of course, there are many additional operations required, like parsing the dictionary data and isolating the lemma and the quotations in that data. This document assumes that the dictionary is available in digital form and that the task of parsing the basic article structure has already been performed.

In selecting the word forms from the quotations we distinguish two consecutive operations: first an automatic matching operation, using the CitAttest-tool (VI.2) and second a manual correction of the result of the first operation, using the Dictionary Attestation Tool (VI.3).

1.2.1. Automatic preprocessing

Two main complications in finding the proper word form in the quotation are morphological variations and spelling variations.

1.2.1.1. Morphological variation

The lemma of a dictionary entry is generally written in a canonical form (e.g. infinitives for verbs and only singular nouns). The word form in the quotation however quite often is an inflected form. The dictionary entry may contain some information on morphology but often the description is limited. It is, therefore, useful to expand the lemma with full paradigms from a separate lexicon if that is available (in case of the WNT the Dutch electronic lexicon e-lex was used). Additional historical variants can often be obtained from the dictionary itself. It may be fruitful to compile some lists by hand.

1.2.1.2. Matching spelling variation

The spelling of words in historical texts can differ widely from modern spelling. There are two general approaches to match different spellings. First, it is possible to use rewrite rules that transform words in one spelling to another. For historical dictionary which covers a large timespan, and in which variation is not limited to orthography, this approach is not satisfactory. Therefore, the use of statistics is often needed.

A commonly used statistic describing the match between two strings is the Levenshtein Distance[15]. It describes the number of character operations (inserting, deleting, changing) necessary to change one string into the other.

In the matching software of the Attestation Tool, words are considered variants of each other when they have a distance smaller than the length of the shortest word divided by 3. This distance is the aforementioned Levenshtein distance, plus 1 extra if the words differ in their first character. Furthermore, words shorter than six character are not allowed to differ in their first character and words of up to three characters in length are not allowed to differ at all.

So, e.g. why doesn't the word *appules* match with the word *apple*?
The maximum distance allowed is the length of *apple* (5) divided by 3 (1.66667). There are two additions however (the *u* and the *s*) so the distance is 2, which exceeds this maximum.

1.2.1.3. Algorithm

The basic matching operation is quite simple.

We start from a set of already known lemma variants of the dictionary headword, together with an initial set of inflected forms at the one hand (“keyword list”) and a dictionary quotation on the other hand. Note that the keyword list is a list of lists: it contains a list of inflectional forms for each possible lemma variant.

The first step is to tokenize the quotation. Next we compare each element from the token list with each element in the keyword list. In a three-layered loop we make all the comparisons. The results are stored in a database.

[15] Cf. http://www.levenshtein.net/

1.2.2. Manual verification operation

If the matching has been based on statistical methods, it is advisable to have these checked manually. Moreover, in this stage it is possible to address complex matches that were not covered in the automatic process, like clitic combinations, and historical, non-standard word splitting conventions. Manual verification can be done with the Dictionary attestation tool (VI.3).

1.3. Use case: Extracting a diachronic word form lexicon from the WNT

In this section the steps involved in extracting a word form lexicon for Dutch from a diachronic dictionary are documented.

The dictionary that we have used to extract attestation from is the *'Woordenboek der Nederlandsche Taal"* (Dictionary of the Dutch Language, for short; WNT). This is a diachronic dictionary that has been published in the period 1864 - 2001, and has since been digitized; it is marked up according to a proprietary XML format. The dictionary contains approximately 1,5 million quotations. These quotations span the period of 1500 to 1976 and are dated for a large part. These quotations, therefore, give us access to word forms that are dated and that have a relation to a lemma.

Unfortunately, the keywords in the quotations are not specifically marked. Therefore, the main task at hand is to find the word forms in the quotation that correspond to the keyword of the article. We will tackle this challenge by automatically selecting likely candidates from the quotation and checking all these selections by hand.

In V 1.2 we discussed in general terms how this task can be performed; in the present section we will go into the details that are particular for this dictionary and the Dutch language. Moreover, we will describe the particular tools that we have developed for that purpose and indicate how they can be adjusted to another job (other language, other dictionary).

As already explained in the previous chapter, we employ a two-step process in which first a database is filled in an automatic procedure and, second, the content of the database is manually verified. The tools referred to in this section are IMPACT-Tok, a tokenizer, CitAttest, a collection of scripts to attest word forms in dictionary quotations and the Dictionary Attestation Tool, a GUI for manual correction.

Note that a special challenge for the WNT was, that parsing of the dictionary quotations is still a work in progress. Therefore, functionalities like marking ‘bad citations’ were implemented into the Dictionary Attestation Tool, and revision of data enabled.

1.3.1. Automatic preprocessing

1.3.1.1. Lemma selection

There are a number of complications involved in matching lemma and word forms. In the case of the WNT, the first hurdle is to select the proper lemma (key word) from the dictionary. As was already mentioned, the dictionary has been compiled over a period of one-and-a-half century. Many conventions for constructing articles have been introduced and abandoned in these many years. Articles can contain a hierarchy of sub lemma’s in which the quotations are embedded somewhere; the correspondence between quotation and sub lemma is not always obvious.

For quotations in the related entry section of the article, we extract the most appropriate lemma for each quotation by means of a Perl pattern matching procedure which implements the following set of XPath expressions.

- 'parent::CIT/parent::*/HWOPN',
- 'parent::CIT/parent::P/parent::*/HWOPN',
- 'parent::CIT/parent::P/parent::BET/preceding-sibling::BET/P/HWOPN',
- 'parent::CIT/parent::P/parent::BET/preceding-sibling::P/HWOPN'.

The first match among these expressions is selected. As will be clear, the element 'HWOPN' contains the sub lemma keyword. If none of these expressions match for a quotation in the related entry section of the article, the quotation is skipped.

1.3.1.2. Separable verbs in Dutch

A complication in Dutch is formed by separable verbs. These are verbs consisting of two parts which in some cases are connected while in other cases they exist as separate words in a sentence. The verbs are prefixed with another word (a preposition most often) like 'uit' in 'uitkomen' (come out). In some conjugations the prefix becomes detached from the verb. In those cases we want to mark both parts as belonging to the lemma.

Information on separable forms is available in e-Lex[16], the lexicon used in producing morphological variations of the lemma, as well as in the header information from the dictionary entries.

When a separable verb is searched in the quotation, the normal matching algorithm is extended as follows: the basic matching operation is performed with the verbal part of the separated form. If there are good candidates for the verb, we also perform a basic matching operation for the separated parts (the prepositional parts). The prepositional parts however can end up both before or after the verb they belong to. Therefore, for every candidate verb the closest matching preposition is picked (either before or after) in terms of words in between.

Other non-standard word splitting phenomena, like clitic combinations, and non-standard orthography of compounds, however, are still unresolved by the automatic processing. Some of these will be addressed in the manual phase.

1.3.1.3. Initial set of variants

We have added lists to the paradigms of irregular historical forms of verbs, and function words. These lists have been compiled by hand, since no resources were available for Dutch containing this information. Furthermore, we added modern inflected forms from the e-Lex lexicon.

1.3.1.4. Adapting the Automatic Process

The implementation of the automatic process consists of a number of Perl scripts that read XML-data and write the results to a MySQL database (VI,2).

Main.pl handles the dictionary file and processes the articles one by one. The specific functions for handling the idiosyncratic WNT-structures have been collected in '*wnt_article.pm*'.The function 'do_wnt_file has to be adapted in order to parse the dictionary file into articles. The function select one article at the time, picks out the lemma and selects the proper inflected

[16] http://www.inl.nl/tst-centrale/nl/producten/lexica/e-lex/7-25).

forms from the database (see description of 'initialVariants.pm'). The function that essentially handles the matching is 'do_eg'. This function also handles the interaction with the attestations database (see 'AttestationBase.pm') in respect of revision management.

The package *InitialVariants.pm* builds a list of possible word forms derived from the lemma. Two sources are used for that: a (modern) lexicon containing complete paradigms and information on morphological variants from the dictionary. These sets overlap, but are not similar because the dictionary contains many lemma's that are not in the lexicon, and the lexicon contains many derived forms that are not in the dictionary.

HeadWordMatching.pm contains the functions that apply the actual matching algorithms. It is called from 'main.pl'. *AttestationBase.pm* is used to interact with the attestations database.

1.3.1.5 Building ones own preprocessing script

As mentioned above the matching software consists of a Perl script that uses several packages. To customize the software for a new dictionary a new Perl script could be made that prepares the data into the right data structures for these packages to use.

There are in fact only two sub routines that need to be called: HeadWordMatching::matchLem() and AttestationBase::saveQuote().

A typical script would look like this:

- initialize
- read and parse data
- for every lemma
    - gather the lemma head word and any variants that are listed
    - encapsulate these into the right data structure for matchLem()
    - for every quotation
        - call matchLem()
        - prepare the right data structure for saveQuote (using the Quote package)
        - call saveQuote()

Now, let's look at the sub routines in more detail.

HeadWordMatching::matchLem()

This sub routine matches the headword and its listed variants to the words in the quotation with a fuzzy matching algorithm designed especially for this task and described above (VI.1.2.1.2).

```
($bMatch, $aLemma, $arMatchedTokens, $arTokenizedQuote) = matchLem($sQuote,
$hrVariations);
```

The first argument ($sQuote) is simply the quotation as a string. The second argument ($hrVariations) is a reference to a hash that should look like this:

```
$hrVariants = {'apple' => [ ['apple'],
                            ['aepl'],
                            ['eappul'],
                            ['appil'],
                            etc...
                          ]
              }
```

So it is a hash with one key, which is the lemma head word. Its value is (a reference to) an array of arrays. Each array lists variant of the lemma headword, the first one being the lemma head word itself.

The array with variants can hold more than one word when we are dealing with multi word entries like e.g. compounds (*apple juice*) or seperable verbs (*give away*).

For e.g. *apple juice* the variant hash could look like this:

```
$hrVariants = {'apple juice' => [ ['apple', 'juice'],
                                  ['aepl', 'juice'],
                                  ['eappul', 'juice'],
                                  ['appil', 'juice'],
                                  etc...
                                ]
              }
```

You will notice that the second word in this case is the same every time. This need not be the case of course, but it usually is, as 'apple juice' will be a sublemma of 'apple'. So there will be variants for 'apple' available at this step, but not for 'juice' (which are listed at the lemma 'juice' if it exists).

matchLem() returns four things:

$bMatch:	a boolean indicating whether anything matched at all
$arLemma:	a reference to an array containing the lemma
$arMatchedTokens:	a reference to an array containing all the tokens that matched
$arTokenizedQuote:	a reference to an array of tokens representing the quotation

The tokens as mentioned above are arrays that look like this:

```
[
  12,        # Onset in the original text
  34,        # Offset in the original text
  'token',   # Normalized token
  'token,'   # Token as it appears in the text
  1          # The distance between the matched word and the variant it
             # matched with (only in the $arMatchedTokens array)
]
```

AttestationBase::saveQuote()

```
AttestationBase::saveQuote($oQuote);
```

saveQuote needs only one argument which is an instance of a Quote object. The Quote object is declared in AttestationBase.pm so one only has to call the constructor with the right arguments and fill the object with the right data.

```
my $oQuote = Quote::new($sLemmaId, $sQuotationId, $sQuote, $hrVariants);
$oQuote->{lemma} = $sLemmaHeadWord;
$oQuote->{tokenized_quote} = $arTokenizedQuote; # Available from matchLem()
$oQuote->{matched_tokens} = $arMatchedTokens;    # Available from matchLem()
$oQuote->{dateRangeFrom} = $sYear if(defined($sYearFrom));
$oQuote->{dateRangeTo} = $sYear if(defined($sYearTo));
```

saveQuote() is a method of the AttestationBase object, so also, somehwere at the start of the script when initializing, one needs to prepare one of these.

```
my $oAttestationBase =
    AttestationBase::new($sDbHost,
                         $sDbUser,
                         $sPassword,
                         $sDbName,
                         $bRecreateDatabase  # Boolean, usually a true value
                         );
```

With these two objects at hand the results of matching the head word to the quotation can be written to the database.

```
$oAttestationBase->saveQuote($oQuote);
```

When these steps have been performed a database should be filled with lemmata and attestations. These results can be viewed and edited by the IMPACT Attestation Tool (VI,3).

1.3.2. Manual correction

In order to apply the workflow described in section 1.2.2 to a specific dictionary like the WNT, one has to write guidelines describing in detail when a word form can be assigned to a lemma, and in which cases a quotation has to marked as ‘bad’ or ‘unfortunate’. The details are beyond the scope of this document

1.3.3. Managing revisions of the data

The data of the WNT is still under development. It, therefore, could be that the source data (in this case the XML data of the WNT) changes at a later stage and we want to make use of the updated data to improve the attestations database. In that case, the attestations database must also be revised because of the chance that quotations in the original data have been changed. This is accomplished in the by now familiar two steps:

1.3.3.1. Automatic processing.

Quotations in the text will be matched according to location (ID of the quotation section in which the quotation occurs) and content. First, for all records, if the value of variable 'update' is set to 0, the following logic is applied:

If there is a quotation in the WNT that has an ID that is not in the database, then a new entry will be added, and an appropriate candidate word form will be selected. (update is set to value ‘true’)

If the ID matches, but the quotation differs, a new (automatic) match is made and the entry in the database is replaced. (updated=’true’)

If the ID and quotation and head words match and the quotation is not manually revised, a new (automatic) match is made and the entry in the database is replaced. (the updated flag is set to ‘true’)

Otherwise, no automatic match is calculated. and the “updated” flag is set to ‘false’

In all cases presented above, the values of dateFrom and dateTo are checked and updated if there is a difference with the WNT data.

Further, all quotations for which the value of 'update' has remained NULL have no match in the new XML document. These quotations are deleted from the database. Also all entries from 'attstations’ that correspond with that entry are deleted.

1.3.3.2. Manual verification.

All records in the table “Quotation” now have either the value ‘true’ or ‘false’ in field “updated”. All records which have the value true should be reexamined manually using the Dictionary Attestation Tool (VI,3).

In total, 220211 lemma's with a total amount of 1524366 quotations were manually checked. The average speed of the users doing the correction was 1725 quotes/hour, 231 lemma's/hour[17].

2. Corpus-based Lexicon building using a full form lexicon and historical text

For many languages, we are in the situation that a good modern full form lexicon and historical corpus material are available, but there is no easy way to exploit existing historical dictionaries. Within the IMPACT project, this is for instance the case for German. Even when lexicon construction from a historical dictionary is possible, to obtain a focused lexicon for a certain type of text, the dictionary-based lexicon still has to be supplemented by corpus-based lexicon content.
This section describes a recipe for corpus-based creation of lexicon content. A significant part of all manual work involved in lexicon building is covered in this recipe. This means that this part of the workflow has to be extremely efficient. For this purpose, the IMPACT Lexicon Tool for Corpus Processing (section VI.6) can be used in conjunction with the tools for spelling variation (VI.4) and lemmatization (VI.5) to obtain a historical lexicon with attestations.

The main steps in corpus-based lexicon building are:

1) Process the material with the lemmatization tools to obtain, for each word form, if possible, either
- an exactly match with the existing historical lexicon content or with the modern lexicon, or
- an alignment of historical word form and modern word form supported by a matching algorithm with the patterns describing the historical orthography
Otherwise, the word will be labeled as unknown.

The output from this step looks like this:

```
Vaderlandsche {{vaderlands,vaderlandsche,ADJ}, HistoricalExact}
byzondere {{bijzonder,bijzondere}, ModernWithPatterns, 0.22, ^b[ij->y]zondere$}
DAADEN {{daad,daaden,NOU}, HistoricalExact}
slegt {{slecht,slecht,ADV(general,=)}, ModernWithPatterns, 0.056, ^sle[ch->g]t$}
uitgebreid {{uitgebreid,uitgebreid,ADJ}, HistoricalExact}
magtig {{machtig,machtig,ADV(pron,=)}, ModernWithPatterns, 0.056, ^ma[ch->g]tig$}
voorregten {{voorrechten,voorrechten,VRB}, ModernWithPatterns, 0.056, ^voorre[ch->g]ten$}
zugt {{zuchten,zucht,VRB}, ModernWithPatterns, 0.056, ^zu[ch->g]t$}
allerwoestste NoMatch
```

2) Load the resulting data into the lexicon tool database[18], add attestations to verify corpus occurrence and, resolve lemmatization ambiguities and add unknown words to the lexicon.

These steps are to be carried out iteratively. Figure 2 describes the acquisition process, in which not only the lexicon content grows, but also the model of orthographical variation can be retrained, adapting to new example material.

[17] For the Oxford English Dictionary, thanks to adaptations in the Tool and special settings for matching, we were able to handle 400-600 lemma's/hour

[18] Cf. the documentation, section VI.6. Once TEI import is implemented, importing the data will be much easier.

```
Initialization:
Lexicon :=  some full form lexicon, f.i. CISLEX for German, e-LeX for Dutch
Patterns := some initial set of spelling patterns (aa/ae), perhaps the empty set.

While (not satisfied with coverage of lexicon)
{
    Step 1. Process selected texts with lexicon and orthographical variant patterns.
    Step 2. Split the words from the texts in 3 subsets
    $W_1$ = exact match with lexicon
    $W_2$ = match with lexicon, using patterns (= match in the ‘hypothetical lexicon’
    $W_3$ = not found at all
    Step 3. Manual checking, using the corpus-based lexicon building GUI in combination with the context view for token-level
    attestations.
    - For w in $W_1$, possibly check ambiguous word forms for lemma assignment
    - For w in $W_2$,
        - check the matched lemma (e.g. word form: bieck/ lemma: bakken)
        - check the matched ‘normalized’ word form (e.g. historical bieck, normalized biek)
        now either:
        (i) The match ok (no action required)
        (ii) The match is not ok, but a match with existing lemma/normalized word form is possible (correct)
        (iii) No match is possible with existing data. (Move w to $W_3$)
    - For w in $W_3$, there are three possibilities.
    (i) w can be matched with lexicon, using hitherto an unknown pattern. In this case we manually add match with modern
    (normalized) word form, so the pattern inference tool $S_1$ can infer a new pattern
    (ii) word is a new word form corresponding to an existing lemma
    (iii) word belongs to a hitherto unknown lemma
    In case (ii) or (iii): add lemma and/or normalized word form to the database
    Step 4. Rerun pattern inference (CL Tool S1) (we now have new example data)
    Step 5. back to 1.
}
```

Figure 2: corpus-based lexicon building

Example:
Text = ‘*Terwyl wy hier van woningen spreken, moet ik zeggen dat my in deze Stadt vremt voorquam het maexel van huizen, die geheel voltoit hier op de markt te koop gebragt worden.*’
Initial Lexicon = {*terwijl, wij, hier, woning: {woning, woningen), van, spreken, moeten, zeggen, dat, mij, in, deze stad, ik, vreemd, het, huis, huizen, die, voorkwam, geheel, voltooid, hier, op, de, markt, te, koop, gebracht, worden*}
Initial Patterns = { *y/ij, qu/kw, ae/aa, g/ch, ch/g*}
After step 2:
W_1 = { *hier, van, woningen, …* }
W_2 = { *terwyl, wy, my, voorquam, gebragt* }
W_3 = { *stadt, vremt, maexel, voltoit* }
In step 3:
Add to lexicon: new lemma maaksel,
Add for pattern inference: examples (maaksel, maexel); examples (stadt, stad), (vremt, vreemd), (voltoit, voltooid)
After step 4 (Rerun pattern inference): new patterns { x/ks, dt$/d$, oi/ooi, t$/d$}
Please note that this example is not entirely realistic: pattern inference only works for a large example set

Figure 3: example for the workflow in figure 2

Bootstrapping of lexicon content from parallel texts in modern and original spelling

Another opportunity for quick creation of lexicon content and example material for models of historical orthography is the exploitation of parallel texts in historical and modern language, when one has access to both a respelled edition and an edition in original spelling of a certain work. In order to use this material, one has to obtain a word-to-word alignment between the two versions.

In some cases, this is comparatively easy, because a line-by-line alignment is facilitated by the data (cf. for instance the two versions of Cervantes' work at http://users.ipfw.edu/jehle/wcdq.htm). We were able to extract 10.000 historical variants in this way, and computed orthographical variant patterns from them, obtaining typical patterns like v→u, s→ss, á→a, b→u, í→i, ía→ia, c→z, ó→o, z→ç, é→e, j→x, v→b.

In other cases, one has to use alignment software. For Dutch, we tested this scenario on of the 1637 and 1888 "Statenvertaling" versions of the bible. For German, there are 1554 and 1912 versions of the Luther bible. We used GIZA++[19] to align the versions

As an example of the variants that can be added to a lexicon in this way (historical variants underlined):

Am/AM Anfang/anfang schuf/schuf Gott/Gott *Himmel/Himel und/vnd Erde/Erden*

Und/Vnd die/die Erde/Erde war/war wüst/wüst *und/vnd* leer/leer

und/vnd es/es war/war finster/finster *auf/auff* der/der *Tiefe/Tieffe und/Vnd* der/der Geist/Geist Gottes/Gottes *schwebte/schwebet auf/auff* dem/dem Wasser/Wasser

Special use of the IMPACT Lexicon Tool (VI 6) in case of a limited amount of corpus material

With the IMPACT Lexicon Tool manual correction of enriched corpus material is done. In the tool, analyses on type level can be verified and assigned to attestations in the corpus. Disambiguation on token level is also possible. Suggested analyses in the tool are

	Validated	not validated
Attested	**bold** attestations will show in the lower part of the tool	not bold attestations will show in the lower part
Not attested	**bold** no attestations in the lower part	not bold no attestations in the lower part

When starting from scratch, word forms will typically be in the lower right corner where they are neither attested nor validated. Ideally they ought to end up in the upper left corner, where they are both attested and (hence) validated.

[19] Franz Josef Och, Hermann Ney. "A Systematic Comparison of Various Statistical Alignment Models", *Computational Linguistics*, volume 29, number 1, pp. 19-51 March 2003.

However, it *is* possible for an analysis to be validated for a type of the corpus even though there are no attestations to prove it. It could be that you want this in the case where an analysis is completely obvious but your corpus is e.g. somewhat small and it coincidentally has no sentences in which the word occurs in this sense. It is always easy to see when an analysis is verified but not attested as it will not be listed in the middle column (displaying the analyses of a type in the corpus) but it will be displayed in bold in the right one.

	Validated	not validated
attested	**bold** in the right column shown in middle column	not bold in the right column shown in the middle column
not attested	**bold** in the right column not shown in middle column	not bold in the right column not shown in middle column

3. Building a historical morphological OCR lexicon

Morphological analysis is the task of analysing complex words (compounds, derivations) in terms of their composing parts. Morphological analysis can improve recall in IR, because it can help us to produce related search terms, for instance *morphology* → *morphological*. In OCR, morphological analysis can be used to accept words not explicitly listed in the lexicon as probably valid. This is especially relevant for languages like Dutch and German, in which compounds are written as one word (rather than being written separately as they usually are in English) and constitute an important part of the vocabulary which is difficult to cover explicitly in a lexicon. In such cases morphological analysis can come to aid, as it can recognize the parts of the compound, which in turn *can* be listed in the lexicon. In what follows we will focus on morphological analysis for the OCR engine.

We use finite state technology to implement the morphological lexicon, because it is easy to implement, it calls for data that people who will use it are likely to have readily available, and also, very importantly especially for use with the OCR engine, because it runs very fast. Other approaches that could be taken to find out if a word looks like a possible word for a certain language include trigram models, machine learning/classifier based approaches, etc. These approaches will respond to word validity queries with a confidence score rather than with the simple *yes/no* answers returned by our (unweighted) finite state approach. In a setup where linguistic processing can be integrated more tightly with the OCR, and the confidence of character recognition can be balanced against linguistic confidence, this could be an advantage. However, the way in which we integrate lexica in OCR in IMPACT, relying on the FineReader Engine external dictionary interface, does not easily allow us to do this.

The finite state approach to morphological analysis is to implement an efficient "*transducer*", which can be used to "translate" a representation of the analysis of a word into its surface form or vice versa. Thus, it can be used for both word generation and analysis. We make use of the Xerox Finite State Transducer library[20]. It is beyond the scope of this cookbook to go into the exact details of this software. Please refer to the excellent documentation that comes with it.

[20] *Finite State Morphology*, Kenneth R. Beesley and Lauri Karttunen, CSLI Publications, 2003.
http://www.fsmbook.com

The XFST script in figure 3 illustrates how we can use the XFST software to build the transducer from a lexicon of morphemes (morphological building blocks),a set of morphological word formation rules, and a set of spelling variation rules, and apply the network to prune a set of recognition candidates to the set of linguistically valid ones. The steps are as follows:

1. Define a morpheme lexicon by listing the entries for each morpheme category
2. Define compounding rules. The network *validModernWords* now accepts, besides the listed simplexes, combinations like *fietsauto, autobel* and *fietsbel.*
3. Define spelling variation rules as a transducer *spellingRules* mapping "modern" words to their hypothetical historical forms
4. The composition modernToHistorical := validWords ∘ spellingRules now is a partially defined mapping with domain *validModernWords* . The range of this mapping is what we will accept as valid historical words.
5. Obtain the finite state network *validHistoricalWords* accepting the range of *modernToHistorical* by taking, in XFST terms, the "*lower part*" of *modernToHistorical.*
6. Define a network accepting a set of recognition candidates *fuzzyWord* := {*fietsbel, fietsbcl*} by an XFST regular expression
7. The set of valid recognition candidates is simply the intersection of *validHistoricalWords* and *fuzzyWord*

```
! Step 1: define Morpheme lexicon
define NOU [ {fiets} | {auto} | {bel} ] ;
define VRB [ {walk} ] ;

! Step 2: define morphological rules
define validModernWords [
NOU |
NOU NOU |
VRB
] ;

! Step 3: spelling rules
define spellingRules [
{s} (->) [ s c h ] || _,,          ! aa -> ae
o (->) [ o o ] || _ .#.          ! o -> oo but only at the end of a morpheme
] ;

! Compose the two networks
define modernToHistorical validModernWords .o. spellingRules ;
! Only take the lower side
define validHistoricalWords modernToHistorical.l ;

! An ABBYY 'fuzzy word' will be compiled to an xfst regular expression by the
software
define fuzzyWord [ f i e t s b [ c | e ] l ] ;
! Calculate the intersection of the two
define prunedFuzzyWord validHistoricalWords & fuzzyWord ;
! Print the outcome (it should succeed in this case: 'fietsbel')
print words prunedFuzzyWord ;
```

Figure 3: XFST script illustrating the use of finite state morphology to prune a set of recognition candidates

To apply this approach in combination "*in real life*" in combination with the OCR engine, the implementation is split in two parts: a perl script that invokes XFST to compile the data to a finite state network, and a small C header library that performs the task of pruning a set of recognition candidates to the ones accepted by the network.

1: xfstFiles\rules2xfst.pl (implementing steps 1-5 above)

A Perl script that builds a *validHistoricalWords* network from a morpheme lexicon, a set of morphological rules and spelling variation rules. It has to be run only once, unless something is changed to the input data.
The script needs three files: morpheme lexicon, morphological rules and spelling variation rules.

1. *Morpheme lexicon*
   A tab separated file listing all morphemes: category<TAB>morpheme<TAB>frequency. NOTE that the last column is ignored at present.
2. *Morphological rules:* A tab-separated file listing the ways morphemes can be combined to make up legal words, for instance
   ```
   NOU -> NOU NOU<TAB>freq
   ...
   ```
   NOTE that again the frequency is ignored at present.
3. *Spelling variation rules*
   These should be xfst rewrite rules.
   Please find an example in: xfstFiles\multigrams.pruned-xfst.txt

xfst/xfsm_api/src/fsnMorph/fsnMorph.h (steps 6-7)
A C header file. It implements the function checkFuzzyWord().
This is the part that is to be integrated with the ABBYY FineReader OCR engine. It implements one function in particular:

```
void checkFuzzyWord(wchar_t ** wcppFuzzyWord, int iWordLength,
                FST_CNTXTptr fst_cntxt, NETptr netpMorphAndSpell);
```

This function takes an ABBYY fuzzy word as input, plus a pointer to an fst context that the fst software needs and a pointer to the spelling variation rules.

Please refer to xfst/xfsm_api/src/fsnMorph/testFsnMorph.c for example code.

Note: A DLL has been developed that enables the use of XFST morphology with the FineReader engine SDK. External dictionary interface. Documentation for this will follow at a later stage.

V Lexicon building and deployment tools (D-EE 2.4 and D-EE 2.5): Technical documentation

1. ImpacTok - Tokenizer

1.1 Partner

INL

1.2 Deliverable

part of D-EE2.5

1.3 Background

The tokenizer described in this document is used to preprocess the documents that form the corpus used to build the lexicon.

This tokenizer is based on ILKTOK, part of the 'Tadpole' language processing suite (http://ilk.uvt.nl/software/). A rewrite of the code was necessary in order to produce the output required for the database used for the IMPACT Lexicon and to introduce a more modular approach.

1.4 Requirements

ImpacTok requires the Perl program with some additional libraries (‘HTML::Entities’, ‘Getopt::Std’), which can be obtained from CPAN or comparable repositories.

1.5 The ImpacTok package

The software consists of a perl script (‘impactok.pl’) and a number of data files. The data files should all reside in the same folder.

The data files contain special strings that should be treated differently. The names of the data files consist of two parts: <type>.<lang>.

The following types are considered:

- abbreviations (abbr), a list of abbreviations (without the period) that are frequent in a certain language. One entry per line.
- apostrophes (apostrof), a list of words that require an apostrophe.

The second part of the name of data files indicates the language for which it is used. Eg. 'eng' is English, 'deu' is German, etc.

1.6 Using ImpacTok

Run the script with 'perl ImpacTok.pl -dflot'

The following arguments are used:

- -l Language
- -d Path to the datafiles
- -f Input file for tokenisation. This argument is overridden by -t

- -o Output file of tokenisation. This argument is overridden by -t
- -t use this option to run a batch job. The argument specifies a file in which input - and output files are paired.

1.7 Output

Every token with its additional fields is printed on a separate line. Additional fields are the onset and offset of the token in the input file and the complete fragment of the document that contains the token. All fragments together compose the complete original document. The fields are separated by a TAB.

1.8 Adapting the script

It is possible to extend the number of special data types beyond those presently included (abbreviation, apostrophe). The function 'initialize' calls for every file a special function that parses the data file and builds a hash.

The function 'tokfile' does the tokenisation of a file. It first applies a basic tokenisation using spaces and line endings. Then several rounds of adjustments are applied: handling hyphenation, punctuation a the start of the token and punctuation at the end of the token. It is possible to add other adjustments as an extra round.

The function 'tokfile' also contains the output routine. The output format can be altered here.

1.9 Licensing

The licensing follows the consortium agreement.

The tool will be made available to the research community according to the regulations of the Dutch HLT agency, which means that it is freely available for non-commercial use.

2.0 CitAttest Attesting Word Forms in Dictionary Citations

2.1 Partner

INL

2.2 Deliverable

part of D-EE2.5

2.3 Background

In this document we describe the tool to attest word forms from a Dutch dictionary. The purpose of this process is to build a word form lexicon that can be used within the IMPACT Project. Since this tool has been developed for a specific task on specific data, it will need some adaptation to deploy it in other situations. See for further information V.1.

2.4 Requirements

CitAttest requires the Perl program with some additional libraries (‘HTML::Entities’, ‘Getopt::Std’), which can be obtained from CPAN or comparable repositories.

2.5 The CitAttest package

The script main.pl performs the main cycle. It reads the dictionary files and processes the articles one by one. The specific functions for handling the idiosyncratic WNT-structures have been collected in the package 'wnt_article.pm'. This script extracts the citations from the article and binds them to the proper head words. The script 'Main.pl' also calls the package 'InitialVariants.pm' which provides inflected variants for the head words. The operations on the database are placed in the package 'attestationBase.pm'.

2.6 Using CitAttest

Run the script with 'perl main.pl <file name>'

2.7 Matching algoritm

The main procedure for matching is located in the package 'HeadwordMatching.pm'. The basic matching operation is quite simple. We have a list of keywords and their morphological variations at the one hand, and a citation on the other hand.

The first step is to tokenize the citation. Next we compare each element from the token list with each element in the keyword list. Since the keyword list is a list of lists (containing lists of variants) we need a three layered loop to make all comparisons. As was already mentioned, in comparing we calculate the edit distance (Levenstein) between the two strings. Only the candidates that score below a certain threshold are marked.

The algorithm gets more complicated when we want to detect multiple targets in the quote, like separable verbs. In dutch some verbs are prefixed with another word like 'op' in 'opspelen' (play up). In some conjugations the prefix becomes detached from the verb. In those cases we want to mark both parts as belonging to the lemma.

Information on separable forms is available in the lexicon that we use in producing morphological variations of the lemma, as well as in the header information from the dictionary entries.

The algorithm is extended as follows: The basic matching operation is performed with the verbal part of the separated form. If this verb delivers the best candidate, we also perform a basic matching operation with the prefix.

2.8 Adapting the script

This tool has been developed for a special task performed on specific data. In order to adapt it on other data, large portions will require extensive rewrite. The most generic part is the matching algorithm which is located in HeadWordMatching.pm'.

2.9 Licensing

The licensing follows the consortium agreement.
The tool will be made available to the research community according to the regulations of the Dutch HLT agency (TST-Centrale, www.inl.nl), which means that it is freely available for non-commercial use.

3.0 Dictionary Attestation Tool

3.1 Partner

INL

3.2 Deliverable

part of D-EE2.5

3.3 Background

This tool is meant to be used for manual evaluation and correction of large quantities of automatically matched occurrences of a headword in the quotations of the particular article in a comprehensive dictionary (see V.1).

For the processing of attestations we defined the following general design principles for an annotation tool that enables the verification of automatically generated attestations of word forms.

1. Multiple concurrent sessions (by different staff members) should be possible. In order to meet deadlines it is often necessary to have more than one staff member working on evaluating the data. The tool should allow several users to access the database and deliver their input.
2. The verification tool should be in the form of a web application that can be run from any computer in the local network.
3. Input actions, especially the frequent ones, should be from the keyboard, since this allows for faster responses than clicking the mouse on screen buttons.
4. Information should be presented such that quick evaluation is possible.

3.4 Features and system requirements

The Attestation Tool is based on a LAMP[21] architecture. Users need a web browser. We tested the user interface on: Firefox 3.0.5, and some earlier versions, on Linux, Mac OS X 10.4 and Windows XP; Safari 3.1.2 on Mac OS 10.4; Internet Explorer on Windows XP. The web server needs to have MySQL (The tool was tested on MySQL 5.0.27) and PHP installed. The server side was tested with PHP version 5.2.0, Apache 1.3 on Mac OS X 10.4, and Apache 2.0 with PHP 5.1.6 on Red Hat Enterprise Linux 5.

The interface consists of just one page: attestationTool.php. It is a so-called rich Internet application which means that it uses AJAX to communicate with the database server and display the results.

[21] The acronym LAMP refers to a solution stack of software, usually free and open source software, used to run dynamic Web sites or servers. The original expansion is as follows:

- Linux, referring to the operating system;
- Apache, the Web server;
- MySQL, the database management system (or database server);
- PHP or others, i.e., Perl, Python, the programming languages.

http://en.wikipedia.org/wiki/LAMP_(software_bundle)

The tool has been built for speed. When the automatic matching has worked out reasonably well users can very easily scan through the results, correct some mishaps and hit the spacebar to get the next lemma.

3.5 Configuration

The tool for manual annotating the attestations requires some minor adaptations during installation. Mainly the addresses for services and the names of databases, database users and passwords have to be adapted in the PHP scripts. Also, some settings must be added to the Apache configuration file, usually called “httpd.conf”.

3.6 Attestation Tool database

Table lemmata

Field	Type	Description
id	number	Internal identifier. Primary key.
lemma	string	Head word corresponding to the word form.
partOfSpeech	string	Part of speech of the lemma.
initialVariants	string	Set of variants. This field is used by the scripts in automatic pre-processing.
revisionDate	date	Date of revision.
revisorId	string	Identifier of staff member performing the revision.
externalLemmaId	string	The number of the article in the dictionary (in our case, the WNT).
marked	boolean	Indicates whether or not the lemma is marked.
hide	boolean	Indicates whether or not the lemma is shown in the tool.
comment	string	Comment field.

Table quotations

Field	Type	Description
id	number	Internal identifier. Primary key.
lemmaId	number	Identifier of the lemma this quotation belongs to.
quotation	string	The actual quotation.
tokenizedQuotation	string	The quotation split in tokens.
quotationSectionId	string	This field is for internal use of the scripts.
dateFrom	int	Year indicating the first occurrence of the word quoted.
dateTo	int	Year indicating the last occurrence of the word quoted.
specialAttention	bool	Can be set when the quotation is somehow out of the ordinary.
unfortunate	bool	Can be set when the headword doesn't really occur in the quotation.
updated	bool	Can be set if, after an update of the dictionary data, there is a discrepancy between the quotation in the data and the quotation in this record.

Table attestations

Field	Type	Description
quotationId	number	Identifier of the quotation this attestation belongs to.
onset	number	Character position of the start of the word attested.
offset	number	Character position of the end of the word attested.
reliability	float	Indicates how certain the match is (the more different the higher this number).
wordForm	string	The word attested as occurring in the quotation.
typeId	number	Only relevant when doing multiple typed attestations (e.g. named entity attestations where there is distinction between NE_LOC, NE_ORG and NE_PER).
error	boolean	Indicates whether or not a word was marked as erroneous.
dubious	boolean	Indicates whether or not a word was marked as dubious.
elliptical	boolean	Indicates whether or not a word was marked as being elliptical. This can be used to mark a word like *North* in *North and South America*.

Table groupAttestations

Field	Type	Description
id	number	Identifier of the group
attestationId	number	Identifier of an attestation.
pos	number	The position number the attestation has in the quotation.

Table types

This table should only exist when typed attestations are needed. The interface of the tool will automatically detect whether or not this table exists. If it does the appropriate interface functions are loaded. For ‘simple’ attestation (like marking a headword in a citation) just leave ou the entire table.

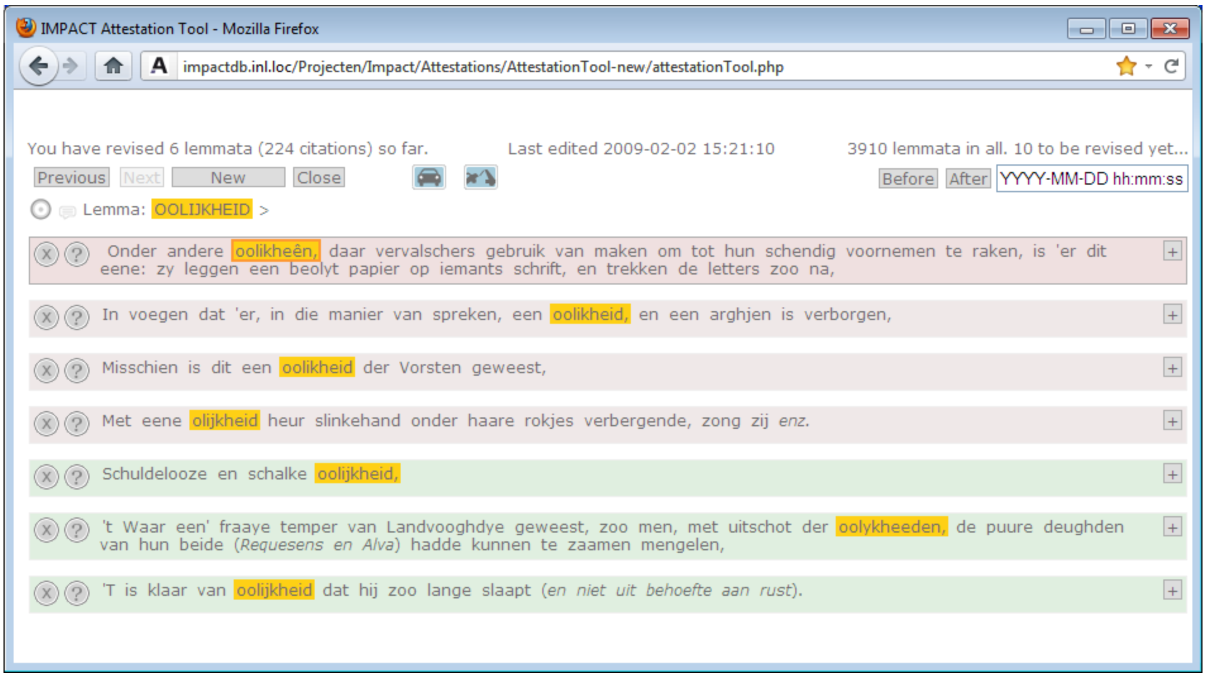
Field	Type	Description
id	number	Identifier of the type
name	string	The name of the type/
color	string	The colour attestations of this type should have in the tool (as HTML hexadecimal colour codings (e.g. ‘#A1BFF7’).

Table revisors

Field	Type	Description
id	number	Identifier of the revisor
name	string	Name of the revisor

3.7 User interface

The attestation tool was designed to enable multiple concurrent users to view the data in the database and to make changes to it. In the screenshot below you see an example page containing seven quotations for the Dutch word 'oolijkheid'.


IMPACT Attestation Tool - Mozilla Firefox
impactdb.inl.loc/Projecten/Impact/Attestations/AttestationTool-new/attestationTool.php
Previous
Next
New
Close
Before
After
YYYY-MM-DD hh:mm:ss

You have revised 6 lemmata (224 citations) so far. Last edited 2009-02-02 15:21:10 3910 lemmata in all. 10 to be revised yet...

Lemma: OOLIJKHEID >

Onder andere oolikheên, daar vervalschers gebruik van maken om tot hun schendig voornemen te raken, is 'er dit eene: zy leggen een beolyt papier op iemants schrift, en trekken de letters zoo na,

In voegen dat 'er, in die manier van spreken, een oolikheid, en een arghjen is verborgen,

Misschien is dit een oolikheid der Vorsten geweest,

Met eene olijkheid heur slinkehand onder haare rokjes verbergende, zong zij *enz.*

Schuldelooze en schalke oolijkheid,

't Waar een' fraaye temper van Landvooghdye geweest, zoo men, met uitschot der oolykheeden, de puure deughden van hun beide (*Requesens en Alva*) hadde kunnen te zaamen mengelen,

'T is klaar van oolijkheid dat hij zoo lange slaapt (*en niet uit behoefte aan rust*).

The tool lists quotations per lemma. Quotations are listed by uncertainty. The most uncertain ones (containing words least similar to the headword) appear at the top and are marked red(dish). Literal matches are at the bottom, marked green.

By using the arrow keys or the mouse, users can select or deselect words or move a selection.

The 'X' button can be used to mark quotations requiring special attention (e.g. because they were extracted in the wrong way). The '?' button can be clicked to mark quotations that are 'unfortunate' (e.g. the headword doesn't appear in the quote as such but only in a compound).

The target button left of the lemma head word can be used to mark an analysis. The text balloon icon between the mark button and the lemma headword can be used to add a comment to a lemma.

3.7.1 Auto attestation

When a very frequent variant has been missed in automatic matching, auto attestation can come in handy. A user can select a word and, by hitting the auto attestation button, all occurrences of this word form can be highlighted.

3.7.2 Auto de-attestation

When multiple occurrences of the same word form should be de-attested the auto de-attestation button can be used. By clicking on it the selected word form and all identical occurrences will be de-attested.

3.7.3 Keyboard shortcuts

To enhance the usability the interface can be used with the mouse, with the keyboard or both.

Key	Action
F2 or d	Previous revised lemma
F4 or f	Next revised lemma
F8	Add a comment
F9 or x	Toggle quotation as wrongly/correctly parsed
Spacebar	Save current lemma, and display a new unrevised one
a	Auto attestation
m	Mark the lemma
u	Toggle quotation as fortunate/unfortunate
z	Auto de-attestation
INSERT	Insert a new attestation
DELETE	Delete currently selected attestation
CTRL	Walk through attestations of the selected quote

3.8 Licensing

The licensing follows the consortium agreement.

The tool will be made available to the research community according to the regulations of the Dutch HLT agency (TST-Centrale, www.inl.nl), which means that it is freely available for non-commercial use.

4. Impact EE2 Spelling Variation Tool

4.1 Partner

INL

4.2 Deliverable

part of D-EE2.5

4.3 Background

The IMPACT Lexicon Tool is a tool for dealing with historical spelling variation. It provides tools to estimate a model of spelling variation from example data, and to match a historical word, or a list of historical words, to a list of ‘modern’ words (or historical words in normalized, modern-like spelling).

4.4 Installation and system requirements

The tool is a java command line application developed for SUN java 1.6. Other versions have not been tested. It relies on the following libraries:

Library	Version used	File
JgraphT (Java graph library that provides mathematical graph-theory objects and algorithms)	0.8.1	Lib/jgrapht-jdk1.6.jar
The Apache Commons CLI library	1.2	Lib/commons-cli-1.2.jar
Mysql connector for java	3.0.17	Lib/mysql-connector-java-3.0.17-ga-bin.jar
Weka 3: Data Mining Software in Java	3.6.2	Lib/weka.jar

Accordingly, the corresponding jar files and the file “impact_spellingvariation_1.0.jar” should be on your CLASSPATH to be able to run the tool.

The tool can be invoked by calling one of the main executable classes, currently either

```
        java spellingvariation.MultigramTransducer
```

for the inference of a set of weighted substitution patterns or

```
        java spellingvariation.MemorylessMatcher
```

to match historical words to a list of “modern” words.

A minimum amount of 2G internal memory is needed to run the tools. For large datasets, more may be needed.

4.5 Obtaining a model from example data

The tool takes an input file consisting of example pairs and produces an output of two files: one with the patterns found during training and one containing the optimal alignments for the example pairs.

The input file is a tab-separated file with three columns

First column	Some arbitrary ID (for instance the ID of the lemma or a line number). This fields is not used by the tool, but it is included for the convenience of the user
Second column	One or several “modern” word forms, separated by blanks
Third column	Historical word form

The output consist of a file with weighted patterns with six columns

First column	The pattern, in the form modern→historical[22]
Second column	Joint relative frequency (based on a weighted avarage of different possible alignments)
Third column	P(historical|modern) Relative frequency of the pattern conditioned on the modern
Fourth column	P(modern|historical) Relative frequency of the pattern conditioned on thehistorical part
Fifth column	Joint relative frequency of the pattern, based on the best alignments after parameter estimation
Sixth column	Number of uses of the pattern in the optimal alignment of the training data

Example: the toy Czech input file (randomly selected from data obtained from aligned modern and original versions of Karel Hynek Mácha's poem *Máj*, http://www.lupomesky.cz/maj/maj-puv.html)

```
1  vřelé  wřelé
2  nejvejš       neyweyš
3  dvacátý       dwacatý
4  uspávati      uspáwati
5  nynější       nyněgšj
6  jaké   gaké
7  nesmírné      nesmjrné
8  neví   newj
9  v      w
10 zván   zwán
11 přijdem       přigdem
12 jeden  geden
13 růžojasné     růžogasné
14 pohrává       pohráwá
15 větýrek       wětýrek
16 svadlý swadlý
17 jezero gezero
18 leží   ležj
19 roucha raucha
20 pravý  prawý
```

Running the command:

```
java spellingvariation.MultigramTransducer
-f Properties/maj.properties
--trainFile=Data/Czech/Maj/mini.txt
--patternOutput=Data/Czech/Output/patterns
--alignmentOutput=Data/Czech/Output/alignments
```

Yields the following two outputs:

[22] The arrow reverses time, which may appear illogical. For what it is worth, the reason why we write it this way is we tend to think (technologically, not linguistically) of historical spelling as noise introduced by a channel model in the "underlying" modern spelling.

1: Patterns:

```
v→w      0.06452      0.909 0.9090909090909091      0.062899    10
j→g      0.03870611679708703     0.75  1.0   0.03773584905660377     6
í→j      0.012904876847530205    0.5   0.5   0.012578616352201259    2
í$→j$    0.01289920101719448     1.0   1.0   0.012578616352201259    2
ej→ey    0.01290203258814683     1.0   1.0   0.012578616352201259    2
v$→w$    0.00643672962632532     1.0   1.0   0.006289308176100629    1
ou→au    0.006451019466128365    1.0   1.0   0.006289308176100629    1
á→a      0.0064510    0.2   0.1111111111111111      0.0062893081       1
```

2: Alignments:

```
1  ^vřelé$      ^wřelé$      [^][v→w][ř][e][l][é$]
2  ^nejvejš$    ^neyweyš$    [^][n][ej→ey][v→w][ej→ey][š$]
3  ^dvacátý$    ^dwacatý$    [^][d][v→w][a][c][á→a][t][ý$]
4  ^uspávati$   ^uspáwati$   [^][u][s][p][á][v→w][a][t][i$]
5  ^nynější$    ^nyněgšj$    [^][n][y][n][ě][j→g][š][í$→j$]
6  ^jaké$       ^gaké$       [^][j→g][a][k][é$]
7  ^nesmírné$   ^nesmjrné$   [^][n][e][s][m][í→j][r][n][é$]
8  ^neví$       ^newj$       [^][n][e][v→w][í$→j$]
9  ^v$   ^w$    [^][v$→w$]
10 ^zván$       ^zwán$       [^][z][v→w][á][n$]
11 ^přijdem$    ^přigdem$    [^][p][ř][i][j→g][d][e][m$]
12 ^jeden$      ^geden$      [^][j→g][e][d][e][n$]
13 ^růžojasné$ ^růžogasné$ [^][r][ů][ž][o][j→g][a][s][n][é$]
14 ^pohrává$    ^pohráwá$    [^][p][o][h][r][á][v→w][á$]
15 ^větýrek$    ^wětýrek$    [^][v→w][ě][t][ý][r][e][k$]
16 ^svadlý$     ^swadlý$     [^][s][v→w][a][d][l][ý$]
17 ^jezero$     ^gezero$     [^][j→g][e][z][e][r][o$]
18 ^leží$       ^ležj$       [^][l][e][ž][í$→j$]
19 ^roucha$     ^raucha$     [^][r][ou→au][ch][a$]
20 ^pravý$      ^prawý$      [^][p][r][a][v→w][ý$]
```

4.6 Applying the model to data: matching

The simple models obtained in the previous section are unable to transcribe a historical word to modern spelling. However, they can be used to match historical words to a list of known modern words ant to use the pattern weights (i.c the conditional probability p(modern|historical) to select the most plausible match.

The matching tool takes three input files (refer to the next section for the way to pass these parameters)

1) A Pattern file as produced by the procedure in the previous section (option *patternInput*)
2) A reference “modern” word list to match to (option *lexicon),* containing one word per line
3) The input (option *testFile*), again a word list containing one word per line, or (when evaluating performance)

Furthermore, we distinguish two running options:

1) of testing performance on labeled example material of the form describe in the previous section (option command=test)
2) running the matching procedure on unseen materiaal (option command=

For instance, to run the patterns obtained for the example in the previous section on the larger labeled example set “pairs.txt”, we execute

```
Java spellingvariation.MemorylessMatcher
 --command=test
 --testFile=Data/Czech/Maj/pairs.txt
 --patternInput=Data/Czech/Output/patterns
 --lexicon=Data/Czech/Maj/modern.list
 --addWordBoundaries=true
```

(The result in this case is 909 out of 930 correct matches, which clearly indicates we selected an easy example)

4.7 Simultanuous Parameter Estimation and Picking of Matches

The tool provides an option to simultaneously infer pattern weights and choose matches from a limited list of possibilities. In this case, the second column in the example file may contain more than one word.

A possible application is to list all modern words within a certain Levenshtein distance to the hostrical word, and bootstrap pattern inference from there.

Though this option has been found useful in matching word lists from 19th century and 17th century Dutch bibles, we found no further applications of this scenario.

4.8 The Command Line Syntax

The tools’ behaviour is determined by a set of options which can be either specified on the command line or in a properties file. The command line options are:

```
 -a,--alignmentOutput <arg>           output file for alignments
 -b,--addWordBoundaries <arg>         add word boundaries ^ and  $ to
                                      strings before matching and inference
 -c,--command <arg>                   action to perform: train | test |
                                      run
 -C,--minimumConfidence <arg>         minimum conditional probability lhs
                                      | rhs
 -D,--allowDeletions <arg>            allow empty right hand side during
                                      matching or not (true|false)
 -d,--outputDirectory <arg>           default output directory
 -f,--properties <arg>                property file with job options
                                      use long option names in the
                                      properties file
 -h,--help                            print this message and exit
 -I,--allowInsertions <arg>           allow empty left hand side during
                                      matching or not
 -i,--trainFile <arg>                 input file for pattern training
 -J,--minimumJointProbability <arg>   minimum joint probability for rule
```

```
                                            to be included in pattern matching
 -l,--lexicon <arg>                         lexicon (word list file to match to)
 -M,--multigramLength <arg>                 maximum multigram length
 -o,--patternOutput <arg>                   output file for patterns
 -P,--maximumPenalty <arg>                  maximum matching penalty
 -p,--patternInput <arg>                    input file for patterns used by the
                                            matcher
 -s,--maximumSuggestions <arg>              maximum number of matching
                                            suggestions
 -t,--testFile <arg>                        input file for testing
 -X, --forbidInsertsAndDeletes <arg>       (if true) Do not save inserts and
                                            deletes in pattern output
 -r, --pruner                               Java class used for determining which
                                            multigrams are acceptable patterns
```

The option “-f <configuration file>” allows reading the options from a standard properties file using the long option names as keys, f.i.

```
trainFile=my.input.file
patternOutput=my.patterns
alignmentOutput=my.alignments
```

4.9 Example dataset

The folder “Data” contains some more realistic example material.

4.9.1: Data/Slovene

Contains the following files:

gooLex.pairs: examples of modern and historical words extracted from the Slovene google corpus
gooLex.train: training set randomly chosen from the above
gooLex.test: corresponding test set (= gooLex.pairs \ gooLex.train)
gooLex.mforms.txt: modern Slovene word forms from the Slovene google corpus
multext.mforms.txt: modern Slovene word forms from the multext lexicon
slovene.properties: property file for Spelling variation tool

Running

Unzip the zip file.
Before running the tool, be sure to set the class path, for instance by running

```
source setClassPath.sh
```

in the root directory of the extracted zip file.

To train, run

```
java spellingvariation.MultigramTransducer –f slovene.properties
```

The pattern output will be gooLex.multigrams.out

To test using gooLex.test, run

```
Java spellingvariation.MemorylessMatcher –f Slovene.properties
```

5. IMPACT Tools for Lemmatization and Reverse Lemmatization

5.1 Partner

INL

5.2 Deliverable

Part of D-EE2.5

5.3 Background

EE2 provides tools for 1) Reducing historical wordforms one or several possible modern lemma’s (lemmatization) and 2) Expanding lemma lists with part of speech information to possible (“hypothetical”) full form. The purpose of lemmatization in IMPACT is improved retrieval in historical documents. The reverse lemmatization is used to create hypthetical lexicon content to be used mainly in lexicon building, but possibly also in OCR and information retrieval.

5.4 Installation and system requirements

The tool is a java command line application developed for SUN java 1.6. Other versions have not been tested. It relies on the following libraries:

Library	Version used	File
JgraphT (Java graph library that provides mathematical graph-theory objects and algorithms.)	0.8.1	Lib/jgrapht-jdk1.6.jar
The Apache Commons CLI library	1.2	Lib/commons-cli-1.2.jar
Mysql connector for java	3.0.17	Lib/mysql-connector-java-3.0.17-ga-bin.jar
Weka 3: Data Mining Software in Java	3.6.2	Lib/weka.jar

Accordingly, the corresponding jar files and the file “impact_lemmatization_1.0.jar” should be on your CLASSPATH to be able to run the tool.

The tool can be invoked by calling one of the main executable classes, currently either (for reverse lemmatization)

```
      java lemmatizer.SuffixGuesser or
      java lemmatizer.PrefixSuffixGuesser
```
[23]

or

```
      java spellingvariation.Lemmatizer
```

to match historical wordforms to modern lemma's using a historical lexicon, a modern lexicon and a set of weighted patterns describing the relation between modern and historicla spelling.

A minimum amount of 2G internal memory is needed to run the tools. For large datasets, more may be needed.

5.5 Reverse lemmatization (expansion of lemma list to hypothetical full form lexicon)

The tool takes an example (modern) full form lexicon as training data, reads in a list of lemma's with part of speech information and produces hypothetical full form information for the lemma's in the list. This full form information is certainly noy always correct, but it can be used in both OCR and lemmatization.

The example lexicon file (specified by option *trainingData)* is a tab-separated file with 4 columns

First column	Word form
Second column	Lemma form
Third column	Part of speech tag with features, describing the word form
Fourth column	Main part of speech tag for the lemma

The lemma list to be expanded has the format

First column	The lemma form
Second column	The main part of speech tag for the lemma

The output produced is in the same format as the example lexicon.

```
java -mx1200m lemmatizer.PrefixSuffixGuesser --
trainFile=Data/Dutch/JVKlex.tab --testFile=Data/Dutch/some_wnt_lemma's
```

To test accuracy of the reverse lemmatizer, run for instance

```
java -mx1200m lemmatizer.PrefixSuffixGuesser
--referenceLexicon=Data/Dutch/JVKlex.tab

--command=test
```

or

```
java -mx1200m lemmatizer.SuffixGuesser
--referenceLexicon=Data/Polish/morfologik.verbs.type_lemma_pos.tab
```
[24]

[23] The difference between these two is that the SuffixGuesser only looks at word suffixes to choose an inflection pattern for a word. This is acceptable in many situations, bu not when f.i. Dutchpast particples with *ge-* have to ve produced.

```
--command=test
```

5.6 Lemmatization

The lemmatization process relies on

1) A “witnessed” historical lexicon from which possible lemma’s are simply obtained by lookup
2) A reliable modern full form lexicon, possibly augmented by the expansion of a historical lemma list in modern spelling to hypothetical full form obtained by reverse lemmatization
3) A compiled double array trie containing the word forms in the modern lexicon
4) A set of weighted patterns used to match historical words which were not found in 1) or 2) to wordforms in 2)

The relevant syntax is

```
java lemmatizer.Lemmatizer\
   --modernLexicon=<modern lexicon file>\
   --lexiconTrie=<compiled double array trie with modern word forms>\
   --historicalLexicon=<historical lexicon file>\
   --patternInput=<pattern file>\
   --lemmatizerInput=<input file, one word per line>
```

The result is written to standard output.

5.7 Preparing the lexica for use

The tool uses the graph database neo4j[25] to store the the lexica. So we have to convert them to that format first. A simple command line tool does that job:

```
java lexicon.PrepareLexicon\
   --targetDirectory <target directory for neo4j lexicon data>\
   --modernLexicon <modern lexicon as text file or mysql database>\
   --historicalLexicon <historical lexicon as text file or mysql database>\
   --databaseHost <database server hostname>
```

The tab-separated plain text input format for a lexicon file has the fields

wordform<tab>*lemma*<tab>*part of speech tag with word form features*<tab>*part of speech tag for lemma*

[24] This dataset containing polish verbs was extracted from the “morfologik” Polish lexicon, http://mofologik.blogspot.com

[25] http://neo4j.org/

By default, the tool will assume plain text input for the lexica. If the value of modernLexicon or historicalLexicon is of the form “database:<database name>”, the tool will compile the lexicon from the IMPACT lexicon database specified. Currently, to avoid problems with wordform groups, the lexicon database needs an extra table simple_analyzed_wordforms which is used for this extraction. The extra table can be added to the database with the command java lexicon. CreateSimpleAnalyzedWordforms <database host> <database name>. User name and password for the database are both assumed to be “impact”.

The tool creates sudirectories ModernLexicon and HistoricalLexicon in the target directory, and stores the wordform tries which are used by the variation pattern matcher in the file “modernWords.datrie”

5.8 Command Line Options

The tools’ behaviour is determined by a set of options which can be either specified on the command line or in a properties file. The command line options are:

```
-a,--alignmentOutput <arg>           output file for alignments
-b,--addWordBoundaries <arg>         add word boundaries ^ and  $ to
                                     strings before matching
-C,--minimumConfidence <arg>         minimum conditional probability lhs
                                     | rhs
-c,--command <arg>                   action to perform: train | test |
                                     run
-D,--allowDeletions <arg>            allow empty right hand side during
                                     matching or not (true|false)
-d,--outputDirectory <arg>           default output directory
-E,--echoTrainFile <arg>             Echo training set (reverse
                                     lemmatizer
-f,--properties <arg>                property file with job options
                                     use long option names in the
                                     properties file
-H,--databaseHost <arg>              Host for lexicon database
-h,--help                            print this message and exit
-I,--allowInsertions <arg>           allow empty left hand side during
                                     matching or not
-i,--trainFile <arg>                 input file for pattern training
-J,--minimumJointProbability <arg>   minimum joint probability for rule
                                     to be included in pattern matching
-L,--referenceLexicon <arg>          reference lexicon - ground truth for
                                     (reverse) lemmatization
-l,--lexicon <arg>                   lexicon (word list file to match to)
-M,--multigramLength <arg>           maximum multigram length
-m,--historicalLexicon <arg>         Historical lexicon file for lookup
-o,--patternOutput <arg>             output file for patterns
-P,--maximumPenalty <arg>            maximum matching penalty
-p,--patternInput <arg>              input file for patterns
-r,--pruner <arg>                    Java class used for determining
                                     which multigrams are acceptable
                                     patterns
-s,--maximumSuggestions <arg>        maximum number of matching
```

```
                                    suggestions
 -T,--lexiconTrie <arg>             Compiled Trie for Modern Lexicon
 -t,--testFile <arg>                input file for testing
 -X,--forbidInsertsAndDeletes <arg>  Do not save inserts and deletes in
                                    pattern output
 -x,--useOldPatternOutputMode <arg>  use old pattern output
 -y,--lemmatizerInput <arg>         Input for the lemmatizer, one word
                                    per line
 -z,--targetDirectory <arg>         Base directory for compiled lexicon
                                    Data
```

The option "-f <configuration file>" allows reading the options from a standard properties file using the long option names as keys, f.i.

```
modernLexicon=/data/Lexicon/OED/ModernLexicon
lexiconTrie=/data/Lexicon/OED/modernWords.datrie
historicalLexicon=/data/Lexicon/OED/HistoricalLexicon
patternInput=/data/Patterns/englishSpellingVariation.pat
```

5.9 Licensing

The licensing follows the consortium agreement.
The tool will be made available to the research community according to the regulations of the Dutch HLT agency (TST-Centrale, www.inl.nl), which means that it is freely available for non-commercial use.

6. IMPACT Corpus Based Lexicon Tool (CoBaLT)

6.1 Partner

INL

6.2 Deliverable

part of D-EE2.4

6.3 Background

The IMPACT Corpus Based Lexicon Tool (CoBaLT) is a tool for corpus based lexicon building. The tool allows the user to upload corpora that are to be used as attestations in a lexicon. An important requirement is that the tool should be fit to quickly process large quantities of data (see V.2).

6.4 Installation

CoBaLT is an AJAX application designed for Mozilla Firefox (though other browsers may work as well). It uses a MySQL database, and is written in PHP and Javascript.
Below some tweaks are discussed that make working with very large data sets possible. Do note however that the tool also works (on more moderately sized data sets) when all these components are installed 'out of the box' and no tweaking is done.

6.4.1 APACHE

A webserver is needed to handle the PHP request. Apache is by no means the only suitable web server around, but it is very widely spread and the tool was developed and tested on it.

Apache request limit

Sometimes POST requests can be very large due to very large amounts of data being sent to and from the server. It might be necessary therefore to increase the limit the web server imposes on such requests.
With Apache this can be done by setting the LimitRequestLine directive in the server's httpd.conf file.

```
LimitRequestLine 100000
```

With Apache it is necessary to restart the web server for this change to have effect.

6.4.2 PHP

As noted above, a webserver should be running that can handle PHP requests. The source code of the tool should be placed in a directory for which this PHP support is turned on.
The tool was tested on PHP 5.1.6 running on Red Hat Enterprise Linux Server release 5.2.

PHP memory limit

Because PHP reads through files uploaded to the tool, it is advisable, when the files are big, to set the maximum memory for PHP scripts to a value higher than what it usually is (16M in our case).
This can be done by setting the memory_limit variable in the php.ini file:

```
memory_limit = 256M      ; Maximum amount of memory a script may consume
```

Of course, the 256M can be any value you like.
It may be necessary to restart the webserver for this change to have effect.

PHP upload file size

For some systems the maximum size for a file that is uploaded is only 2 Mb. If you have files larger than this, you might want to increase this limit somewhat. This can be done by setting the upload_max_filesize in the php.ini file:

```
upload_max_filesize = 10M      ; Maximum amount of memory a script may consume
```

Of course, the 10M can be any value you like.
It may be necessary to restart the web server for this change to have effect.

Uploading multiple files, PHP zip support

PHP can not read directory listings, so it is not possible to upload an entire directory in one go. As a workaround, zip files can be used. When a directory is zipped it can be uploaded to CoBaLT.
In order for this to work PHP has to have its zip functionality enabled. Please refer to the PHP documentation for your platform and version.
As a reference, our `phpinfo()` shows this:

zip

Zip	enabled
Extension Version	$Id: php_zip.c,v 1.95.2.6 2007/05/19 22:35:49 pajoye Exp $
Zip version	1.8.10
Libzip version	0.7.1

MySQL support for PHP

The PHP installation needs to be configured with MySQL support. Usually the standard installations provides for this. If not, please refer to the PHP documentation on your system.

6.4.3 MySQL

In order to set up the MySQL database for the tool, root access is needed to the MySQL server.

The tool was tested on MySQL 5.0.45, running on Red Hat Enterprise Linux Server release 5.2.

Create databases

The tool expects at least two MySQL databases to be there with the right table structure. One is used as an index to the word forms in the corpora and one is used is for storing all the lexicon data.

NOTE that the first one (which is used as an index) has to be created only once per database host (so usually once in the lifetime of a distribution of the Lexicon Tool). Of the second type of database several instances can exist. Every new project will normally get its own database, but this token database is shared among all.

The distribution of the tool comes with two files, called sql/emptyLexiconTokenDatabase.sql and sql/emptyLexiconDatabase.sql. These files contain the data structures of both databases. They both need to be loaded into MySQL (though the first one only has to be loaded once, see above).

Before the table structure can be loaded, the databases need to be created. This is done in MySQL by running these queries:

```
mysql> CREATE DATABASE myLexiconTokenDb;   # If you have already
                                              # done this once, you can
                                              # skip this first one
mysql> CREATE DATABASE myLexiconDb;
```

In this query the italic part should be replaced by your own database name.

NOTE that in the php/globals.php file (see below) there is a variable, called $sTokenDbName, that holds the name of the token database. This variable and the database name used in the first query above should match.

Next, the two data structure files that come with the distribution should be loaded into the databases just created. On the command line you can do this by executing the following command (again, you can skip the first one if you have already done it once before):

```
% mysql -u root -p myLexiconTokenDb < emptyLexiconTokenDatabase.sql
% mysql -u root -p myLexiconDb < emptyLexiconDatabase.sql
```

Create a MySQL user

When a database has been created a MySQL user should be added. This can be done by running the following statements:

```
mysql> GRANT ALL ON myLexiconDb.* TO 'newUser'@localhost IDENTIFIED BY
'password';
mysql> GRANT ALL ON myLexiconDb.* TO 'newUser'@'%' IDENTIFIED BY 'password';
mysql> FLUSH PRIVILEGES;
```

Again, the italic parts in the queries should be set to your own desired values.

Fill the user table

The entire database can be empty at start up but for the user table which needs at least one row. New users can be added with the following statement:

```
mysql> INSERT INTO users (name) VALUES ('Amy'), ('Billy'), ('Duffy'),
('Ella');
```

The language table

If analyses should allow for languages (see below) the different options should be listed in this table.

Set group_concat_max_len

There are some queries, most notably the ones that gives the overview of the type-frequency list, that make extensive use of MySQL's built in GROUP_CONCAT() function. The result of this function can quite easily become larger than the default length that is allowed. There two MySQL server variables that control how much data can be sent by the server: @@max_allowed_packet and @@group_concat_max_len. To see what they are currently set to, the following query can be used:

```
mysql> SELECT @@group_concat_max_len;
mysql> SELECT @@max_allowed_packet;
```

It is advisable to set these values to a value of 64 Mb or higher. This can be done by this statement:

```
mysql> SET GLOBAL max_allowed_packet = 67108864;
mysql> SET GLOBAL group_concat_max_len = @@max_allowed_packet;
```

6.4.4 Adjust to local environment

There are two places in the code that need to be adjusted to the local environment.

lexiconTool.php

This file contains a form called loginForm stating the available databases. This form should be altered so it states all the available databases.

globals.php

The file php/globals.php contains all environment specific variables. The global database parameters should be modified to match the values used in the 'Make a MySQL user' part above.

The document root can be set to any available directory, as long as this directory is readable and writable for the tool. It is advisable to avoid having this directory in the same directory the application is in because it might in that case accidentally be thrown away when a new version of the tool is installed.

The tokenizer directory should simply point to the directory the tokenizer is in

6.4.5 Tokenizing

Tokenizer

Any untokenized document will be tokenized by the tokenizer integrated in the tool. However, it is also possible to upload documents that are tokenized already, which will be recognized as such automatically. In order to determine whether or not a file is tokenized the tool looks at the first 10 lines to see if they look ‘tokenized’ like this:

canonicalForm1<TAB>wordForm<TAB>onset<TAB>offset<TAB>....

So, e.g. the next bit could be the output of a tokenizer for the text “Hello world!”:

```
Hello       Hello       0     5
world       world!      6     12
```

helloWorld_tokenized.tab

Some character sequences might not be interesting for lexical processing in the tool even though it would be nice to see them displayed for ease of reading. This goes e.g. for punctuation marks that appear separately in a text.

Text like this, that should be displayed but which should not be treated as a wordform in the tool should have the string 'isNotAWordformInDb' in the fifth column of the tokenized file. Other words (as in the example above) should have an empty fifth column. So PLEASE NOTE that in that case a tab will end the line.

Tokenizing XML

The default tokenizer can also handle XML files in certain formats. Currently the supported formats are IGT XML (the IMPACT ground truth format) and TEI (as provided by JSI).

Position information for OCR'ed material

When working with OCR'ed material there is a relation between the text appearing in the tool and the original image. The tool supports the relation between the two (see section 'Viewing a word form in the original image' below). The 'documents' table in the MySQL database has a column 'image_location' where a path to the image can be set. Also, the tokenized file can contain information in the sixth to ninth column. These columns should specify x-coordinate, y-coordinate, height and width, respectively.

Set execute permissions

For the tool to be able to call the tokenizer the right permissions should be set. The web server runs as a certain user (e.g. 'www-data'). It is necessary for this user to have execute permissions in the tokenizer directory and for the tokenizer executable itself.

For Ubuntu it has proven necessary to add the web server user to the sudoers file and to preprend the call to the tokenizer with a 'sudo' command.

There is a setting for this in the php/globals.php file, called $sSudo.

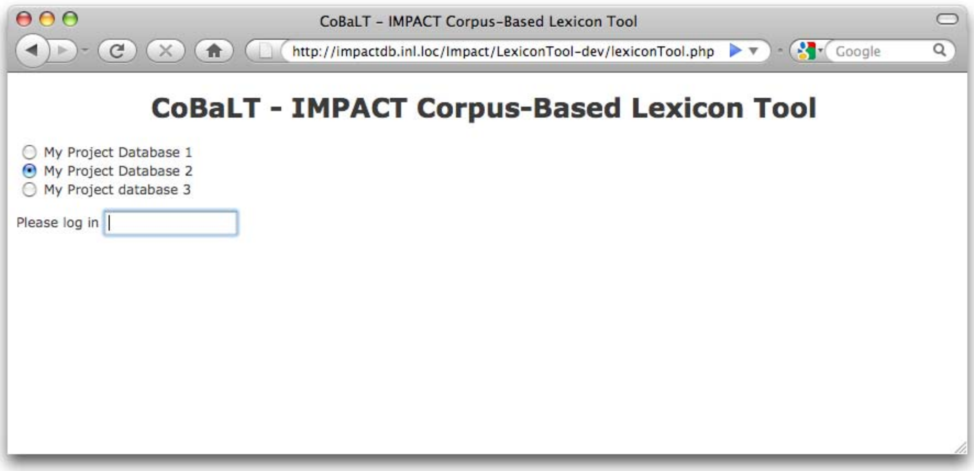
6.5 Troubleshooting

Usually if something isn't working it has to do with permissions. The tool should have read/write access to the right folders.
A good place to start looking when something appears not to work is the error log file of the web server. For Apache it is simply called 'error_log' and on Red Hat Linux it is located in the directory /var/log/httpd/.

Also it might be a good idea to set the display_errors directive in the php.ini file to 'On' (it is 'Off' by default).

6.6 The user interface

6.6.1 Log-in screen


CoBaLT – IMPACT Corpus-Based Lexicon Tool

http://impactdb.inl.loc/Impact/LexiconTool-dev/lexiconTool.php

CoBaLT - IMPACT Corpus-Based Lexicon Tool

( ) My Project Database 1
(●) My Project Database 2
( ) My Project database 3

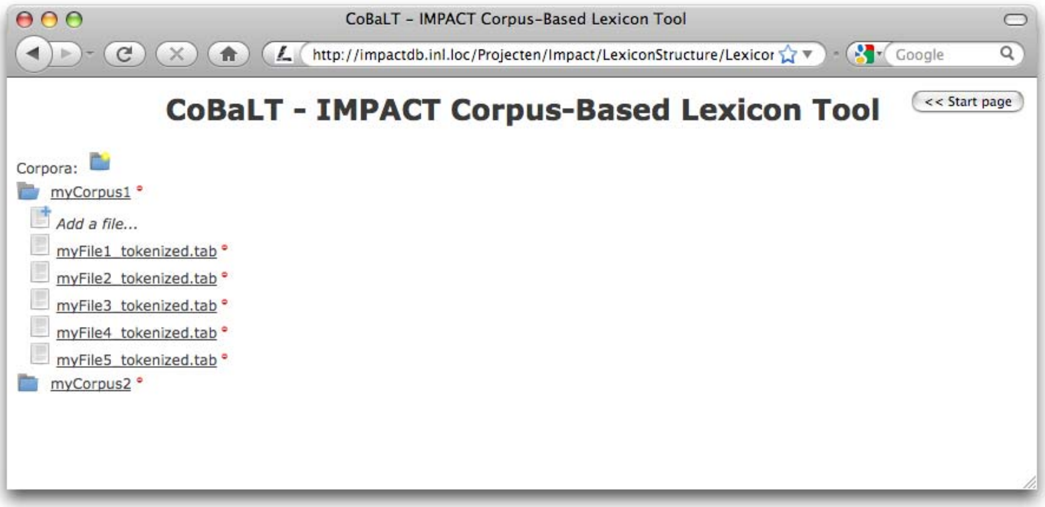
Please log in

Figure 1 - screenshot of CoBaLT – log-in screen

The first screen is a log-in screen. The different choices listed here should have been set in the first 'Adjust to local environment' step described above. The user simply types in his/her name and chooses a database to work on.

6.6.2 Corpus screen

In the next screen the user can choose a corpus to work on, corpora can be deleted, new corpora can be added and files can be added to or deleted from corpora.


CoBaLT - IMPACT Corpus-Based Lexicon Tool
http://impactdb.inl.loc/Projecten/Impact/LexiconStructure/Lexicor
Google
<< Start page

CoBaLT - IMPACT Corpus-Based Lexicon Tool

Corpora:
myCorpus1
Add a file...
myFile1_tokenized.tab
myFile2_tokenized.tab
myFile3_tokenized.tab
myFile4_tokenized.tab
myFile5_tokenized.tab
myCorpus2

Figure 2 - screenshot of CoBaLT – corpora screen

Corpus mode vs document mode

The tool can work in two modes: corpus mode and document mode.

By clicking on a corpus name the tool will load in corpus mode (i.e. the user will work on a list of word forms for an entire corpus). By clicking on a document name the tool will load in document mode, which means that only that particular document is taken into account.

Loading documents

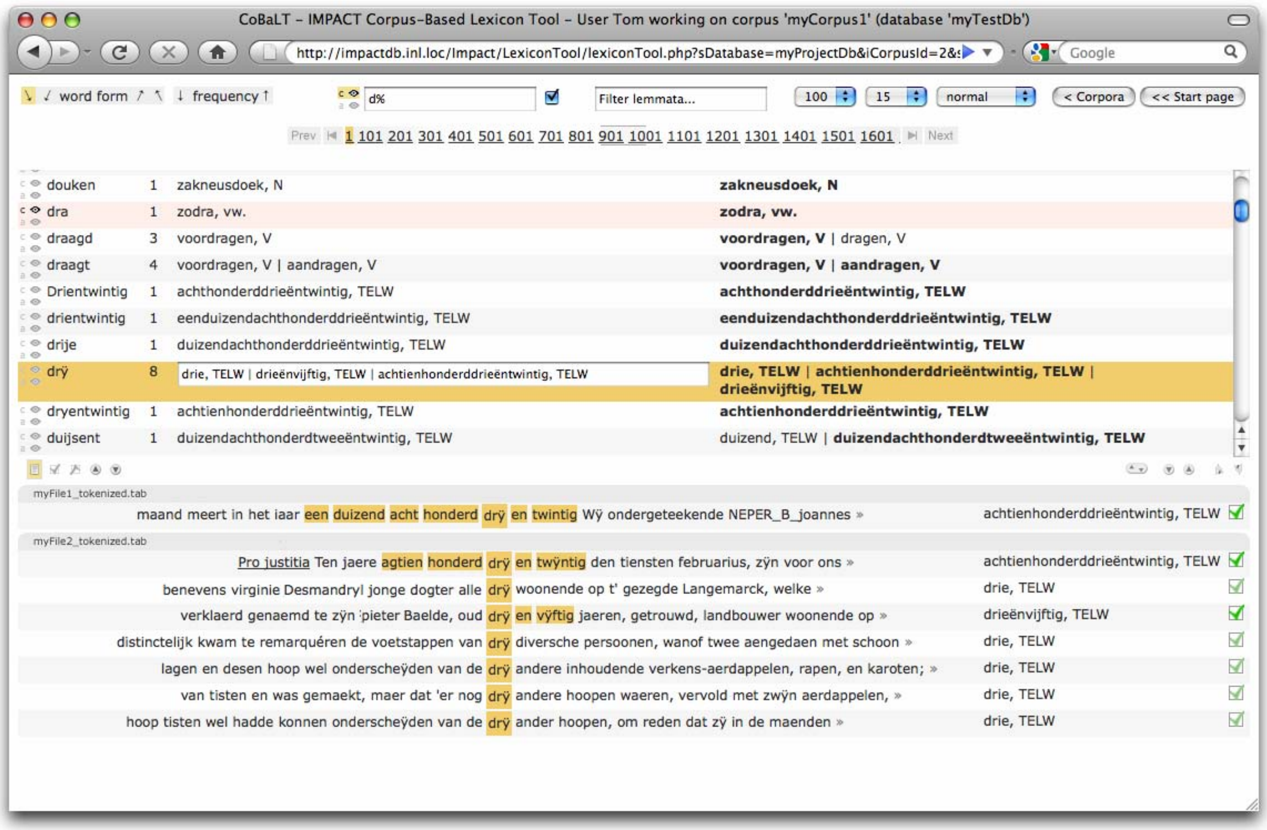
Documents can be loaded into the tool one by one or a lot of them together in a zip file. The tool checks filename extensions. Only .txt files .tab files and .xml files are processed.

Please note that every file should be utf-8 encoded. No other encoding format is supported.

Any untokenized document will be tokenized by the tokenizer integrated in the tool. However, it is also possible to upload tokenized documents which will be recognized as such automatically. This is particularly handy if you need your tokenization to be different from what the internal tokenizer does.

Please refer to section above about tokenizing to see what a tokenized file should look like.

6.6.3 Main screen



CoBaLT – IMPACT Corpus-Based Lexicon Tool – User Tom working on corpus 'myCorpus1' (database 'myTestDb')

http://impactdb.inl.loc/Impact/LexiconTool/lexiconTool.php?sDatabase=myProjectDb&iCorpusId=2&

word form ↓ frequency ↑ | d% | Filter lemmata... | 100 | 15 | normal | < Corpora | << Start page

Prev 1 101 201 301 401 501 601 701 801 901 1001 1101 1201 1301 1401 1501 1601 Next

douken	1	zakneusdoek, N	**zakneusdoek, N**
dra	1	zodra, vw.	**zodra, vw.**
draagd	3	voordragen, V	**voordragen, V** | dragen, V
draagt	4	voordragen, V | aandragen, V	**voordragen, V | aandragen, V**
Drientwintig	1	achthonderddrieëntwintig, TELW	**achthonderddrieëntwintig, TELW**
drientwintig	1	eenduizendachthonderddrieëntwintig, TELW	**eenduizendachthonderddrieëntwintig, TELW**
drije	1	duizendachthonderddrieëntwintig, TELW	**duizendachthonderddrieëntwintig, TELW**
drÿ	8	drie, TELW | drieënvijftig, TELW | achtienhonderddrieëntwintig, TELW	**drie, TELW | achtienhonderddrieëntwintig, TELW | drieënvijftig, TELW**
dryentwintig	1	achtienhonderddrieëntwintig, TELW	**achtienhonderddrieëntwintig, TELW**
duijsent	1	duizendachthonderdtweeëntwintig, TELW	duizend, TELW | **duizendachthonderdtweeëntwintig, TELW**

myFile1_tokenized.tab

maand meert in het iaar een duizend acht honderd drÿ en twintig Wÿ ondergeteekende NEPER_B_joannes »	achtienhonderddrieëntwintig, TELW	✓

myFile2_tokenized.tab

Pro justitia Ten jaere agtien honderd drÿ en twÿntig den tiensten februarius, zÿn voor ons »	achtienhonderddrieëntwintig, TELW	✓
benevens virginie Desmandryl jonge dogter alle drÿ woonende op t' gezegde Langemarck, welke »	drie, TELW	✓
verklaerd genaemd te zÿn pieter Baelde, oud drÿ en vÿftig jaeren, getrouwd, landbouwer woonende op »	drieënvijftig, TELW	✓
distinctelijk kwam te remarquéren de voetstappen van drÿ diversche persoonen, wanof twee aengedaen met schoon »	drie, TELW	✓
lagen en desen hoop wel onderscheÿden van de drÿ andere inhoudende verkens-aerdappelen, rapen, en karoten; »	drie, TELW	✓
van tisten en was gemaekt, maer dat 'er nog drÿ andere hoopen waeren, vervold met zwÿn aerdappelen, »	drie, TELW	✓
hoop tisten wel hadde konnen onderscheÿden van de drÿ ander hoopen, om reden dat zÿ in de maenden »	drie, TELW	✓

Figure 3 - screenshot of the CoBaLT – main screen

In the screenshot above we see CoBaLT in action. It is running in corpus mode, which means that the user selected a collection of documents to work on, rather than just one. In the latter case the tool would have run in document mode. The interface is divided in three main parts.

- Top bar
  For sorting and filtering word forms and lemmas, and adjusting the views on the data.
- Middle part
  Shows the type-frequency list plus analyses in the corpus/document and database.
- Lower part
  Shows the individual occurrences of word forms ('tokens') in context plus their analyses.

Adjusting the size of the window parts

The ratio between the middle part and the lower part can be altered by dragging the resize icon (circled in red in the partial screenshot below).

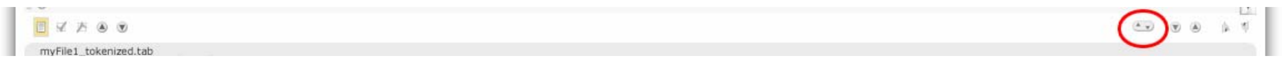


Figure 4 - partial screenshot of CoBaLT – resize window parts

Top scroller bar

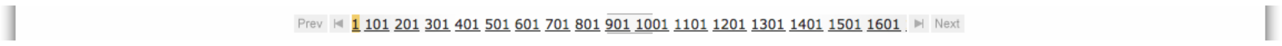


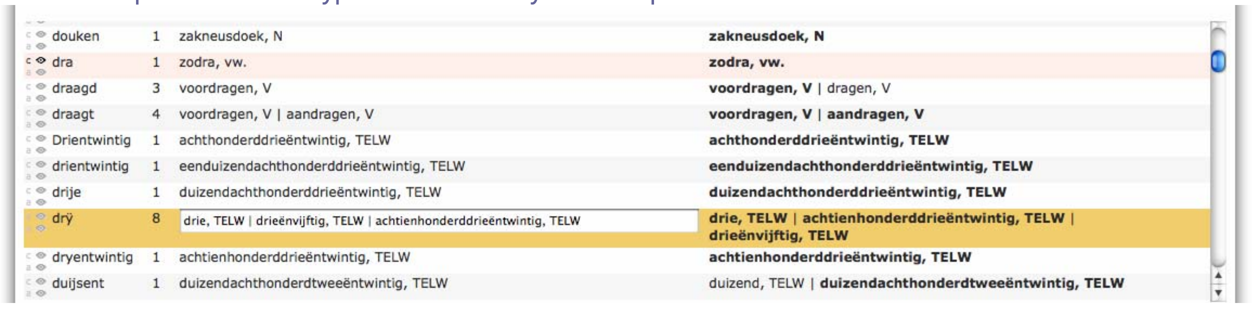
Figure 5 - partial screenshot of CoBaLT – top scroller bar

When there is a large amount of pages available a scroller bar will appear at the top of the screen. Hovering the mouse cursor a the right side of the middle will scroll the list of pages to the left and vice versa. To stop the list from scrolling move the mouse cursor to the middle of the bar.

In the following it is described how analyses can be edited in the lower as well as in the middle part, what analyses may consist of, and how to customize the ways in which the data are presented.

An overview of key functions and mouse actions can be found below. Note that with most buttons and also with the analyses in the right-hand column in the middle part, an explanatory box appears when the mouse cursor is hovered over it.

The middle part: Word form types and their analyses in corpus/document and database



douken	1	zakneusdoek, N	**zakneusdoek, N**
dra	1	zodra, vw.	**zodra, vw.**
draagd	3	voordragen, V	**voordragen, V** | dragen, V
draagt	4	voordragen, V | aandragen, V	**voordragen, V** | **aandragen, V**
Drientwintig	1	achthonderddrieëntwintig, TELW	**achthonderddrieëntwintig, TELW**
drientwintig	1	eenduizendachthonderddrieëntwintig, TELW	**eenduizendachthonderddrieëntwintig, TELW**
drije	1	duizendachthonderddrieëntwintig, TELW	**duizendachthonderddrieëntwintig, TELW**
drÿ	8	drie, TELW | drieënvijftig, TELW | achtienhonderddrieëntwintig, TELW	**drie, TELW** | **achtienhonderddrieëntwintig, TELW** | **drieënvijftig, TELW**
dryentwintig	1	achtienhonderddrieëntwintig, TELW	**achtienhonderddrieëntwintig, TELW**
duijsent	1	duizendachthonderdtweeëntwintig, TELW	duizend, TELW | **duizendachthonderdtweeëntwintig, TELW**

Figure 6 - partial screenshot of CoBaLT – middle part

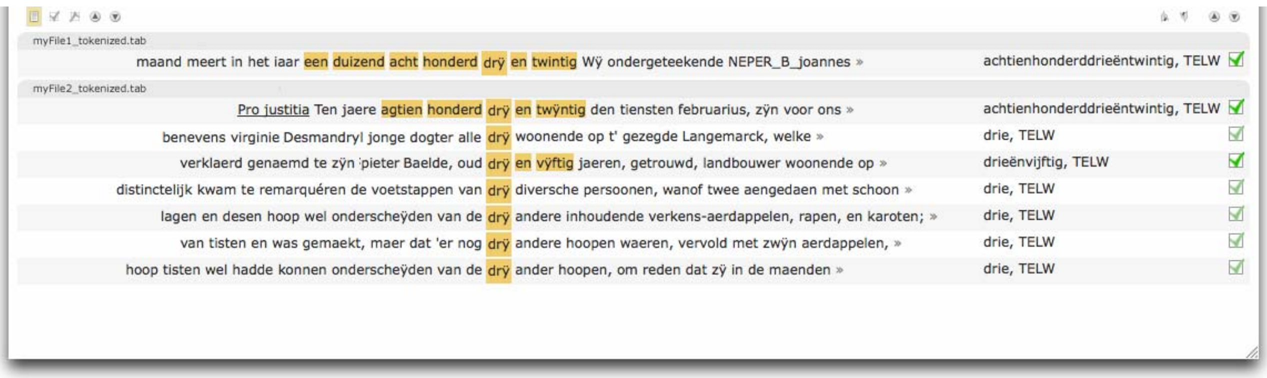
In the middle part of the screen a type-frequency list is displayed with some additional options and information. A selected row is editable, allowing the user to add/delete analyses. A row is selected by clicking on it; it will be highlighted yellow (unless it is hidden, in which case it is displayed pink; see below). Only one row at a time can be selected; by using arrows the previous or the next row becomes selected.

The middle column displays the analyses a word form has in the loaded corpus or document (depending on the mode).
The rightmost column in the middle part shows all the analyses linked to a word form in the entire database; those in bold are validated (see below).

When a word form is selected (like *drÿ* in the example screenshot), its occurrences are loaded in the lower part of the screen together with some context.

As actions performed in the middle part apply to the tokens listed in the lower part, the functioning of the latter will be explained first.

The lower part: occurrences in context



myFile1_tokenized.tab

maand meert in het iaar een duizend acht honderd	drÿ	en twintig Wÿ ondergeteekende NEPER_B_joannes »	achtienhonderddrieëntwintig, TELW ✓

myFile2_tokenized.tab

Pro justitia Ten jaere agtien honderd	drÿ	en twÿntig den tiensten februarius, zÿn voor ons »	achtienhonderddrieëntwintig, TELW ✓
benevens virginie Desmandryl jonge dogter alle	drÿ	woonende op t' gezegde Langemarck, welke »	drie, TELW ✓
verklaerd genaemd te zÿn 'pieter Baelde, oud	drÿ	en vÿftig jaeren, getrouwd, landbouwer woonende op »	drieënvijftig, TELW ✓
distinctelijk kwam te remarquéren de voetstappen van	drÿ	diversche persoonen, wanof twee aengedaen met schoon »	drie, TELW ✓
lagen en desen hoop wel onderscheÿden van de	drÿ	andere inhoudende verkens-aerdappelen, rapen, en karoten; »	drie, TELW ✓
van tisten en was gemaekt, maer dat 'er nog	drÿ	andere hoopen waeren, vervold met zwÿn aerdappelen, »	drie, TELW ✓
hoop tisten wel hadde konnen onderscheÿden van de	drÿ	ander hoopen, om reden dat zÿ in de maenden »	drie, TELW ✓

Figure 7 - partial screenshot of CoBaLT – lower part

The lower part of the screen shows the occurrences (tokens) in context of the word form selected in the middle part. The right-hand column lists the analyses of the word forms in each context.

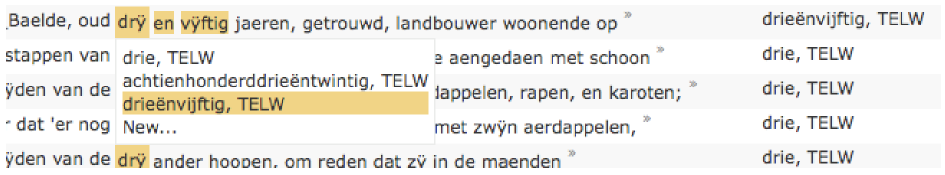
Each occurrence is presented with some context (see below on how to adjust the amount of context in view, or the number of context rows per page). Token plus context are selected and deselected by clicking anywhere in the row[26]. Non-adjacent rows can be selected by Ctrl-clicking (clicking while holding down the Ctrl key). If you hold down the mouse key while going over a couple of rows, you select those. By double clicking one row, you select all rows. In order to deselect a single row, Ctrl-click it. In this part of the tool, any action that is performed is applied to all rows that happen to be selected. Note that not every row that is selected is necessarily in view; see below on the number of context rows per page.

Editing analyses on the token level

Analyses can be added, removed, and validated in various ways on interdependent levels. The most specific level is that of a token in context.

By clicking on a yellow-highlighted word in its context in the lower part of the screen, a drop-down menu appears which lists all corresponding analyses in the database, with those analyses already assigned to the token in question highlighted. By switching analyses on or off they will be (de)assigned to the token(s) in the selected row(s). The option *New...* at the bottom of the menu provides the possibility to type in a new analysis in a text box, with existing analyses being suggested as you type.

[26] You might want to avoid clicking on the analyses on the right though as these will be deleted when you do that.



Baelde, oud drÿ en vÿftig jaeren, getrouwd, landbouwer woonende op » drieënvijftig, TELW
stappen van | drie, TELW | e aengedaen met schoon » drie, TELW
ÿden van de | achtienhonderddrieëntwintig, TELW | dappelen, rapen, en karoten; » drie, TELW
dat 'er nog | drieënvijftig, TELW | met zwÿn aerdappelen, » drie, TELW
New...
ÿden van de drÿ ander hoopen, om reden dat zÿ in de maenden » drie, TELW

Figure 8 - partial screenshot of CoBaLT – close up of analysis drop-down menu

The rightmost column shows the analyses for the occurrence in this context. Clicking on one of them will result in this analysis being deleted from the row it is in, as well as from any other selected rows it featured in.

One analysis for a group of words

In the example screenshot the words *drÿ*, *en,* and *vÿftig* are marked together to make up the analysis *drieënvijftig, TELW*. You can add words to a group, or delete them from it by clicking on them while holding down the F2 or F9 key. If a word you just added to the group already has analyses of its own in this particular context, these will show in the right-hand column.

More lemmata for one word form

Conversely, a word form may occasionally consist of more than one lemma. Multiple analyses can be assigned by using an ampersand (&); e.g., *l'équipe* could be analysed as *la, DET & équipe, N*.

Validation

To indicate whether attestations and analyses are verified by a user, they can be validated.

Word forms in their context can be validated (regardless of whether they have an analysis or not) by checking a validation box at the right of the context row in the lower part of the screen. This can be interpreted as 'the user saw the word form in this context and approves of the listed analyses'. As with any other action in this part of the screen, (de)validation is applied to any sentences that happen to be selected.

By assigning an analysis to a token attestation in the lower part of the screen (either by choosing an option from the drop-down list as described above or by an action in the middle part as will be described below), the token plus its analyses become validated automatically.

When the validation checkbox is grayed out a bit (as in the second and seventh row in the example screenshot) this means that this occurrence of the word form in this context was validated by a different user.

When a token is attested in its context or is validated, the analyses associated with the word form in this context automatically become validated. Validated analyses are displayed in **bold** in the right column of the middle part in the tool (in the example screenshot nearly all analyses are validated).

Analyses can also be (un)validated 'by hand' by Shift-clicking them (clicking while holding down the Shift key).

NOTE that it is not necessary for a validated type analysis to have a token attestation. E.g., the analysis *duizend, TELW* for the word form *duijsent* in the example screenshot is currently unvalidated. It only occurs once in the corpus, apparently as part of a group *duizendachthonderdrieëntwintig*. A user might decide to validate the analysis *duizend, TELW* nevertheless, even though there is no attestation to support it.

Editing analyses in the middle part

The rightmost column in the middle part of the screen shows all the analyses a word form (type) has in the entire database. As mentioned above, those in bold are validated; they can be (un)validated by Shift-clicking on them.

By clicking on an analysis it is applied to the rows that are selected in the lower part of the tool, which will in turn become validated (shown by the checkbox at the right being checked). When no row is selected, the analysis will be assigned to all tokens in the lower part, but these will NOT become validated.
The idea behind this is that in this way it is easy/fast to, e.g., when a word has more than one analysis, assign these to all occurrences of a word form without further disambiguation. If the user goes through the trouble of selecting one or more rows (s)he must be pretty sure about it, and the analyses become validated.

An analysis for a certain word form can be deleted altogether by Ctrl-clicking it in the right-hand column. As a consequence of course, the analysis disappears as well for any token attestation for the word form in question it featured in.

As with the analyses of the entire database, the corpus/document analyses filled in in the text box in the middle part will be applied to the selected row(s) in the lower part of the tool which then will also become validated. Again, when no row is selected in the lower part, the analyses assigned in the middle part will be applied to all token attestations without them being validated.

Which analyses appear in which column?

The lower part of the screen shows occurrences in context from the corpus/document. By assigning analyses to these occurrences the word forms become attested. As said earlier, the middle column in the middle part of the screen shows the analyses a word form has in the current corpus/document. So these will match with the ones in the lower part of the screen. The analyses in the rightmost column however do not necessarily show anywhere, which may be a source of confusion. This can be the case if these analyses are for word forms in a document in the same database but in a different corpus, or because they are not associated with any word form in context at all (i.e. the analysis *is* associated with the word form, or it wouldn't show in the first place, but there is no attestation in context anywhere yet). The latter may e.g. be the case when a database comes preloaded with information from an external lexicon/dictionary (e.g. the analysis *duizend, TELW* for the word form *duijsent* in the example screenshot).

On word form analyses

What do analyses look like?

An analysis as it is used in the tool (and this manual) refers to a tuple that can be described as:

modern lemma (, <modern_equivalent>) (, [set of patterns]) , part of speech (, language) (, gloss)

The parts between round brackets are optional. Only the lemma and the part of speech are obligatory, so a typical, simple lemma would be e.g. *the, DET*.

Modern equivalent and patterns

Optionally, a modern equivalent word form and some patterns can be specified in an analyses.

E.g. for the German word form *theyle* an analysis could be *teil, <teile>, [(t_th, 0), (ei_ey, 2)], N*. The part between angled brackets (<>) is the modern equivalent word form (which is possibly inflected) and the part in between the square brackets represents a series of pattern substitutions to get from the modern equivalent to the historical one. In the example, the substitution *th* for *t* should be applied at position *0* (the first character).

Neither modern equivalent nor patterns are obligatory. Nor are they required to be specified both. In other words

teil, [(t_th, 0), (ei_ey, 2)], N and *teil, <teile>, N*

are valid lemmas as well.

Language

Next to the part of speech, a language may be specified. This could be used e.g. to keep Latin phrases apart from other text.

Gloss

To be able to keep similar lemmas apart, a gloss may be added. E.g. *paper, N, material* versus *paper, N, essay* or *paper, N, newspaper*.

NOTE that if language names are to be used the table 'languages' in the database should be filled with the various options. It is only when the first word after the part of speech in the analyses matches one of these options that it is treated as a language. If it doesn't match it is treated as (part of) the gloss.

One word form, more lemmas

Sometimes a word form might better be analysed as consisting of two or more lemmas rather than one. In the tool this can be done by separating analyses by ampersands (&'s). E.g. *l'Afrique* could be analysed as *le, DET & Africa, NELOC*.

NOTE that in these cases the analysis cannot contain modern equivalents or patterns.

One analysis for several words

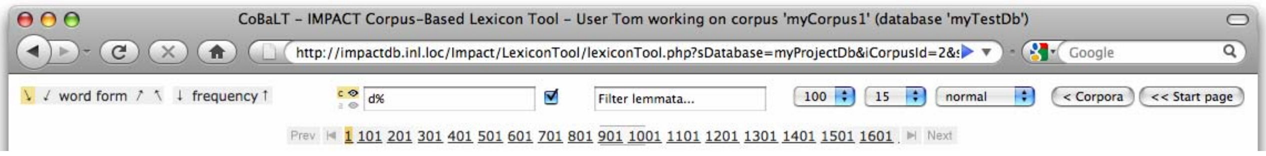
Sometimes, what can be thought of as one word appears as two or more. Consider e.g. separable verbs, which are very common in Dutch. *Meedoen* means to participate, and *"I participate"* would translate to *"Ik doe mee"*. In this phrase, the word forms *doe* and *mee* together make up the analysis *meedoen, V*.

The word forms forming one lemma do not have to be next to each other. *"Nina en Ella <u>doen</u> morgen <u>mee</u>"* (*"Nina and Ella will participate tomorrow"*) can be analysed as well by clicking on them while holding down the F2 or F9 key.

Customizing the views on the data

Sorting

The type-frequency list in the middle part of the tool can be sorted by using the arrows at the left side of the top bar. The list can be sorted alphabetically by word form, or by frequency. The alphabetical ordering can also be done from right to left (so e.g*.* *blueish, reddish* and *greyish* will appear near each other).


CoBaLT – IMPACT Corpus-Based Lexicon Tool – User Tom working on corpus 'myCorpus1' (database 'myTestDb')

http://impactdb.inl.loc/Impact/LexiconTool/lexiconTool.php?sDatabase=myProjectDb&iCorpusId=2&

Google
word form
frequency
d%
Filter lemmata...
100
15
normal
< Corpora
<< Start page
Prev 1 101 201 301 401 501 601 701 801 901 1001 1101 1201 1301 1401 1501 1601 Next

Figure 9 - partial screenshot of CoBaLT – top bar

Filter on word forms

There are two filterboxes in the top bar. The filter in the left-hand one applies to the word forms in the type-frequency list. You can filter on word form beginnings, endings or, in fact, anything in between. The filter is directly passed to the MySQL LIKE operator.

For not-MySQL-guru's: the most frequently used wildcard is % which means any sequence of characters. So %a% means: any word form matching an arbitrary sequence of characters, then an *a* and then possibly some more characters. So *ball* would match, as would *dance* or *pizza* or indeed any word form with an *a* in it (including the word *a* itself). In the screenshot d% means all the word forms starting with a *d* (so *dance* would match again, but e.g. *adorable* wouldn't).
For further documentation please refer to the MySQL documentation.

The filter is case-insensitive by default, but unchecking the box next to it will make it case-sensitive.

To de-activate a filter, empty the filterbox and apply it (by hitting 'Enter').

Filter by lemma

To the right of the box for filtering word forms is a box for filtering by lemma. In this box you can type in a lemma and its part of speech, separated by a comma (e.g., *lemma, NOU*) and only word forms that have this lemma assigned to them are shown. Please note that patterns are not supported in this box, only complete analyses.

Edit a lemma or delete it from the database

When a lemma filter is applied and matches with a lemma, an additional icon appears next to the lemma filter box. By clicking on this icon a new sub window opens in which a lemma can be edited or deleted.

Please note that editing or deleting a lemma will apply to that particular lemma in the entire database (not just the corpus selected).

Hiding/showing word forms

There can be various reasons for hiding word forms. It could be, e.g., that one feels that words in certain closed categories (let's say determiners like *the* and *a*) have been dealt with sufficiently and there is no need to analyse them time and time again for every new document or corpus. Or it could be convenient for a particular task to temporarily hide all word forms that have no characters in them (so e.g. cipher words will not show in the list).

At the left side of each row in the middle part of the screen there are two buttons labeled *c* and *a* in corpus mode (for "don't show in the corpus" and "don't show at all", respectively), or *d* ("don't show for this document) and *a* in document mode. The *d* button (only visible in document mode) is for hiding the word form of that row for the current document. The *c* button (only visible in corpus mode) is for hiding the word form in that row for the current corpus.

The *a* button is for hiding the word form for the entire database regardless of corpus or document. “Never show me this word again!”.

By switching on one of these buttons the row will be shown as hidden and displayed in pink. When the type-frequency list is reloaded, e.g., when a new filter is applied, or the user logs in again, the word form in question will not show again, unless the relevant show/hide button in the top bar is switched on.

In the top bar, just left of the filtering box, there are two buttons for showing, or hiding again, hidden word forms. They too are labeled *c* and *a* in corpus mode, or *d* and *a* in document mode. By default, these buttons are switched off (i.e., hidden word forms are not shown).

Word forms may be marked — by using the buttons in the middle part, or by means of a script — to be e.g. "hidden for this corpus". If the *c* button at the top is inactive, the word form will not be shown. The word forms are 'unhidden' if the *c* button is activated.

E.g., the row for *dra* in the example screenshots above is hidden for the current corpus, but is shown nevertheless because also the *c* button is switched on in the top bar, which means, "do show all word forms that were hidden for this corpus".

Number of word forms per page

The number of word forms displayed per page in the middle part is set on 100 by default. In the top bar, to the right of the filtering boxes, this can be changed into 10, 20, 50, 100 or 'all'. The latter means that all word forms that match the filter are displayed on a single page. In a large corpus this set can be very large, resulting in a long loading time.

The horizontal bar with the page numbers (that says 0 100 200 in the example screenshot) is shown only if there is more than one page to be displayed. By clicking on a number in this bar you jump to the corresponding page and the number will be highlighted. If there are a lot of word forms, the bar becomes scrollable.

Number of context rows per page

When working with large corpora some word forms might occur very often. By clicking on them in the middle part of the screen all the contexts they occur in will be loaded in the lower part, which may slow down things considerably. Because of this, the number of context rows shown per word form per page is 15 by default. Do note however that the speed advantage is most striking when the sentences are not sorted, which is the default (‘sorted by document’). If they are sorted, the tool has to actually collect *all* the sentences in the background, sort them, and then show a subset of them, so this is somewhat slower.

The number of context rows to be loaded in one page can be adjusted in the top bar, from 15 up to ‘all’. If there are more rows than fit on one page, a horizontal bar appears with clickable numbers to go to other pages.

Beware that with more rows per page than visable on the screen, rows may happen to be selected, and thus affected by actions, without actually being in view.

Amount of context

The amount of context surrounding the word form occurrences in the lower part of the tool, by default set to ‘normal’, can be increased by a drop-down menu in the top bar.

The user can also see more context for a particular token by clicking the » (guillemet) at the right of the context row in question. A pop-up window will appear in which more context is shown. In this window the user can be shown even more context. Or even more, as long as there is more context to show.

Sorting the tokens in context

By default, the context rows in the lower part of the tool are displayed in the order in which they appear in the documents (‘sorted by document’).

Sort rows by context

It can be convenient however to sort the rows by the immediate context of the occurring word forms. By clicking on the small arrow buttons at the left side the rows will be sorted according to the words to the left of the token (either ascending or descending).

For the example screenshot these would be:

benevens virginie Desmandryl jonge dogter alle	drÿ	[…]
hoop tisten wel hadde konnen onderscheÿden van de	drÿ	[…]
lagen en desen hoop wel onderscheÿden van de	drÿ	[…]
maand meert in het iaar een duizend acht honderd	drÿ	[…]
Pro justitia Ten jaere agtien honderd	drÿ	[…]
van tisten en was gemaekt, maer dat 'er nog	drÿ	[…]
verklaerd genaemd te zÿn pieter Baelde, oud	drÿ	[…]
distinctelijk kwam te remarquéren de voetstappen van	drÿ	[…]

By using the arrow buttons at the right side of the screen the rows can be sorted by the right-hand side of the context. For the example screenshot this would be:

[…]	drÿ	ander hoopen, om reden dat zÿ in de maenden
[…]	drÿ	andere hoopen waeren, vervold met zwÿn aerdappelen,
[…]	drÿ	andere inhoudende verkens-aerdappelen, rapen, en karoten;
[…]	drÿ	diversche persoonen, wanof twee aengedaen met schoon
[…]	drÿ	en twintig Wÿ ondergeteekende joannes
[…]	drÿ	en twÿntig den tiensten februarius, zÿn voor ons
[…]	drÿ	en vÿftig jaeren, getrouwd, landbouwer woonende op
[…]	drÿ	woonende op t' gezegde Langemarck, welke

Sort rows by validation

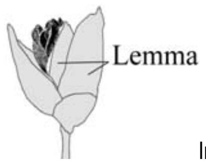
Rows can be sorted by validation by clicking on one of the checkbox buttons next to the 'sort by left context' buttons. This way all validated rows are grouped at the top/bottom, so it is e.g. easy to see what has been done and still needs to be done.

Sort rows by lemma

At the right-hand side, there are two buttons for sorting by lemma.[27] Clicking on of these buttons results in the rows being sorted by their analyses.

Again, for the sample screenshot this would amount to:

[…]	drÿ	[…]	achtienhonderddrieëntwintig, TELW
[…]	drÿ	[…]	achtienhonderddrieëntwintig, TELW
[…]	drÿ	[…]	drieënvijftig, TELW
[…]	drÿ	[…]	drie, TELW
[…]	drÿ	[…]	drie, TELW
[…]	drÿ	[…]	drie, TELW
[…]	drÿ	[…]	drie, TELW
[…]	drÿ	[…]	drie, TELW

27 

In case you are wondering what the little icons are supposed to look like… they are schematic representations of 'lemmas'. A lemma (also) is a "phytomorphological term used in botany referring to a part of the spikelet of grasses (Poaceae). It is the lowermost of two chaff-like bracts enclosing the grass floret". (Wikipedia: http://en.wikipedia.org/wiki/Lemma_%28botany%29)

Back to corpus page/start page

At the right-hand side of the top bar there are two buttons. One, labeled *Corpora*, for going back to the available corpora in the current database. The other one, labeled *Start page*, gets you back to the start screen where a database can be selected.

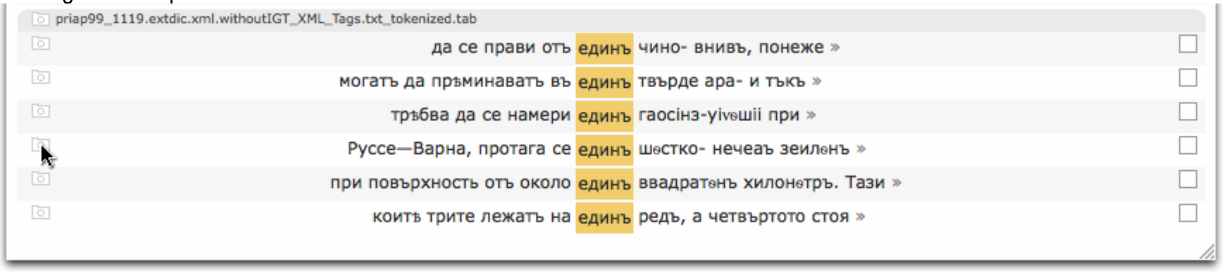
Working with OCR'ed material

The Lexicon Tool is a word form based tool. Word forms are grouped together and can be processed together. It is not always the case though that the word forms themselves are definite. Especially when working with OCR'ed material it could be the case that an OCR error occurred and that a particular word form was mistaken for another.

If the right data is available, the word form can be viewed in the tool in the original image it was scanned from. This way it is very easy, with one mouse click, to see whether or not the word was scanned correctly. If not, the tools allowes you to edit the word form.

Viewing a word form in the original image

If the right data is provided in the database some extra icons will show in the tool.

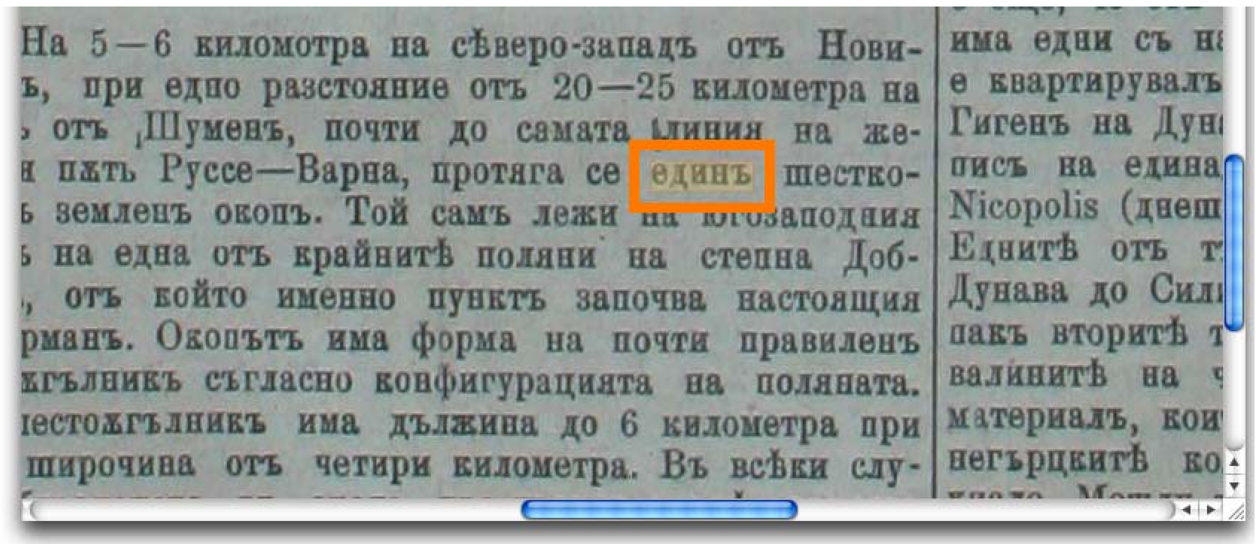


priap99_1119.extdic.xml.withoutIGT_XML_Tags.txt_tokenized.tab

	да се прави отъ	единъ	чино- внивъ, понеже »	☐
	могатъ да прѣминаватъ въ	единъ	твърде ара- и тъкъ »	☐
	трѣбва да се намери	единъ	гаосінз-уіvөшіі при »	☐
	Русе—Варна, протага се	единъ	шөстко- нечеаъ зеилөнъ »	☐
	при повърхность отъ около	единъ	ввадратөнъ хилонөтръ. Тази »	☐
	коитѣ трите лежатъ на	единъ	редъ, а четвъртото стоя »	☐

Figure 8 - partial screenshot of CoBaLT – display image

Clicking on a 'view image' icon in the document title bar will open a new window with the image. If an icon in a word form row is clicked on, the same image will load and the word form in question will be highlighted.



На 5—6 киломотра на сѣверо-западъ отъ Нови-
ъ, при едно разстояние отъ 20—25 километра на
отъ Шуменъ, почти до самата линия на же-
ı пжть Руссе—Варна, протяга се единъ шестко-
ь земленъ окопъ. Той самъ лежи на югозападния
ь на една отъ крайнитѣ поляни на степна Доб-
, отъ който именно пунктъ започва настоящия
рманъ. Окопътъ има форма на почти правиленъ
жгълникъ съгласно конфигурацията на полянната.
иестожгълникъ има дължина до 6 километра при
широчина отъ четири километра. Въ всѣки слу-

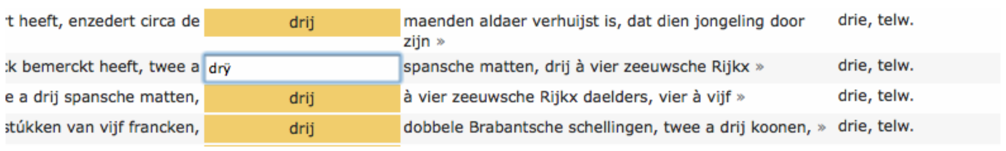
има едни съ на
е квартирувалъ
Гигенъ на Дун
писъ на едина
Nicopolis (днеш
Едцитѣ отъ т
Дунава до Сил
пакъ вторитѣ т
валинитѣ на
материалъ, кои
негърцкитѣ ко

Figure 9 - partial screenshot of CoBaLT – word highlighted in image

Note that the border/highlight can be removed by pressing a key (any key will do). Releasing the key will restore the highlighting again. This can be handy because the highlighting border sometimes covers surrounding characters.

Correcting word forms

By Ctrl clicking on a word form in its context (i.e. in the lower part of the screen) it becomes editable. The user can delete/alter/add characters to it and save the change by hitting the Enter button.
Please note that deleting the word entirely is not supported by CoBaLT yet.



't heeft, enzedert circa de	drij	maenden aldaer verhuijst is, dat dien jongeling door zijn »	drie, telw.
:k bemerckt heeft, twee a	drÿ	spansche matten, drij à vier zeeuwsche Rijkx »	drie, telw.
e a drij spansche matten,	drij	à vier zeeuwsche Rijkx daelders, vier à vijf »	drie, telw.
stúkken van vijf francken,	drij	dobbele Brabantsche schellingen, twee a drij koonen, »	drie, telw.

Figure 10 - partial screenshot of CoBaLT – alter word form

The tool will change the word form in this context. So the corresponding row will not show again for the old word form. Also, the frequency of the old word form will be diminished by one and of course the frequency of the new word form will go up by one.

When spaces are added in the altered word form this will result in the token to be split in several tokens with spaces in between (so the spaces will be added as well). A token can also be split by |'s (pipes). This will split the token without adding a space.

Normally an altered token will keep any analyses it has. This is also the case when a token is split and it belonged to a word form group. However, if the word form is split while it did not belong to a word form group any analyses it had will be lost, as it is impossible to tell which of the new tokens should get the old analyses.

To elaborate on this last case, consider splitting a token like *thebook* where e.g. the OCR engine missed a space between *the* and *book*. It could be that the token had the analysis *book, N* or *the, ART* or even *the, ART & book, N*. None of these analyses would be appropriate for all of the new tokens however, and it is quite unfeasible to automatically determine which analysis belongs to which part.

Please NOTE that this feature should only be used with uttermost caution and consideration.

Usually, it only makes sense to alter a word form when e.g. an OCR error occurred, and you are very, very certain of what the token actually should be.

Overview of keys and mouse clicks

Below is a table that lists key and mouse behaviour in the tool. Please note that if you hold the mouse cursor over any clickable item in the tool a short message will be displayed about its function.

Lower part	Select multiple rows	Click and drag	Moving the mouse cursor over any rows in the lower part of the screen (word forms in their contexts) while holding down the mouse button will result in all these rows being selected. (Please NOTE that if you do this very quickly, some rows might be left out of the selection, so go slow (enough)).
	Add to/subtract from selection	Ctrl-click	Clicking on a row in the lower part of the screen while holding down the Ctrl key will cause this row to be added to/subtracted from any existing selection (while if you don't hold down the Ctrl key you will start making a new selection).
	Select all rows	Double click	Double clicking in the lower part of the screen will result in all rows being selected.
	Add analysis	Click form	Clicking on a word form in context gives the available analyses in a drop-down-menu, or the option to type a new one.
	Delete analysis	Click analysis	Clicking on an analysis in the lower part will result in this analy-sis being deleted for this row and for any other rows selected.
	Toggle validation	F8	Pressing the F8 button will cause any selected rows to be validated/unvalidated.
	Alter word form	Ctrl-click	Clicking on a word form in the lower part of the screen while holding down the Ctrl key will make the word form editable.
	Make word form group	F2/F9	Clicking on a word form in the context of another word form while holding down F2 or F9 will result in a word form group.
	Go to middle part	Ctrl-m	Pressing the 'm' button while holding down the Ctrl key will put the focus on the middle part (so, e.g. the up/down arrows will select the row above/below again).
Middle part	Previous/next word form	↑ ↓	If the focus is on the middle part, the arrow keys will select the previous/next row.
	Add analysis	Click row	Clicking in the left-hand part of a row a in the middle part gives the available analyses in a drop-down-menu, or the option to type a new one.
	Add analysis to forms in lower part	Click analysis	Clicking on an analysis in the right-hand column of the middle part will result in this analysis being assigned to all selected rows in the lower part.
	Delete analysis	Ctrl-click	Clicking on an analysis in the right-hand column of the middle part while holding down the Ctrl key will cause this analysis to be deleted for the word form.
	Toggle validation	Shift-click	Clicking on an analysis in the right-hand column in the middle part while holding down the Shift key will (un)validate it.
Drop-down menu	Browse through menu	↑ ↓	When a drop-down menu is active, the arrow keys allow you to browse through it. Hit 'Enter' to select the highlighted option.

NOTE: pay attention to where you click!

Clicking on a row in the middle or lower part will select this row. However, as described above, clicking on an analysis in the middle part causes it to be applied, and clicking on an analysis in the lower part causes it to be deleted. So, if you just want to select a row, you will usually want to avoid doing so by clicking on any analysis it contains. You would better click anywhere else in the row.

Another thing to pay attention to is that some behaviour can make it appear to the tool as if the Ctrl key is held down when it isn't. This situation is particularly perilous if you click on an analysis in the middle part in order to assign it to some rows below, because it will be deleted in stead!

There is an indicator next to the 'sort by left context' buttons in the middle of the screen that will say whether or not the tool thinks the Ctrl key is held down or not (to see it in action, fire up the tool and press/release the Ctrl-key).

6.7 Import and export

TEI XML

The TEI format used by the scripts below is based on TEI P5[28]. Some partners (the Jožef Stefan Institute in particular) make extensive use of this format .The software described below expects and puts out TEI in this format.

Data import

Import text based lexicon

If information is available about word forms and their lemmata these can be loaded into the MySQL database CoBaLT uses by a command line script.

```
$ ./loadLexicon.pl

 Imports word forms plus lemmata into a Lexicon Tool database.

 ./loadLexicon.pl [OPTIONS] -d DATABASE FILE [FILE2 [FILE3 ...]]

 -d DATABASE
    Database to be imported to.

 OPTIONS:
 -f SEPARATOR
    Separator used to separate columns in the input file (default: tab).
 -h DATABASE HOST
    Host the database is on (default: localhost).
 -l POS
    Default part of speech, in case the input file lacks them (for some
    entries).
 -m SEPARATOR
    Separator used to indicate multiple lemmata analyses in lemma headword, pos,
    and gloss.
```

[28] http://www.tei-c.org/release/doc/tei-p5-doc/en/html/DI.html

```
 -p DATABASE_PASSWORD
    Password for the database user (default: INL setting).
 -u DATABASE_USER
    Database user (default: INL setting).

 Input files are expected to have four columns:
   Column 1: word form
   Column 2: lemma_headword
   Column 3: pos
   Column 4: gloss (is just left empty if not provided).
```

Import TEI XML

Usually, corpora are loaded just by using the tool interface as described above. A special script has been made for importing TEI XML (as used by e.g. the Jožef Stefan Institute in Slovenia). With this script both the documents and the lemma information in it will be loaded to the database in one go.

```
$ ./loadXmlInfo.pl

 ./loadXmlInfo.pl [DB_OPTIONS] -d DATABASE DIR1|FILE1 [DIR2|FILE2 [...]]

 DB_OPTIONS:

 -h HOST
    Database host.
 -u USER
    Database user.
 -p PASSWORD
    Database password.

 File names in absolute paths.

 Directories (also full paths) can also be passed as arguments, in which case
 all XML files in them will be processed (recognized by their extension).
```

Data export

The data the tool uses is stored in a MySQL database. Several ways of exporting are supported.

Export corpus as vertical text

Every file that is loaded in the tool can be exported as vertical text. This means the every word form is printed on a separate line with additional information in tab separated columns.

```
./exportAsVerticalText.pl -c CORPUS -d DATABASE -t TOKEN_DB -o OUTPUT_DIR [OPTIONS]
FILE [FILE2 [FILE3 ...]]
  If the name of the original is abc.xml the output will end up in
  OUTPUT_DIR/abc.exported.xml.
```

```
OPTIONS:

-e
   Print explicitely as utf-8 (somehow this is not always needed).
-n
   Print hader with column names
-h HOSTNAME
   Defaults to localhost.
-u USERNAME
   Defaults to the Lexicon Tool setting.
-p PASSWORD
   Defaults to the Lexicon Tool setting.
```

The columns in the output are:

wordform	The word form as it appears in the left column in the middle screen of CoBaLT.
analysesInCorpus	All single analyses the word has in the corpus (the middle column in the middle part of the CoBaLT interface).
multipleLemmataAnalysesInCorpus	All multiple analyses the word has in the corpus (the middle column in the middle part of the CoBaLT interface).
analysesInDb	All single analyses the word has in the entire database (the right column in the middle part of the CoBaLT interface).
multipleLemmataAnalysesInDb	All multiple analyses the word has in the entire database (the right column in the middle part of the CoBaLT interface).
wordform_group_id	If a word form is part of a group this column has a non NULL value. This value is the same for all word forms belonging to the same group.
dontShowForDocument	If a word is hidden in the tool for this document this column has a non NULL value.
dontShowForCorpus	If a word is hidden in the tool for this corpus this column has a non NULL value.
dontShowAtAll	If a word is hidden for all any document or corpus in the tool this column has a non NULL value.

Export corpus as TEI XML

When a corpus has been imported as TEI (see above) it can also be exported as TEI. The exported files will contain the updated lemma information as edited in CoBaLT. The punctuation and further mark up will be restored form the original XML files.

```
$ ./exportTei.pl

  ./exportTei.pl -d DATABASE -o OUTPUT_DIR [DB_OPTIONS] FILE [FILE2 [FILE3 ...]]

  If the name of the original is abc.xml the output will end up in
  OUTPUT_DIR/abc.exported.xml.

  DB_OPTIONS:
```

```
-h HOSTNAME
   Defaults to the JSI setting.
-u USERNAME
   Defaults to the JSI setting.
-p PASSWORD
   Defaults to the JSI setting.
```

Export lexicon as TEI XML

Lexicon databases can be exported as TEI XML. When a corpus has been imported as TEI (see above) the lexicon can also be exported as TEI. The exported files will contain the lemma information as edited in CoBaLT including the citations. The punctuation and further mark up of the citations will be restored form the original XML files. If the originally uploaded files were not in TEI XML the quotations can be exported as well. In that case the are rebuild from the tokenized files that CoBaLT uses, or they vcan be restored from the database itself.

```
./exportLexiconAsTei.pl -d DATABASE [OPTIONS]

OPTIONS:

-a
   Only export validated analyses.
-b
   Only export validated attestations.
-c
   Only export lemmata that have attestations.
-e
   Encode output as utf-8 explicitely.
-E
   Decode the data from the database (use this if the lemma and part of
   speeched look garbled).
-f
   Cache the content of all the files (don't open and close then all the
   time).
-g
   Get the context of the attestations from the tokenized files (rather
   than from the original XML which is default).
-G
   Use the german way of linking derivations
-h HOSTNAME
   Defaults to 'localhost'.
-l LIMIT
   Specify a limit (e.g. only print the first 100 entries).
-m
   Don't use JSI mapping of part of speeches.
-M
   Try to insert metadata from the documents table into the <bibl> tag
-o FILE
   File to write the output to.
```

```
-p PASSWORD
   Password for the database.
-q
   Use the quote field in the token_attestations table for context
-s CHUNK_SIZE
   For large lexica (databases) it is considerably faster to get the
   lemmata in chunks rather than all in one go. This options allows you to
   set the chunk size.
   Default is 1000.
-u USERNAME
   User name for the database.
```

NOTE that options a/b can have rather subtle effects. Analyses can be validated while none of their attestations are. All attestations of a certain analysis can be validated even while the analysis itself is not. Usually though this isn't the case and the data will be straightforward.

ALSO NOTE that in case of groups of word forms, the analysis of one member can be validated even if the analysis of the other member isn't. This leads to unexpected results when using the -a option. So you might opt not to.

6.8 Licensing

The licensing follows the consortium agreement.
The tool will be made available to the research community according to the regulations of the Dutch HLT agency (TST-Centrale, www.inl.nl), which means that it is freely available for non-commercial use.

The LMU Lextractor Tool

In IMPACT, several partners[29] have used the LMU Lextractor tool for corpus-based lexicon building. Although this tool is currently not an IMPACT deliverable, we include a description of its main features.

The Lextractor Tool supports the collaborative and web-based construction of lexica from given electronic corpora containing proofread texts in authentic historical spelling. The idea is to store in the lexicon for each word form (type) *w* occurring in the corpus the possible correct ``interpretations''. An interpretation in our sense specifies a modern word wmod corresponding to *w* as well as the part-of-speech and the lemma of wmod It also includes a description of the patterns that are used to derive *w* from *wmod* if there exists such a pattern based derivation. Furthermore, for each interpretation, witnesses in the background corpus are stored. A witness is a particular occurrence of *w* where the given interpretation is correct.

The general philosophy of the tool is to support lexicographers in the sense that - whenever possible - meaningful interpretations for word forms *w* from the historical corpus are fully suggested by the system. The manual work of the lexicographers is then reduced to two steps, (i) accepting or rejecting suggested interpretations, and (ii) selecting witnesses for confirmed interpretations using a list of concordances for w shown in the graphical user interface. As a matter of fact, the historical corpus will often contain particular word forms *w* where the system cannot offer plausible interpretations. In this case

[29] LMU, Bulgarian Academy of Sciences, Jožef Stefan Institute.

it is the task of the lexicographers to give the necessary interpretation(s). Such an interpretation may specify a new pattern yet not ``known'' to the background mechanisms (s.b.).

If a new pattern has been used several times in distinct interpretations, the team may decide to add it to the ``official'' list of patterns used by the background matcher.

In this sense the actual work with the Lextractor Tool contributes to building up an adequate set of patterns for historical spelling variation in a particular language.

The active role assigned to the system explains that the Lextractor Tool needs specific language resources that are then built-in before the suggestion mechanisms can work in a satisfactory way. As a first prerequisite, a lexicon of the modern words of the given language is needed. The second ingredient is a collection of patterns that explain the typical differences between modern and old spelling in the language. Given these resources, a special built-in tool automatically computes possible interpretations for word forms occurring in the historical corpus and displays all interpretations computed for a word form *w* to the user.

Part II: Building 'NE lexica', i.e. lexica of Named Entities (D-EE2.3, D-EE2.4 and D-EE2.5)

I Procedure

1. Introduction

Named entities (NE) are specific words or word groups that refer to a single particular item in the real world. For example, *London* is a named entity, referring to a particular city in Great Britain, and so is *Barack Obama*. For IMPACT, the creation of a named entity lexicon is an important addition to the general lexicon. Named entities show a different behaviour as to variation, hence the importance of an attested NE lexicon of the language period dealt with so as to get a good view on this aspect. The NE lexicon can further be used in retrieval to help searching for variants of names. In the library community, a lot of attention is paid to named entities, since they are frequently searched for by users in the digital collections. Apart from that, the Named Entities get special attention by their description in the Authority Files for different languages. That is why in IMPACT, the NE lexica are linked to the authority files by adding the persistent identifier.

In this chapter, we discuss the necessary steps in building a named entity lexicon, and discuss tools that are used in the process.

2. Steps in the creation of a NE-Lexicon

There are three main steps when building a lexicon of named entities, and these steps will be discussed in the sections below. These steps are:

1) data collection
2) data enrichment
3) NE-lexicon building

2.1 Data collection

As a first step in the process, a large, diverse set of named entities should be collected. Named entities come in different flavors (locations, dates, numbers, etc.), but we have decided to limit ourselves to the three main types:

• Locations (LOC)	London, Roman Empire, Nile, Alps
• Persons (PER)	Lionel Messi, Cleopatra, Noach, Zeus
• Organizations (ORG)	New York Times, House of Commons, Procter & Gamble

Since Named Entities have a cross language character, and a lot of effort is needed for collecting data, especially when wanting to link to the authority files, it was decided to create a central database (NE repository) to make the collected material easily accessible and to avoid duplication of effort in collecting and storing data. In chapter II, 1 of this part of the cookbook, a technical description is given of the conversion tool that is integrated into the repository responsible for converting different resources into one XML-format.

An IMPACT NE lexicon contains Named Entities attested in the language period the NE lexicon is used for. For the collection of these types of named entities, the Ground Truth material from IMPACT can be used (cf. Part I, chapter II). In addition to the ground truth text collections, named entities can also be collected from historical documents that specifically contain lists of persons, locations or organizations. Examples are biographical works on historical figures, university almanacs and registers of birth, marriage, death or commerce. For our collection of Dutch named entities, we have used the text collections that were also used for general lexicon building, as well as the following keyed NE-lists:

Name	*NE-type*	*Description*
Adresboek Amsterdam	LOC	Amsterdam street names
Batenburg	LOC	Towns in the Netherlands, Belgium and Luxemburg
Gosselin	LOC	Towns in the Netherlands, Belgium and Luxemburg
Album Groningen	PER	Groningen University staff
Album Utrecht	PER	Utrecht University staff
Boek Amsterdam	PER, ORG	Amsterdam business register
Godgeleerdheid	PER	Historical religious persons
Leidsche Almanak	PER,LOC	Leiden University students and their addresses
Letterkunde	PER	Historical literary persons
Levensberichten Zeeuwen	PER	Zeeland inhabitants
Molhuysen	PER	Biographies of historical Dutch people
Naamlyst 's Gravenhage	PER	Birth, marriage and death register of Den Haag
Naamlijst Kolonien	PER	Register of inhabitants of former Dutch colonies
Naamregister Rotterdam	PER,ORG	Rotterdam business register
Universiteit Amsterdam	PER	Amsterdam University staff
Wie is dat	PER	Biographies of historical Dutch people
Staatsalmanak	ORG	Chamber of commerce listing

While these keyed NE-lists provide ‘instant’ named entities, this is not the case for the named entities that are present in the ground truth text collections, because they are not marked. A crucial step is therefore to distillate the named entities and their types from these texts. This step will be discussed next.

Tagging named entities in text

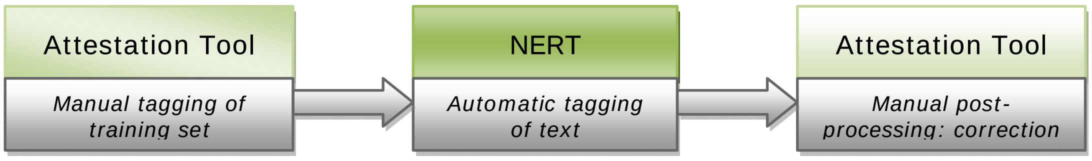
Below, a fragment from a 19th century text by the Dutch writer Willem Bilderdijk is shown:

> *De oorlog die nu ontstond tegen Rome is bekend, zoo wel als die tegen Tutor en Sabinus door Vespaziaan zelv' geëindigd wierd*.

In this fragment, *Rome* is a NE-location, and *Tutor, Sabinus* en *Vespaziaan* are NE-persons, and thus, we would like to have the text enriched with this information, e.g.:

> *De oorlog die nu ontstond tegen <NE type=”LOC”>Rome</NE> is bekend, zoo wel als die tegen <NE type=”PER”>Tutor</NE> en <NE type=”PER”>Sabinus</NE> door <NE type=”PER”>Vespaziaan</NE> zelv' geëindigd wierd*.

In a three-step process, we use two IMPACT tools for this job: the Named Entity Recognizing Tool (NERT) and the Attestation Tool:


Attestation Tool
Manual tagging of training set
NERT
Automatic tagging of text
Attestation Tool
Manual post-processing: correction

The NERT automatically tags NE's in a text, but it is necessary to feed the tool with sufficient training material. So, for example, if we have a collection of 19th century newspapers from which we wish to extract NE's, it is necessary to manually tag the NE's in a small part, which is then used as training material to automatically tag the remaining part. This manual tagging is done in the IMPACT NE Attestation Tool. Then, if the NERT has tagged the NE's in the remainder of the text, the output needs to be manually checked for errors, again in the IMPACT NE AttestationTool (section II.2).

We do not discuss the initial tagging process of the training material here, but instead refer to Part II, chapter II,2. Note that, before getting started, it is very important to be familiarized with the named entity tagging rules, in order to create a correct and consequent data set. These rules are defined in the *Named Entity Recognition Task Definition (NERTD[30])*, with an extension for specific Dutch cases in the document *Tagging NEs - Applying NERTD to historical Dutch* (Appendix II)

The performance of the NERT depends strongly on two factors: the amount of training material and the degree of similarity between the training material and the actual material that needs to be tagged. 'Similarity' is a broad term, which can refer to any kind of overlap in text style, genre or time period. The amount of training material that is needed for decent performance of the tool depends strongly on the type of texts that is used. Generally, the more homogeneous a collection of texts, the less training material is needed. The Dutch collection of parliamentary proceedings (*Staten Generaal*) is an example of a very homogeneous text type, whereas the *DBNL*, a collection of prose, poetry, non-fiction and fiction from different writers, is an example of a very heterogeneous text type. It makes sense that for the latter, more training material is needed to achieve a similar amount of coverage than is the case for the former. In the table below, some performance scores are shown for training sets of different sizes for various Dutch data.

Text type	Training set size (# words)	Recall	Precision	F1
DBNL (19th c. fiction)	10166	22,77	79,06	35,36
	50081	22,55	70,05	34,12
	100973	24,01	78,45	36,77
	178834	36,08	80,2	49,77
Staten Generaal (19th c. parliamentary proceedings)	10074	42,51	85,3	56,74
	50052	67,43	88,56	76,56
	100079	72,25	87,62	79,2
	179030	77,8	89,07	83,06

[30] Nancy Chinchor, Erica Brown, Lisa Ferro and Patty Robinson, *1999 Named Entity Recognition Task Definition*, MITRE, 1999. http://www.nist.gov/speech/tests/ie-er/er_99/doc/ne99_taskdef_v1_4.pdf

There are two ways of improving the tool's performance given a training set of a particular size. Firstly, a gazetteer list, a list of known NE's, can be added. Second, we found that the way the training set is constructed matters as well. A training set that consists of randomly selected sentences from the total set leads to a better performance than, for example, simply using the first 100,000 words. This makes sense, since the former will give a better coverage, especially with heterogeneous text collections. For a complete evaluation of the tool with different datasets and varying training sets, the reader is referred to the technical documentation in Part II, chapter II,3.

2.2 Data enrichment

After the creation of a dataset of named entities, the next step is enrichment. It is desirable to add the following information to the NE's:

- Locations: modern lemma

Haerlem	*modLem=Haerlem*
Fransch Guiana	*modLem=Frans-Guyana*

- Person names: name part information, variants, persistent id's from authority files (if applicable)

Mahatma Gandhi	*given name=Mahatma*
	surname=Gandhi
	PND id=118639145
Elisabeth Aalberts	*given name=Elisabeth*
	variants=Elizabeth, Elisabet, Elijzabet
	surname=Aalberts
	variant=Aalbers

- Organizations: modern lemma

Nederlandsche Suyker Maetschappy	*modLem=Nederlandse Suiker Maatschappij*

Note that the treatment of person names differs from that of locations and organizations: we group person variant names, but without assigning a single modern lemma to them.

The enrichment process can be split up in a series of steps. For person names, a first step is to tag the separate entities of each full name, which we have done with a simple Perl script that does a first attempt in identifying parts, after which the tags are manually checked and corrected. We identify the following name parts:

given name	*Jan, John, Lizzy*
surname	*Vries, Lippe Biesterfeld*
title	*baron, hertog, duke, sir, dr.*
particle	*de, van, der, à*
suffix	*jr, sr, junior, senior*

A second step is to pair NE's from the source data to applicable external identifiers from authority files. This way, entries in the lexicon are linked to their counterparts in other catalogs. For the Dutch NE-lexicon, we matched all person names to the

Personennamendatei (PND) from the German National Library and added found PND-id's as external-id's to the respective lemma.

Note the status of these links. First of all, the person names in the lexicon only have a 'formal' status. That is, there is no distinction in the lexicon between the French ruler Napoleon Bonaparte and any accidental namesake. This means that if we add a link to the PND-id of Napoleon Bonaparte, we add one to all occurrences of Napoleon Bonaparte in our source data, regardless of him being the actual French ruler. Similarly, be aware that the lexicon could contain many names such as, in the case of Dutch, 'Abraham', 'Johannes' or 'Karel' of which it is not clear whether they refer to generally known persons (e.g. from the bible or from politics) or just accidental first names from e.g. fictitious characters.

For Dutch, we linked only full person names (with either both a first and last name or a name followed by any kind of suffix, e.g. 'Karel de Grote' or 'Hendrik VIII') to the entries in the PND. If this lead to multiple possible external id's, we limited this number to 10. The status of these links should be regarded as 'for this lemma, there is a possible link with this entry/these entries in the PND'.

The third step for person names is to link each name (part) to its variants, which is a perfect job for the NE-Matcher module that comes with the NERT-package. The Matcher is a tool that identifies matching NE's in a list. For each NE, it returns a set of variants, each with a matching score, which is an indication of the similarity of the two NE's. The tool is discussed in more detail below. As regards the lemmatization of locations and organizations, the Matcher module can be used as well. For this type of job, it is fed both a list of historical NE's and a list of modern NE's, and the tool matches NE's in the former against those in the latter.

Regardless the type of NE, the final step after using the Matcher is to manually verify the output. For the lemmatized locations and organizations, the IMPACT Lexicon Tool is used.

The NE-Matcher

The Matcher is a command-line java tool that is part of the NERT-package. In this section, its general use will be discussed. For a more detailed account on running the tool and customizing its settings, the reader is referred to the NERT 's technical documentation, Part II, chapter II,3.

The Matcher compares NE's in the following way: each NE is converted into a phonetic transcription and this transcription is broken up into s-grams of varying length. For each pair of NE's, the Matcher calculates the number of shared s-grams, with a correction for string length differences. This leads to a matching score between 0 – 100 for each pair, with 100 indicating a perfect match as far as the Matcher is concerned (note that this does not mean the two NE's are identical, since the comparison is done with the two phonetic forms).

The conversion from a string to a phonetic transcription is done with a set of java regular expressions that can be customized for each language and each task. For example, we found that our patterns that yielded good results for matching historical Dutch locations against a modern lemma led to a high amount of incorrect matches for person names. Below, some examples of patterns are shown that were used for the matching of Dutch locations:

\W=>	# remove all non-word characters
(\w)\1=>$1	# remove all double characters
eij|ij|ei|y|ey=>Y	# convert any /eij,ij,ei,y,ey/ into /Y/
(u|a|o)e=>$1	# convert any /ue,ae,oe/ into /e/
ch=>g	# convert any /ch/ into /g/

As mentioned above, the Matcher can either compare NE's within a single list or compare NE from one list against those in another. The former groups variants, the latter is a sort of lemmatizing. The Matcher comes with several output settings, and a typical output of the matcher looks like this:

NE	match (matching score)
Franciscus	Franciskus (100), Fransiskus (100), Franziscus (100), Françiskus (100), Franssiscus (100), Francicus (83), Françiscus (80), Franscisca (76), Francisco (70), Franciska (70), Françisca (70)

In this example, scores are shown with a minimum of 70, but this value can be adjusted. The lower the score, the higher the probability of a match being incorrect. Note, however, that using a higher threshold might lead to a 'cleaner' output (with fewer false positives), but will increase the risk of having false negatives, that is, correct matches that are not shown. Because a manual correction round is always necessary, a relatively low threshold, i.e. 70, is recommended, because it is easier to remove false positives than to add the missing false negatives.

For lemmatization, e.g. of locations, it is necessary to create a list of modern lemmata before the actual matching can take place. There are a few simple rules of thumb for this:

use trial and error: take a look at your data and try to establish the kinds of modern lemmata needed. E.g., Dutch newspapers contain a lot of Dutch place names but also many general European (London, Rome, Waterloo) and worldwide (Suez, Japan, Alaska) locations. Do a test matching and see if there are 'gaps' that can be filled (e.g. Dutch newspapers from the 1850s-1950s contain a lot of Indonesian locations).

don't overdo it: it might seem easiest to dump all the universe's locations into the matcher, but this is actually counterproductive. Firstly, it seriously slows down the matcher's performance, and secondly, it will most probably contaminate the results with many useless matches. So think first before you decide to add that list of Kalmukkian town names.

Where to get your location lists from? For Dutch, we used, among others, the following sources:

- lists with Dutch and foreign locations with their official spelling
- USA Geonames[31] (GNS): a site with very complete lists of locations per country
- Wikipedia (e.g. for lists of continents, rivers, oceans, U.S. states, German federal states)

31 http://earth-info.nga.mil/gns/html/country_files.html

Needless to say, it is important to double check the locations from any of these sources. For example, both the list with official spellings and the GNS contain many variant spellings (*Zeeuwsch Vlaanderen-Zeeuws Vlaanderen, Caïro-Kairo*, *Congo-Kongo*), and a list of biblical locations from Wikipedia used different spelling than that from the official Dutch Bible Society. Let us have a look at a typical output for lemmatizing locations:

NE	match (score)
Alkmaar	Alkmaar (100)
Harderwyck	Harderwijk (100), Harderdijk (77)
Babel	Kabel (77), Bakel (73), Bavel (73)
Egyptenland	
Kuilenburg	Zuilenburg (76)

Of these examples, *Alkmaar* and *Harderwyck* are matched with a correct lemma (*Alkmaar*, *Harderwijk*). *Harderwyck* is also given the incorrect match *Harderdijk*. The biblical *Babel* was not in our list of modern lemmata, and the Matcher did not find a correct match. The same happened with *Egyptenland*. *Kuilenburg* is an old spelling variant of the Dutch town *Culemborg* which was in the list of modern lemmata, but got a matching score lower than 70 and is therefore missing from the results. The example of *Egyptenland* is worth noting, because we found that for the lemmatizing of an 18th century book on history for children and a collection of fiction from different writers, many of such locations popped up for which a present day variant does not exist. Other examples are *Oostersche Keizerryk* ('Eastern empire') and *Batavia* (the former Dutch colonial name for *Jakarta*).

Post-correction

After having used the NE-Matcher, a final round of manually inspecting its output is required. For the lemmatization of locations, the IMPACT Lexicon Tool is used (cf. Part I, chapter VI,6). This round of post-correction should not be underestimated in time and effort. While the Matcher will identify many of the most common locations (*Amsterdam*, *Berlin,* etc.), many of the more difficult ones will need revision, either because multiple matches have been given and without detailed inspection it is unclear which one is correct, or simply because no match was given. An example of the former is that historical *America* is matched against *Amerika* and *America*. The first match is correct if the attestation refers to the continent, the latter if a Dutch town is meant. Also, we distinguish *London*, with modern lemma *London*, from *Londen*, with modern lemma *Londen*: the first is the English name for the city, the second the Dutch, and both get their own lemma. Often, closer inspection of the context is necessary to decide whether you are dealing with a location in a foreign language or a spelling variant in your own. Finally, we advise to have a clear, unambiguous set of rules for this post-correction process, especially when the job is done by multiple people. For example, when it is necessary to try and find a missing modern lemma, it is recommended to have a hierarchy of sources, since each can have a different spelling. This means that first, a dictionary is consulted, and if the lemma is missing in that source, the GNS is used, and so on.

2.3 NE-Lexicon building

The NE-Lexicon is more than just a bare collection of NE's: it contains information about attestations, wordforms, lemmata, name structure and variation. The structure of the NE-Lexicon is the same as that of the general lexicon, and we therefore refer to the Lexicon Structure document (D 2.1) for details on adding the NE's to the database.

II NE Lexicon building and deployment tools (D-EE 2.4): Technical documentation

1. Named Entity Repository Converter

1.1 Partner

INL

1.2 Deliverable

part of D-EE2.4

1.3 Background

The IMPACT Named Entity Repository is an online repository used to store named entities gazetteers. Users can upload lists in various formats. The software that loads the data into the database however only knows one format, which is the IMPACT Interchange format. The Named Entity Repository Converter converts data to this Interchange format. It supports several formats, including MARC21, MAB and CERL.

1.4 Requirements

The Converter is a Perl program. It was tested on Perl version 5.8.8, running on Red Hat Enterprise Linux Server release 5.2. NOTE that for conversion of XML files (like e.g. MAB) XML::LibXML needs to be installed.

1.5 The Converter program

The Named Entity Repository Converter is a program called convert2interchange.pl. It is an object oriented program that allows for different formats to be plugged in. A new package might be writen for a new format and can be plugged in as such. The Converter software will not have to be rebuild for it.

Usage

```
$ convert2interchange.pl [OPTIONS] FILE
```

Example

```
$ perl convert2interchange.pl -f TAB -d
"name=1,variants=2,extId=3,bConvertToUtf8=0,bPrintOriginalRecord=1"
example.tab > example_interchange.xml
```

For further examples cf. the README.TXT file in the Examples/ directory that comes with the distribution.

Options:

- [-c CONF_FILE|-d CONF_STRING]
  CONF_FILE should just be a filename.
  CONF_STRING is a string describing the options.
  The file shoud have one option per line, as a *name=value* pair. E.g:
  extId=1
  name=2

The CONF_STRING is a comma separated string stating the options.
E.g: -d "name=2,extId=1"

- If neither options are provided the default options are chosen based on the format (see the documentation of the formats for the options).

- -f FORMAT
  Obligatory.
  Different formats are:
  - CERL for line oriented CERL data.
  - MAB for Person Namen Datei data from the Deutsche Nationalbibliothek
  - MARC21 for e.g. for Person Name Datei in MARC 21 format.
  - TAB for tab separated files (though a different separator can be used).
  - XML for arbitrary XML files.

It is possible to add a format of your own. See the Converter documentation.

- -h Print help information and exit.
- -o FILE Print output to file (if not provided, print to standard output).
- -t Print time information (how long it took).

1.6 The packages

As noted above the software is object oriented. There is one abstract Converter object that all the other formats inherit from. In this way it is possible to implement a converter for a new format without having to alter any other code.

Converter.pm

This package implements a class that other formats can inherit from.
It has methods for reading configuration string and files and printing interchange XML.

Options

For any (object derived from a) Converter object you can set:

- bPrintOriginalRecord
  If this options is set, the original record from the input file is written to the output as well (enclosed in a <![CDATA[...]]> tag).
  Default: 1 (true)
- bConvertToUtf8
  If this options evaluates to a true value, the output string is converted to utf8. This is only needed when the input isn't in utf8 already.
  Default: 0 (false)

- sGlobalType

  Sets the type for every record in the entire file. It is overruled however by any record that has its own type specified.

  Default: *not defined*
- iMaxNrOfNoSenseLines

  The converter quites if the number of lines read exceeds this figure while no record was recognised yet (e.g. because the format of the input was not right).

  At the moment this works for MAB and MARC data.

  Default: 1000

Methods

The methods that (object derived from) Converter objects can use are:

readConfigurationString

```
$oConverter->readConfigurationString($sConfigurationString);
```

Sets all the options in $oConverter->{hrConf} according to the configuration string.

readConfigurationFile

```
$oConverter->readConfigurationString($sConfigurationFile);
```

Sets all the options in $oConverter->{hrConf} according to the configuration file.

setDefaultConfiguration

```
$oConverter->setDefaultConfiguration();
```

Gives the general options (see above) in $oConverter->{hrConf} their default values.

printNameRecord

```
$oConverter->printNameRecord()
```

Prints the contents of $oConverter->{hrNameRecord} as an XML record in interchange format.

printHeader

```
$oConverter->printHeader()
```

Prints the XML header for the file in interchange format.

printFooter

```
$oConverter->printFooter()
```

Prints the XML footer for the file in interchange format.

Implementing your own converter

A converter for an additional format can be made by implementing a class Formats::myNewFormat that inherits from this Converter class. The only method it needs to have is convert() (though more can be added of course).

The convert() method is called with one argument, which is a file handle that is opened for reading. The convert method typically would fill the $oConverter->{hrNameRecord} hash and call $oConverter->printNameRecord().

CERL.pm

This file implements an CERL object. It inherits from the Converter object.
It is important to note that the line oriented CERL format is supported. *NOT* CERL XML.

Options

The CERL specific options and their default values are:

- extId
  Number of the field that holds the identifier.
  The default is 001.
  NOTE that CERL has encodes type information in its identifier field (001).
  The field starts with three characters:
    - cnp
      Personal name
      Related fields: 200, 400.
    - cni
      Imprint name
      Often, but not always, a personal name, see below.
      Related fields: 210, 410.
    - cnc
      Corporate body name
      Related fields: 212, 412. Neglected, see below.
    - cnl
      Imprint place
      Related fields: 215, 415.
    - caf
      Source of reference
      Work titles. Not relevant for conversion and hence neglected.

      See below for further details.

- personName
  Number of the field holding the names.
  Defaults: 200, 210

  Since CERL doesn't make a clear distinction between fields containing person names and corporate bodies (e.g. with the cni records) we try to deduce the type from the data. In the 210 case we can only know for sure that we are dealing with a person name if there is a $b field. If there is only a $a field it can be a person name but also a firm's

name.
We neglect the latter case.
So, please NOTE that no company names are extracted since it is impossible to distinguish them from person names in the data.

- orgName
  As noted above 212 and 412 hold corporate names. Unfortunately however, place names also occur here and there is no way to separate these from the corporate names.
  Because of this these fields are neglected.

- geoName
  default: 215
- variants
  default: 400, 410, 415

  400 holds variants for 200 fields, 410 for 210 fields and 415 for 215 fields. NOTE that the same goes for 410 fields as goes for 210 fields (see above) so no corpate names are extracted.
- type
  Implicit, see above.

MAB.pm

This file implements an MAB object. It inherits from the Converter object.
It is essential that XML::LibXML is installed.

NOTE that when converting PND or GKD MAB XML you need not specify a configurations file with the -c option. The option can simply be left out, and provided the program is run form the directory the Configurations/ folder is in, it will choose the right configuration itself based on the content.

Options

The MAB specific options and their default values are:

- extId
  Number of the feld in the Personnamen Datei that holds the identifier.
  The default is 001.
- name
  Number of the feld in the Personnamen Datei holding the name.
  The default is 800.
- variants
  Regular expression describing the fields containing variants.
  NOTE that field 801 and 811 or only taken into account if their indicator attribute is set to "b", and 810 only if its indicator field is " " (a blank).

NOTE that if the variant field is 820 or 830 and it is a comment field (indicator "v") containing the word 'falsch' (e.g. 'falsche Schreibweise'), the variant before is neglected.
Default: not defined.

- type
  Number of the field describing the type.
  NOTE that this option is used for the GKD. For the PND it is easier to set a global type (cf. Converter documentation).
  Default: not defined.
- reOrgType
  Regular expression decribing what the indicator attribute should look like for an organisation.
  Default: not defined.
- rePersonType
  Regular expression decribing what the indicator attribute should look like for a person name.
  Default: not defined.
- rePlaceType
  Regular expression decribing what the indicator attribute should look like for a geographical name.
  Default: not defined.
- NOTE that if none of the three expressions above is defined the value found in the type field is copied as is.

For general options, read the Converter documentation.
NOTE that bConvertToUtf8 is by default set to a true value for MAB conversion.

MARC21.pm

This file implements an MARC21 object. It inherits from the Converter object.
It is essential that XML::LibXML is installed.

Options

The MARC 21 specific options and their default values are:

- extId
  Number of the *datafield* that holds the identifier.
  The default is 010 (with code="a").
- personName, orgName, geoName
  Number of the *datafield* in holding person names, organisation names or geographical names respectively.
  The assumption is that either of these three fields is there and not more than one them.
  Only subfields with code="a" are taken into account.
  Peculiarities:
  Person records can also have a subfield with code="f" indicating a period.
  Person records can have a subfield code="b" or code="c" which will get appended to the name in the code="a" subfield. Person records can have subfields with code="q" holding a fuller form of the name which is treated as a variant.

Organisation fields can have sevaral sub fields code="b" next to the code="a" subfields. In these cases, only the last subfield code="b" is taken into account.
Defaults:
personName: 100
orgName: 110
geoName: 151

- variants
  See the bit on *personName* above.
  Also by default the fields 400 and 500 (for personal names), 410 (only ind1="2") and 510 (for organisations) and 451 and 551 (for locations) are treated as variants.
- period
  See the bit on *personName* above.
- type
  Implicit, since we have *personName*, *orgName*, *geoName*.

For general options, read the Converter documentation.
NOTE that bConvertToUtf8 is by default set to a true value for MARC21 conversion.

TAB.pm

This file implements a TAB object. It inherits from the Converter object.
This format can be useful if someone has a simple list of names that has to be uploaded. The list can just be uploaded, or additional information can be added using the several options.

Options

The TAB-specific options and their default values are:

- extId
  Number of the column in the input file that holds the identifier.
  The default is 1.
- name
  Number of the column in the input file that holds the name.
  The default is 2.
- variants
  Number of the column in the input file that holds the variants of the name. All variants get their own record in the interchange XML, with the same external identifier, if there is an extId column.
  The default is 3.
- type
  Number of the column stating the type of the name.
  Default is 4.

- language
  Number of the column stating the language.
  Default: 5.
- periodStart
  Number of the column stating the start of the period.
  Default: 6.
- periodEnd
  Number of the column stating the start of the period.
  Default: undefined.
- bSkipFirstLine
  Boolean value indicating whether or not the first line should be skipped (e.g. because it contains column headers).
  Default: 0 (false).
- sSeparator
  The (regular expression) pattern describing the sequence of characters separating the different columns in the input file.
  Default: "\t" (a tab)
- sVariantSeparator
  The (regular expression) pattern describing the sequence of characters separating the different variants in the variant column.
  Default: ", *" (a comma followed by any number of spaces)
- reOrgType
  Regular expression decribing what the type field should look like for an organisation.
  Default: not defined.
- rePersonType
  Regular expression decribing what the type field should look like for a person name.
  Default: not defined.
- rePlaceType
  Regular expression decribing what the type field should look like for a geographical name.
  Default: not defined.
- NOTE that if none of the three expressions above is defined the value found in the type field is copied as is.

For general options, read the Converter documentation.

XML.pm

This file implements an XML object. It inherits from the Converter object. It is essential that XML::LibXML is installed.
The idea behind this format is the same as for the TAB format. It can be useful if someone has a database or a list of names that need to bee uploaded. This format allows for several options, described below.

In XML a value can be set as an attribute or as a text enclosed in tags.

```
<myName id="1" kind="singer">John</myName>
<myName id="2" kind="drummer">Ringo</myName>
```

In the example above, the value for the name is given as content text in between the myName tags, while the id is set as an attribute.

With this package it is possible to extract both types of information even while filtering on a certain attribute value (e.g. only in cases when kind is set to singer). This is also possible in cases like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<myRoot>
 <myRecord id="1">
  <myName kind="singer">John</myName>
 </myRecord>
 <myRecord id="2">
  <myName kind="drummer">Ringo</myName>
 <myRecord>
</myRoot>
```

where a piece of information (in this case the id) is set as an attribute of the enclosing tag rather than of the tag itself.
In order to achieve this, set the value for the tag name (in this case extId) to '<context>'.

Options

The XML-specific options and their default values are:

- extId
  The name for the XML tag containing an identifier.
  The default is set to '<context>' (see above for explanation).
- name
  The name for the XML tag that holds the name.
  The default is 'name'.
- variants
  Name of the XML tag containing variants.
  Default: *not defined*.
- type
  Name of the tag stating the type of the name.
  Default is 'type'.
- language
  Name of the XML tag holding the language.
  Default: *not defined*.

- period
  Number of the XML tag stating the period.
  Default: *not defined*.
- *x*AttrName
  For *x* being any of the options above an attribute name can be given (so e.g. 'periodAttrName'). If this value is set and xAttrValue is not set the value of this attribute is taken as the value for this item.
- *x*AttrValue
  Only makes sense when *x*AttrName is set as well.
  When both *x*AttrName and xAttrValue are set they are taken as a filter for item *x*. Only tags that have the attribute set to this value are taken into account.
- context
  XPath of the records in the XML file that we are interested in.
  E.g. 'myRecord' in the example above.
  Default 'names'.
- reOrgType
  Regular expression decribing what the type field should look like for an organisation.
  Default: not defined.
- rePersonType
  Regular expression decribing what the type field should look like for a person name.
  Default: not defined.
- rePlaceType
  Regular expression decribing what the type field should look like for a geographical name.
  Default: not defined.
- NOTE that if none of the three expressions above is defined the value found in the type field is copied as is.

1.7 License and IPR protection

The licensing follows the consortium agreement.
The tool is integrated in the NE repository but will also be made available separately to the research community according to the regulations of the Dutch HLT agency (TST-Centrale, www.inl.nl), which means that it is freely available for non-commercial use.

2 IMPACT NE Attestation Tool

2.1 Partner

INL

2.2 Deliverable

part of D-EE2.4

2.3 Background

This tool is meant to be used to for manual evaluation and correction of automatically matched occurrences of Named Entities in text material. This functionality is used to build Gold Standard Corpora of Named Entities.

For the processing of the tagged NE's the following general design principles were formulated.

1. Multiple concurrent sessions (by different staff members) should be possible. In order to meet deadlines it is often necessary to have more than one staff member working on evaluating the data. The tool should allow several users to access the database and deliver their input.
2. The verification tool should be in the form of a web application that can be run from any computer in the local network.
3. Input actions, especially the frequent ones, should be from the keyboard, since this allows for faster responses than clicking the mouse on screen buttons.
4. Information should be presented such that quick evaluation is possible.

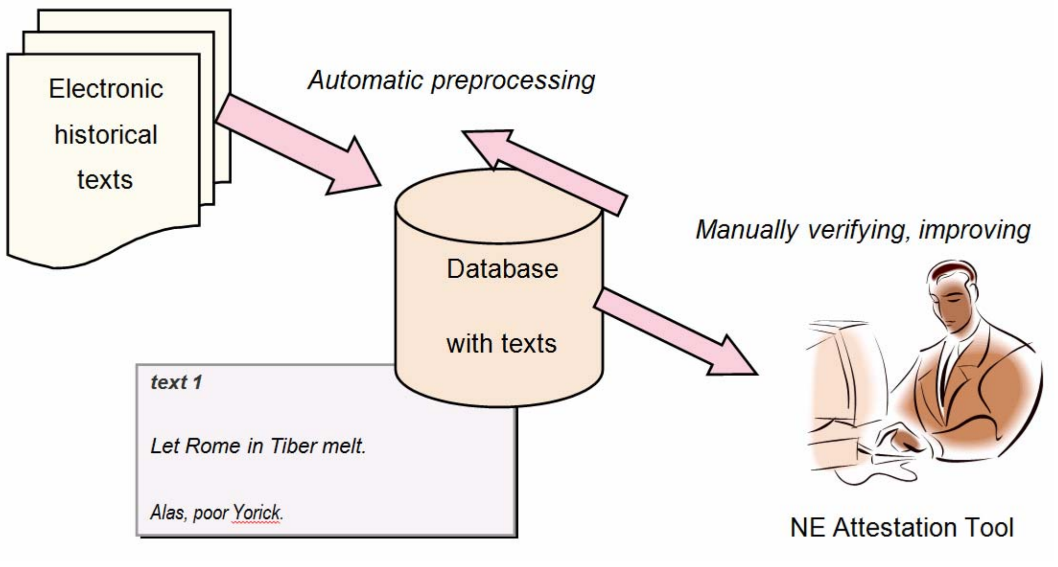


Figure 1

2.4 Features and system requirements

The Attestation Tool is based on a LAMP[32] architecture. Users need a web browser. We tested the user interface on: Firefox 3.0.5, and some earlier versions, on Linux, Mac OS X 10.4 and Windows XP; Safari 3.1.2 on Mac OS 10.4; Internet Explorer on Windows XP. The web server needs to have MySQL (The tool was tested on MySQL 5.0.27) and PHP installed. The

[32] The acronym LAMP refers to a solution stack of software, usually free and open source software, used to run dynamic Web sites or servers. The original expansion is as follows:

- Linux, referring to the operating system;
- Apache, the Web server;
- MySQL, the database management system (or database server);
- PHP or others, i.e., Perl, Python, the programming languages.

http://en.wikipedia.org/wiki/LAMP_(software_bundle) server side was tested with PHP version 5.2.0, Apache 1.3 on Mac OS X 10.4, and Apache 2.0 with PHP 5.1.6 on Red Hat Enterprise Linux 5.

The interface consists of just one page: attestationTool.php. It is a so-called rich Internet application which means that it uses AJAX to communicate with the database server and display the results.

2.5 Configuration

The tool for manual annotating the attestations requires some minor adaptions during installation. Mainly the addresses for services and the names of databases, database users and passwords have to be adapted in the PHP scripts. Also, some settings must be added to the Apache configuration file, usually called “httpd.conf”.

2.5.1 PHP/HTML

For the tool to run, two PHP files need a small adjustment.
The file attestationTool.php contains a form called loginForm that lists all available databases. Change these to your own.
The file php/attestationToolBox.php has the database parameters at the top (host, user name and password). Change these to match your own.

2.5.2 MySQL database

2.5.2.1 Filling the database

The database needs to be filled with lemma’s, quotations, users (revisors) and types.
Attestations are added by the user using the tool, though they may be preloaded as well, in which case the user can use the tool to check/improve the preloaded data.

2.5.3.2 Lemma’s table

The lemma field actually is the name of the text. It is called lemma for historical reason, and the name remains for backwards compatibility. In case the text has an external identifier that means something outside this database this numer or string can be stored in the externalLemmaId field. The id field itself is for MySQL internal use. It is an auto increment field that is best left to the MySQL server to determine a value for.

The revisorid and revisionDate columns are filled by the tool (see below).
(The initial variants column was used for a WNT-purpose and may be (ab)used for any other purpose as deemed appropriate. The tool doesn't use it).

2.5.2.3 Quotations table

The quotation field should simply contain the text at hand (though it might in fact be empty, as the tool only works on the tokenizedQuotation). The tokenizedQuotation should be filled with newline separated tuples describing tokens in the text. It should look look like this:

```
onset1<TAB>offset1<TAB>canonical wordform1<TAB>wordform1
onset2<TAB>offset2<TAB>canonical wordform2<TAB>wordform2
etc...
```

While the quotations table may, strictly speaking, be empty, the idea is that the onset and offset in the tokenizedQuotation table refer to the onset and offset in the quotations field. This makes retrieval later on easier (if not possible at all).

NOTE that the attestations table states the same onset and offset again. While this is formally redundant it does make retrieval later on a lot easier.
(The quotation section column was used for a WNT-purpose and may be (ab)used for any other purpose as deemed appropriate. The tool doesn't use it).

2.5.2.4 Attestations table

As noted above the onset and offset in the attestation table are exactly the same as the ones in the tokenizedQuotation field of the quotations table. The reliability column only makes sense with preloaded data. Different levels of reliability can be given to texts. The reliability comes in 5 levels where 0 is (maybe somewhat counter-intuitively) most reliable, anything up to 2 is a bit less reliable, everything up to 4 even more unreliable, etc. Everything above 8 is maximally unreliable (and equally so: 1000 is just as unreliable as 8.01).

Any attestation done/altered by the user gets reliability 0, as the user is ultimately reliable. The reliabilities of all attestations in a quotation are added op by the tool and the background color of the text on screen is determined by the resulting figure.
A quotation with 0 (maximum) reliability will end up in green, less reliable in grayish pink, even less reliable in a pink that's slightly more red, etc. In this way the color reflects the amount of uncertainty there is about the (preloaded) attestations (where more red means more uncertain).

NOTE that texts lacking any attesation are deemed highly suspicious and hence always show as being maximally uncertain (very reddish pink).

2.5.2.5 Multiple types

This table should be filled with the different types attestations can have.
Currently the tool supports 4 types: PERSON, ORGANISATION, LOCATION and NOT KNOWN. The colors they have in the tool are in the table as well, and they can be changed if desired. The short keys ('p', 'o', 'l' and 'n' respectively) are hardwired into the code, so the names of the types should not change.

2.5.2.6 Group attestations

Multiple attestations can form a group. These groups are listed in the groupAttestations table. The group index numbers shown in the tool bear no relation with the group id's in the database. Displaying the 'real' group id's might result in very long numbers on screen where they serve no purpose but to keep the different groups apart.

Group members need not be next to each other (i.e. other words not belonging to the group might be in between). Also, group members need not be of the same type.

Groups always contain more than one member. If, by deleting members of it, a group is left with just one member, its entry in the groupAttestations table is deleted by the tool.

2.5.2.7 The way the tool gets data from the database

Freshly loaded data should have no values for the revisorId and revisionDate fields in the table lemma's (i.e. they are NULL).

When a user logs in, the tool displays the first lemma it can find that has NULL values for these columns. The user id field of this text is temporarily set to the negative user id value (-5 for user id 5). If the user logs out, or hits the previous button, this unsaved lemma is set to NULL again. It is then back in the pool again, so to say, so any other might start working on it. When a user changes something or hits the spacebar ('Save & next') the user id and current date/time is saved to the database, and as a consequence no other user will ever see the text on his/her screen.

When there are no lemma's left with NULL columns, the tool shows nothing and the job is assumed to be finished.

2.5.2.8 Attestation Tool database

Table lemma's

Field	Type	Description
Id	number	Internal identifier. Primary key.
lemma	string	This field is called lemma for historical reason. For backwards compatibility it is stil called lemma, but in the NE Attestation Tool case it actually holds the name of the document.
revisionDate	date	Date of revision.
revisorId	string	Identifier of staff member performing the revision.
externalLemmaId	string	If the text has an external identifier, it can be filled in here.

Table quotations

Field	Type	Description
Id	number	Internal identifier. Primary key.
lemmaId	number	Identifier of the lemma/text this quotation belongs to.
quotation	string	The actual text.
tokenizedQuotation	string	The text split in tokens.
quotationSectionId	string	This field is for internal use of the scripts.
dateFrom	int	Year indicating the first occurrence of names in the text.
dateTo	int	Year indicating the last occurrence of names in the text.
specialAttention	bool	Can be set when the quotation is somehow out of the ordinary.
unfortunate	bool	Can be set when the headword doesn't really occur in the quotation.

Table attestations

Field	Type	Description
Id	number	Internal identifier. Primary key.
quotationId	number	Identifier of the quotation this attesation belongs to.
onset	number	Character position of the start of the word attested.
offset	number	Character position of the end of the word attested.
reliability	float	Indicates how certain the match is (the more different the higher this number).
wordFrom	string	The word attested as occuring in the quotation.
typeId	number	Identifier of the type this attestation has (obviously, this only makes sense if there actually are types in the database).
dubious	boolean	If the attestation was marked as dubious in the tool, this field has a true value.
error	boolean	If the attestation was marked as erroneous in the tool, this field has a true value.
elliptical	boolean	If the attestation was marked as elliptical in the tool, this field has a true value.

Table groupAttestations

Field	Type	Description
Id	number	Identifier of the group.
attestationId	number	Identifier of an attestation belonging to this group.
pos	number	Position of the attestation in the quotation.

Table types

Field	Type	Description
Id	number	Internal identifier. Primary key.
name	string	Name of the type.
color	string	Hexadecimal presentation of the color attestations of this type will get in the interface.

Table revisors

Field	Type	Description
Id	number	Identifier of the revisor.
name	string	Name of the revisor.

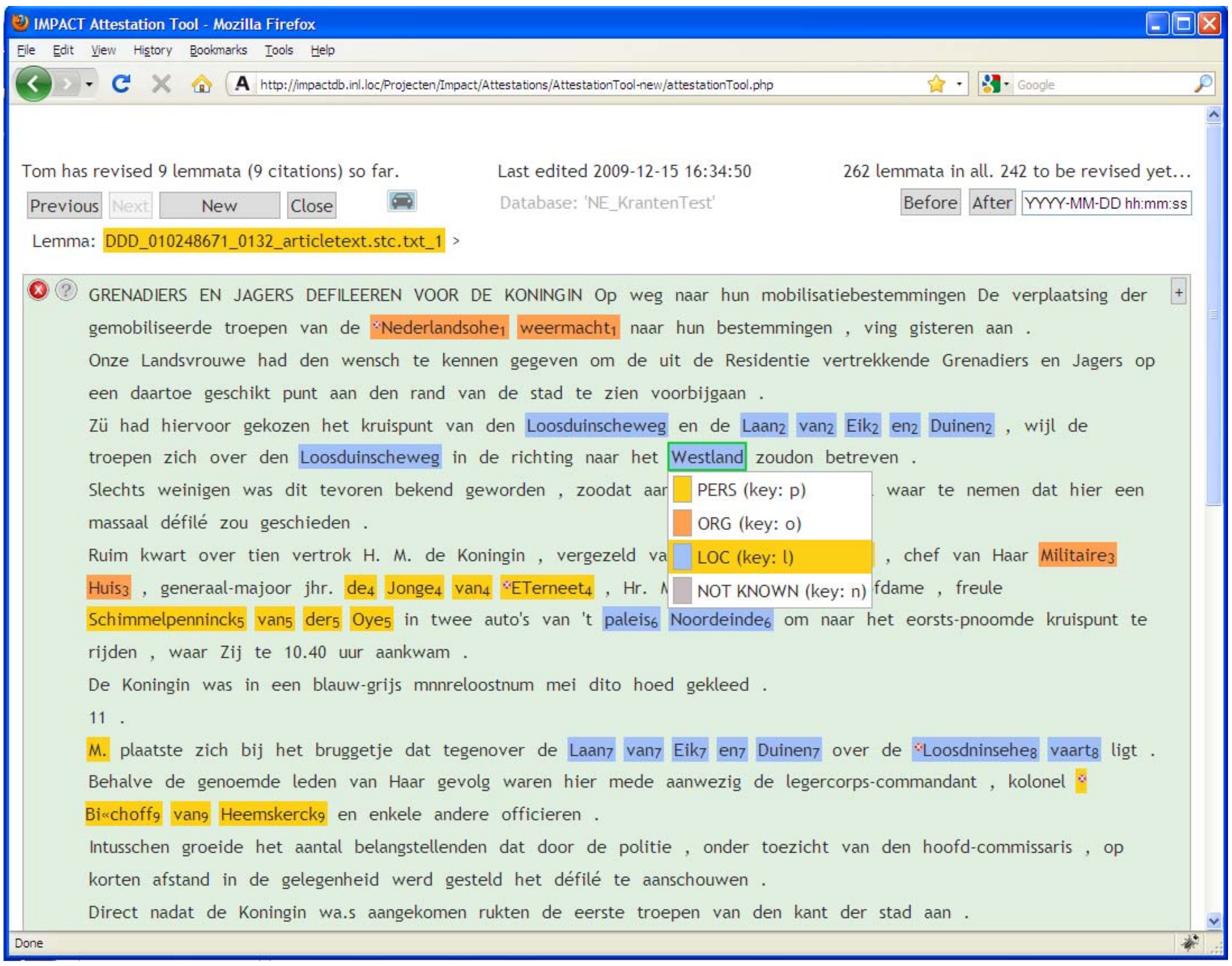
2.6 Attestation Tool GUI

The Named Entity Attestation Tool was designed to enable multiple concurrent users to view the data in the database and to make changes to it.

In the screenshot below you see the tool in action on an example page of a historical Dutch newspaper.

Names of organizations are highlighted in orange, location names are blue and person names are yellow. As is visible from the screenshot there is also a fourth category, marked grey, with which named entities of an unknown type can be marked.

By using the arrow keys or the mouse, users can select or deselect words or move a selection.


IMPACT Attestation Tool - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://impactdb.inl.loc/Projecten/Impact/Attestations/AttestationTool-new/attestationTool.php
Google
Previous
Next
New
Close
Before
After
YYYY-MM-DD hh:mm:ss
Done

Tom has revised 9 lemmata (9 citations) so far.

Last edited 2009-12-15 16:34:50

Database: 'NE_KrantenTest'

262 lemmata in all. 242 to be revised yet...

Lemma: DDD_010248671_0132_articletext.stc.txt_1 >

GRENADIERS EN JAGERS DEFILEEREN VOOR DE KONINGIN Op weg naar hun mobilisatiebestemmingen De verplaatsing der gemobiliseerde troepen van de Nederlandsohe$_1$ weermacht$_1$ naar hun bestemmingen , ving gisteren aan .

Onze Landsvrouwe had den wensch te kennen gegeven om de uit de Residentie vertrekkende Grenadiers en Jagers op een daartoe geschikt punt aan den rand van de stad te zien voorbijgaan .

Zü had hiervoor gekozen het kruispunt van den Loosduinscheweg en de Laan$_2$ van$_2$ Eik$_2$ en$_2$ Duinen$_2$, wijl de troepen zich over den Loosduinscheweg in de richting naar het Westland zoudon betreven .

Slechts weinigen was dit tevoren bekend geworden , zoodat aar waar te nemen dat hier een massaal défilé zou geschieden .

Ruim kwart over tien vertrok H. M. de Koningin , vergezeld va , chef van Haar Militaire$_3$ Huis$_3$, generaal-majoor jhr. de$_4$ Jonge$_4$ van$_4$ ETerneet$_4$, Hr. M fdame , freule Schimmelpenninck$_5$ van$_5$ der$_5$ Oye$_5$ in twee auto's van 't paleis$_6$ Noordeinde$_6$ om naar het eorsts-pnoomde kruispunt te rijden , waar Zij te 10.40 uur aankwam .

De Koningin was in een blauw-grijs mnnreloostnum mei dito hoed gekleed .

11 .

M. plaatste zich bij het bruggetje dat tegenover de Laan$_7$ van$_7$ Eik$_7$ en$_7$ Duinen$_7$ over de Loosdninseheg$_8$ vaart$_8$ ligt .

Behalve de genoemde leden van Haar gevolg waren hier mede aanwezig de legercorps-commandant , kolonel Bi«choff$_9$ van$_9$ Heemskerck$_9$ en enkele andere officieren .

Intusschen groeide het aantal belangstellenden dat door de politie , onder toezicht van den hoofd-commissaris , op korten afstand in de gelegenheid werd gesteld het défilé te aanschouwen .

Direct nadat de Koningin wa.s aangekomen rukten de eerste troepen van den kant der stad aan .

🟨	PERS (key: p)
🟧	ORG (key: o)
🟦	LOC (key: l)
⬜	NOT KNOWN (key: n)

2.6.1 Names consisting of more than one word

Very often names consist of more than one word. E.g. Laan van Eik en Duinen in the example screenshot is such a word group. In the tool this can be attested very easily just by clicking or dragging the mouse over the words. Group members don't have to be next to each other. Other words may be in between.

Words making up a group are easily recognizable by the small group index number, as is visible in the example screenshot.

2.6.2 Splitting words

Sometimes what would generally be considered as several words are spelled as one, with no spaces in between. This can be due historical spelling or scanning errors. If part of the word is part of a name the word can be split in the tool to make the right separation.

2.6.3 Scan errors

Texts might contain scan errors. If a scan error occurs in a name that has to be attested this can be indicated in the tool. An icon appears next to the word in question. E.g. Nederlandsohe in the example screenshot (which should have read Nederlandsche).

2.6.4 Dubious words

Sometimes it is not immediately evident whether a certain word (group) is a named entity or, though its type may be clear. Consider e.g. phrases like England-Wales or Johnnyboy. There might be some discussion about how to handle cases like these. In the meanwhile they might be marked as dubious, so they can be dealt with later.

2.6.5 Elliptical forms

Words may make up names, but sometimes bits are left out. Consider e.g. North and South America. North America is mentioned here, but the phrase 'North America' actually doesn't occur. The word *North* in this case can be marked as elliptical in the tool.

2.6.6 Auto attestation

When a very frequent variant has been missed in automatic matching, auto attestation can come in handy. A user can select a word and, by hitting the auto attestation button, all occurrences of this word form can be highlighted.

2.6.7 Keyboard shortcuts

To enhance the usability the interface can be used with the mouse, with the keyboard or both.

Key	Action
F2 or d	Previous revised lemma
F4 or f	Next revised lemma
F8 or Spacebar	Save current lemma, and display a new unrevised one
F9 or x	Toggle quotation as wrongly/correctly parsed
U	Toggle quotation as fortunate/unfortunate
A	Auto-attestation
E	(Un)mark attestation as elliptical (e.g. North and South America).
S	(Un)mark attestation as erroneous (scan error).
W	(Un)mark attestation as dubious.
INSERT	Insert a new attestation
DELETE	Delete currently selected attestation
< >	Walk through attestations of the selected quote
\ (backslash)	Split the token
`(back quote)	Hold this key down while clicking on an attestation to start a group attestation. `-clicking on other tokens will add them to the group.
1	Normally if an attested token is clicked on it will become unattested. If you hold the '1' key while clicking it will just be selected.

In the case of multiple typed attestations for named entities there are several more keys:

Key	Action
M	Pull down the type menu for the currently selected attestation.
L	If this key is held down while clicking on an attestation, it will be attestated as a LOCATION. Also if you press it, the currently selected attestation will become a LOCATION.
N	If this key is held down while clicking on an attestation, it will be attestated as NOT KNOWN. Also if you press it, the currently selected attestation will become NOT KNOWN.
O	If this key is held down while clicking on an attestation, it will be attestated as an ORGANISATION. Also if you press it, the currently selected attestation will become an ORGANISATION.
P	If this key is held down while clicking on an attestation, it will be attestated as a PERSON. Also if you press it, the currently selected attestation will become a PERSON.

2.6.8 Searching by date

You can look for a text that was last edited before or after a certain date.

NOTE that hitting the Previous/Next button after a 'Search by date' action will still display the previous/next lemma by id which isn't necessarily the lemma edited before/after the one resulting from the date search. This can be somewhat confusing sometimes.

The reason no better search function (e.g. search by words in the text) was implemented is because the tool is built for speed. Usually the user attests the text at hand, hits the spacebar, attests the next, spacebar, next, spacebar, and so on.

Browsing through texts previously dealt with is time consuming and should normally not be necessary. Therefore it is discouraged somewhat in this way.

2.7 Licensing

The licensing follows the consortium agreement.
The tool will be made available to the research community according to the regulations of the Dutch HLT agency (TST-Centrale, www.inl.nl), which means that it is freely available for non-commercial use.

3. Named Entity Recognition Tool

3.1 Partner

INL

3.2 Deliverable

D-EE2.3

3.3 Background

NERT is a tool that can mark and extract named entities (persons, locations and organizations) from a text file. It uses a supervised learning technique, which means it has to be trained with a manually tagged training file before it is applied to other text. In addition, version 2.0 of the tool and higher also comes with a named entity matcher module, with which it is possible to group variants or to assign modern word forms of named entities to old spelling variants.

As a basis for the tool in this package, the named entity recognizer from Stanford University is used. This tool has been extended for use in IMPACT. Among the extensions is the aforementioned matcher module, and a module that reduces spelling variation within the used data, thus leading to improved performance.

For more information on the working of the Stanford tool, see Finkel, Grenager and Manning (2005) or visit the tool's website: http://nlp.stanford.edu/software/CRF-NER.shtml. The Stanford tool is licensed under the GNU GPL v2 or later.

3.4 Differences with earlier versions

- Some bug fixes regarding error handling.
- Added setting to show the actual phonetic transcription used in the matcher
- In NERT 2.0 and up, the IMPACT extensions are separated modules from the Stanford package. That is, one can download the tool from Stanford apart from the IMPACT modules. However, the IMPACT module only works together with the Stanford package.
- The present version can handle text and (simple) xml-formats as input, as an addition to the BIO-format from version 1.0. Its spelling variation reduction module has been improved and there have been some changes on how to pass arguments and parameter settings. Finally, a matcher module has been added.

3.5 NERT requirements

NERT is a Java application and requires Java 1.6 (note that version 1.0 used Java 1.5).

3.6 The NERT package

NERT consists of a directory with the tool itself, example data and scripts:

```
NERT
data
                  matcher
models
          phontrans
          props
sample_extract
sample_train
doc
out
scripts
     tool
```

Figure 1: contents of the NERT package

The directory tool contains two jar files: *nert3-0.jar* and *stanford-ner.jar*. Both are needed to run NERT. If you don't use the NERT package but simply have the jar file *nert3-0.jar* and you get the jar file from Stanford yourself, it is necessary to rename the latter one to *stanford-ner.jar* and put it in the same directory as *nert3-0.jar* to run NERT. Another option is to unpack *nert3-0.jar* and change the classpath and main class settings in the manifest.mf file.

3.7 Extracting named entities with NERT

At the very least, three files are needed for NE-extraction with NERT. If you have those three, you are ready to go:

1) a tagged 'training file'
2) a (tagged or untagged) 'target file' from which NEs are extracted
3) a properties file

'Tagged' means that all NEs in the file have been tagged. The target file can be either tagged or untagged. If it is tagged, it is possible to calculate the tool's performance with the 'conlleval' script from the CONLL conferences (provided that the output is set to BIO-format, see below). This script can be downloaded at http://www.cnts.ua.ac.be/conll2000/chunking/output.html. However, note that for the actual extraction of NEs, tags in the target file are not necessary.

The properties file consists of a list of features, parameter settings and locations of the necessary files. This file will be discussed below. In the directory *`data/props`*, an example properties file is included.

The script `run_nert.sh` in the *`scripts`* directory can be used as an example. It trains a model with Dutch example data using the properties file from the directory *`data/props`*. It then uses its model to identify NEs in a target file.

Stanford is a statistical NE-tool. This means it needs to be trained on tagged material, which is what the training file is for. For good performance, it is key to train on material that is as close to the actual target data as possible in terms of time period and genre. More information on how to create training and target files is given below.

Training and extracting are two separate commands. After training, the tool produces a classifier ('model'), which is stored as a file. This model can then be used for extracting at any later stage.

Training the model is done by running the jar file *nert.jar* in the directory *`tool`* with the following command:

Training:

```
$ java -jar nert3-0.jar -t -props [properties file]
```

If necessary, memory can be increased as follows:

```
$ java -mx4000m -jar nert3-0.jar -t -props [properties file]
```

4000MB should be enough for the training of the model, but, if necessary and available, more memory can be used as well. When the tool does not successfully create a model during training, insufficient memory might be a reason.

The properties file gives the tool the location of the file or files it has to train with, parameter settings and the location where to write its model to (see below for more detail).

In the examples below, nert3-0.jar is called from the main directory. Note that the paths to all files in the training, extraction and matching examples are relative, so beware that the paths are correct.

Basic extraction with BIO-input and BIO-output is done as follows:

```
$ java -jar tools/nert3-0.jar -e -loadClassifier [model] -testFile [testfile]
```

We experienced cases in which the tool crashed during extraction, and this had to do with an out-of-memory error that was solved by increasing memory (similar as that for the training process).

The '-loadClassifier' and '-testFile' (or 'testDir', see below) arguments are compulsory. There are several optional extraction settings that can be added, and that will be discussed below:

```
$ java -jar tools/nert3-0.jar -e -loadClassifier [model] -testfile [testfile] -in
[txt/bio/xml] -out [txt/bio/xml] -nelist [file] -xmltags [tags] -starttag [tag] -
endtag [tag] -sv -svphontrans [file] -svlist [file]
```

NERT sends its output to STDOUT. Again, a higher amount of memory can be used as well. For extraction, a properties file is not needed. In principle, the settings from the training file will be passed on through the model. A set of relevant parameter settings can be passed to the tool via the command line. They will be discussed in the next section.

Settings

Input and output files

For training, one or more files or a reference to a directory with relevant files can be used, and the path has to be given in the properties file. There are three options:

```
trainFile=FILE
trainFiles=FILE1;FILE2;FILE3
trainDirs=DIR
```

For extraction, a single file or a reference to a directory can be used in the command line:

```
$ [ … ] -testfile [target file]
$ [ … ] -testDirs [directory]
```

Note that NERT prints the results to standard output. This means that when using a directory, all files within this directory are printed subsequently, as a whole. In order to be able to distinguish the original target files, NERT starts the output of each target file with a print of the filename when the flag '-testDira is used'.

NERT can create a file that contains a list of all found NEs with the following command:

```
        $ [ … ] –NElist FILE
```

Input and output formats

NERT 2.0 can handle three text formats: BIO, text and xml. As default, it expects BIO-format as input, and it will use this for output as well. When you are using files in text or xml format or you want a particular output in the extraction process, you need to tell NERT:

- training, in the properties file:	format=txt/xml/bio
- extracting, on the command line:	–in bio/txt/xml –out bio/txt/xml

<u>BIO-format</u>

'BIO' is an acronym and stands for the kind of tags used: B(egin), I(nside) and O(ut). Basically, each word is on a separate line, followed by the tag:

```
  Arjen POS B-PER
  Robben POS I-PER
  should POS O
  have POS O
        scored POS O
  against POS O
  Spain POS B-LOC
  . POS O
```

The middle POS tag is optional; it is not used by the tool. However, if you leave it out, it is necessary to tell the tool in the properties file the structure of your bio-input:

<u>Default:</u>

```
        format=bio
        map= word=0,tag=1,answer=2
```

<u>without the POS-tag:</u>

```
        format=bio
        map= word=0,answer=1
```

It is recommended to add a whitespace after each sentence, and tokenize your data so that periods, commas, etc. are on separate lines instead of being glued to the end of a word, since this improves performance.

If the BIO-format is needed, the script `tag2biotag.pl` in the `scripts` directory can be used. For input, it needs a text file with each word on a new line, and NEs tagged as <NE_PER|ORG|LOC>Named Entity</NE>, e.g.:

```
  <NE_PERArjen Robben</NE>
        should
```

```
have
      scored
against
<NE_LOC>Spain</NE>
```

Txt-format

NERT can also handle text format, in which the tags are wrapped around the NEs:

```
<NE_PER>Arjen Robben</NE> should have scored against <NE_LOC>Spain</NE>.
```

Again, NERT needs to know which format you are using, both in training and extraction:

```
- training, in the properties file:       format=txt
- extraction, on the command line:     -in txt
```

With text format, NERT expects the tags in the example above as default: <NE_PER>JOHN</NE>. If different tags are used, these need to be specified. In this specification, the actual tag (e.g. PER, LOC, or ORG), is represented by the word 'TAG" (in capitals):

```
- training, in the properties file:       starttag=<NE TAG>   #for <NE PER>, <NE LOC>, <NE ORG>
                                           endtag=</TAG>        #for </LOC>, </PER> etc.

                                           starttag=<NE type="TAG">       #for<NE type="PER">, possibly followed #by attributes,
                                                                          e.g. <NE type="PER" #id="2>

- extraction, on the command line:     -starttag '<NE TAG>' or -starttag <NE type="TAG">'
                                           -endtag '</TAG>'
```

If a wrong starttag and/or endtag is given, NERT will most likely crash.

In extraction, when a text file is given that has tags, NERT will use the structure of these tags for its own output, while marking the original reference tags with the tags <REF_ORG>Timbouctou</REF>. For example:

```
<PER>John</PER> and <PER>Yoko</PER>
```

with starttag '<TAG>' and endtag '</TAG>' will be outputted as:

```
<PER><REF_PER>John</REF></PER> and <PER><REF_PER>Yoko</REF></PER>
```

in which the inner tags represent the original tags and the outer tags the ones supplied by the NERT.

As a final note, although NERT is trying to preserve the original outline of a text document, there will most probably be differences in the output of whitespaces.

Xml-format

When using xml-format, the same principles apply as for txt regarding the tags. NERT deals with xml input, provided that it is told to consider only text between specific tags. Say we have the xml-file below:

```
<?xml version="1.0" encoding="UTF-8"?>
 <…>
  <…>
   <Text>
```

```
     Sally sells sea shells at the sea shore.
    <Text>
   <Text>
     Peter Piper picked a pack of pickled peppers.
    <Text>
  </…>
 </…>
</xml>
```

We have to tell NERT to only consider the text between the <Text> tags. This is done as follows:

- training, in the properties file:	xmltags=Text
or with multiple tags:	xmltags=Text;Unicode;
- extracting, on the command line:	-xmltags Text
or with multiple tags:	-xmltags 'Text;Unicode'

NERT deals with XML by simply skipping all text that is not between the specified tag(s). The relevant chunks are considered subsequently. Note that this means that in the above example, it will first train/extract the first sentence and then the following. Any NEs that would be stretched over these two chunks, would therefore be missed. Thus, the xml-format is recommended only when large chunks of text are covered by a specific tags. In other cases, it is necessary to convert the text to either text- or BIO-format.

The spelling variation reduction module

In training, NERT learns to recognize NEs by trying to identify relevant clues about both the NEs and their context. Examples of clues are use of capitals, position in the sentence or preceding or following words or groups of words (such as *in* + LOCATION).This means that the tool is sensitive to variations in the spelling of words. For example, the sentences *I come from London*, *I come fro London* and *I come frcm London* all have different words preceding the location *London* for the tool, although they are all (made up) variants of the word *from*. Thus, the tool would benefit if these variations would be diminished, and this is what the spelling variation reduction module intends to do.

The module tries to reduce spelling variation on the input data by matching potential variants, creating internal rewrite rules and by executing these rewrite rules before the tool actually uses the input. The actual output remains unchanged. In the above example, it would identify the words *from*, *fro* and *frc* as variants and create the rewrite rules *fro=>from* and *frc=>from*. These rewrite rules are applied to the input data, the tool is ran, and, in the case of extraction, the original text is used for output.

In extraction, the module looks in both the target file, the words from the original training file and, if present, gazetteer lists (which are all stored in the used model). For example, if a model has been trained with the word *fro*, it pays to create a rewrite rule in which variants of this word in the target file are rewritten to *fro*. Similarly, if the gazetteer lists contain the location *London* while the target file has the location *Londen*, a rewrite rule *Londen=>London* is created, thus enabling the tool to recognize *Londen* as a gazetteer.

The module works by transforming all words to a phonetic transcription and by comparing these versions of the words with each other. Words with the same phonetic transcription are considered variants.

This means that the rules for phonetic transcription are crucial for a proper working of this module. The module has a set of default rules, but the user can load its own set if needed:

```
- training, in the properties file:              useSpelvar=true
svPhonTrans=FILE
- extraction, on the command line:    -sv –svphontrans FILE [ … ]
```

The arguments 'useSpelvar=true' and '-sv' are the ones that initiate the spelling variation reduction module.
The rules are read by the tool and used in a simple Java *replaceAll* function. Thus, regular expressions can be used in them, but this is not necessary:

```
sz=>s
sz\b=>s
\w=>
\bcometh\b=>come
```

Before the module applies the rules, each word is put in lowercase, so only lowercase characters should be used on the left hand side of the rules. The first example rule tranforms all occurrences of 'sz' to 's'. The second uses '\b' which means it will only consider 'sz' at word boundaries. The third example rule replaces all non-word characters with nothing, thus removing them from the string. One can also use the rewrite rules to replace (or remove) complete words.

For each word, the rules are applied one by one, in the order of the file they are in. It is important to consider this order: sz=>s after the rule z=>s is useless, because all 'z' will already have been removed from the string.

Tests on Dutch historical data have shown that the module is capable of improving the scores up to a few procent. However, having the proper rewrite rules is key here. We found that more rules did not necessarily lead to better performance, due to the fact that more rules lead to more wrong variant matches. In general, the following advice can be given:

- Remove non-word characters such as dashes, whitespaces, commas and periods (\w=>)
- Check the data for commonly occurring variations. For example, Dutch 'mensch' vs. 'mens', and 'gaen' vs. 'gaan'.
- Check the effect of the rewrite rules. 'sch=>s' would work for 'mensch' but would also wrongfully change 'schip' (ship) into 'sip'. 'sch\b=>s' works better but skips the plural 'menschen'.
- Focus on words that identify named entities, such as prepositions and titles. For example, Dutch 'naer' and 'naar' (to). For those words, it pays to write a specific rule, e.g. '\bnaer\b=>naar'.

Regarding the latter remark, a script `find_NE_identifiers.sh` is added to the scripts directory, which can be used to help identifying useful words. When run on a text (in BIO-format) in which the NEs are tagged, like the training file, it lists all words preceding the NEs. These preceding words are often important predictors for NEs, and performance generally improves when reducing the amount of variation in them. The list will generally contain many prepositions and titles. The script is run as follows:

```
$ sh find_NE_identifiers.sh [file] > [outputfile]
```

NERT can print a list of created rewrite rules (variant=>word) to a file when using the following command:

- training, in the properties file:	printspelvarpairs=FILE
- extraction, on the command line	-svlist FILE

Creating training, target and properties files

Training and target files

A first step is to select and produce an appropriate training file. NERT's performance depends strongly on the similarity between the training file and the test file: when they are exactly alike, the tool can reach an f1-score of 100%. Generally speaking, the more different both files are, the lower the performance will become (although other factors also affect the tool's performance). We therefore recommend using part of a particular batch of texts for training. That is, if you have a 1 million words dataset of 19th century newspapers and 1.5 million words dataset of 18th century books, we recommend to keep them separate and to create two training files.

The size of the training file affects performance as well: the larger, the better. Below the f1-scores for a training file of ~100,000 words on different Dutch text types are shown to give an indication (table 1). The parliamentary proceedings score best, because OCR-quality is good, but mainly because it is a very homogeneous text type.

Dataset	***Time period***	***OCR - quality***	***Time period***	***f1-score***
prose, poetry, plays, non-fiction	18th c.	n/a	18th c.	70.80
	19th c.	n/a	19th c.	78.68
Parliamentary proceedings	19th c.	okay	19th c.	83.31
	20th c,	okay	20th c.	88.50
various Dutch newspapers	18th c.	poor	18th c.	73.49
	19th c.	poor	19th c.	83.92

Table 1. F1-scores of various datasets with a training file of ~100,000 words, without the use of the spelling variation module.

Another way of giving the training file a better coverage of the target file is to randomly select sentences from the data. We found that this method leads to a better performance then when, for example, the first 100,000 words from the data is used for training and the rest for testing. The script `splitFiles.pl` in the `scripts` directory can be used to create such a random set of sentences. For input it needs a text file with each sentence beginning on a new line and the desired number of words. It then creates two output files, one with the desired number of words and one with the remaining text. These files can then be used as training and target files.

```
$ perl splitFiles.pl [textfile] [number of words of output file 1] [num]
```

The third argument [num] is the total number of files that are created. Use 1 to create 1 training file and 1 target file. The script `splitFiles_BIO.pl` works the same as `splitFiles.pl`, but uses a file in BIO-format as input.

For the tagging of the training file we used the Attestation Tool from deliverable EE2.4, but other tools can of course be used as well. In the documentation of the current deliverable EE2.3, a document with NE-keying guidelines is included that can be useful. Although it is written for use with the Attestation Tool, its guidelines are generally applicable.

If the BIO-format is needed, the script `tag2biotag.pl` in the `scripts` directory can be used. For input, it needs a text file with each word on a new line, and NEs tagged as <NE_PER|ORG|LOC>Named Entity</NE>.

'Improving' data
When using OCR'd data, tool performance on person names generally increases when the training and target files are cleaned up a bit. Generally, the main things to look out for are 'errors' due to faulty OCR and tokenization as shown below.

Where	should be	*Where*
Is		*Is*
Dr		*dr.*
.		*Who*
Who		*?*
?		
O.	should be	*O.*
J		*J.*
.		*Simpson*
Simpson		
The	should be	*The*
New		*New*
TOM		*Tom*
W		*Waits*
A		*album*
I		
T		
S		
Al		
Bum		

The NER-package comes with a few Perl scripts that can fix most of the above, but it is always a good idea to double check the results. Note also that using these scripts affects your source text. The scripts work with BIO-text input and print in standard output. The scripts can be used as follows:

```
$ perl convertToLowercase.pl < [BIO-file]
```
changes all CAPITALIZED WORDS to words with Initial Capitals

```
$ perl fixInitials.pl < [BIO-file]
```
detects periods that are preceded by a single capitalized letter and a whitespace, or words listed in the script ('mr', 'Mr', 'dr', 'Dr', 'st', 'St', 'ir', 'Ir', 'jr', 'Jr', 'wed', 'Wed').

```
$ fixAbbrev.pl < [BIO-file]
```
a script specific for Dutch: changes 'v.' to 'van' and 'd.' to 'de'

Creating a properties file

A properties file consists of a list of features, parameter settings and locations of the necessary files and a link to its location should be added as an argument when training the model. An example properties file can be found at `data/props/`. Below, the contents of a properties file are shown, with a short description of the most important features:

```
trainFile=[path and name of single training file]
trainFiles=[training file1;training file2;training file3]
trainDirs=[directory with training file]
serializeTo=[path and name of the model that will be created]
map= word=0,tag=1,answer=2                        # structure of the BIO-format

useSpelVar=true                                   # use any of the spelvarmodules below
svphontrans=[path and name of file]               # file with phonetic transcription rules
printSpelVarPairs=[path and name of file]         # print all created and listed rewrite rules to file

useGazettes=true                                  # use gazetteers listed below
sloppyGazette=true
gazette=[path to list1;list2;list3; …]            # location of gazetteer lists

format=[txt/bio/xml]                              #input format. Default=bio
starttag=<NE_TAG>                                 #shape of NE-tags in txt, xml format
endtag=</NE>
xmltags=[tag1;tag2;tag3]                          #relevant xml-tags. Leave out <>

#the following features can be left like this:
noMidNGrams=false
useDistSim=false
useReverse=true
useTitle=true
useClassFeature=true
useWord=true
useNGrams=true
maxNGramLeng=6
usePrev=true
useNext=true
useSequences=true
usePrevSequences=true
maxLeft=1
useTypeSeqs=true
useTypeSeqs2=true
useTypeySequences=true
wordShape=chris2useLC
useDisjunctive=true
```

Note: in order for the spelling variation reduction module to work properly, 'useWord=true' is necessary, and if gazetteers are used, 'sloppyGazettes=true' is necessary as well.

3.8 Using the NERT named entity matcher module

The matcher module matches variants of named entities (NEs), such as *Leijden* and *Leyden*. It can also be used to match historical spelling variants of NEs to their modern form, such as *Leyden* to *Leiden*. It compares phonetic transcriptions of NEs, and calculates the distance between them by breaking them up in chunks and by calculating the number of chunks two NEs have in common. This value is then corrected for string length and normalized on a scale from 0 – 100, with 100 being a perfect match.

Phonetic transcription takes place on the basis of a set of rules, which have to be given to the matcher. Examples of phonetic transcription are /mastrigt/ for the NE *Maastricht* and /franserepublik/ for *Fransche Republiek*. NERT comes with a set of default rules that have proven to work well for Dutch. However, for other languages, (some of) the rules might have to be altered.

Using the matcher

You can tell NERT to start the matcher by using the *–m* flag as a first flag, and use the *–props* flag to tell the matcher the location of a properties file. This properties file holds the values of a set of parameters and the location of all relevant files.

```
$ java –jar tools/nert.jar –m –props propsfile.props
```

The matcher needs the following data:

- One or more files with NEs (format: one NE on each line)
- A properties file
- For lemmatizing: one or more files with NEs (format: one on each line)
- A file with phonetic transcription rules (optional)
- A file with surname identifiers for person names (optional)
- A file with transcription rules for roman numbers (optional)

The exact use of this data and all possible settings in the properties file are discussed below.

Examples

Say we have a single file with NEs and we would like the matcher to group all NEs within that file that are variants. The file is /myfiles/NE/NE-file.txt. In the properties file we then put the following:

```
file=/myfiles/NE/NE-file.txt
```

If you have your NEs in more than one file, they can be referred to by their directory:

```
dir=/myfiles/NE
```

If you want your NEs in NE-file.txt not to be matched to each other, but to NEs in a different file, e.g. NE-file2.txt, you can use the 'lemmaFile' or 'lemmaDir' option:

```
file=/myfiles/NE/NE-file1.txt
lemmaFile=/myfiles/lemmata/NE-file2.txt
```

The matcher's output will be the NEs from NE-file1.txt, with their possible variants from NE-file2.txt.
The matcher can be told in which column in the input to look for the relevant data:

```
line=type:0,ne:1
        lemmaLine=type:2,ne:3
```

The first line indicates that in the general file(s), the type of NE can be found in the first column and the actual NE in the second. The second line indicate that in the lemma file(s), the type is in the third column and the NE in the fourth. The matcher prints all output preceding the first indicated column.
The option 'ignoreWordsWithTag' can be used when you would like the matcher to ignore parts of an NE's string:

```
ignoreWordsWithTag=%
```

For example, in the NE *Jean Philippe d'Yvoy %Baron% van Ittersum tot Leersum*, the matcher will ignore the part *Baron*. It is important that both opening and closing tags are used, otherwise the ignore-option will be skipped.

Output options
The Matcher outputs only those files that are listed in the option 'onlyShowFile', and this can deviate from the actual input:

```
dir=/dir_A
onlyShowFile=/dir_A/file-A
```

This is particularly useful if we would like to have the matcher group variants from a set of lists, but we are only interested in the output of one of them. If you want the output of more than one file, use 'onlyShowFiles', with semi-colon separated filenames.

The NE matcher has different ways to print its output. The default output is as follows:

```
Amsteldam    Amsterdam
Amsteldam    Amstelredam
Amstelredam  Amsteldam
Amstelredam  Amsterdam
Amsterdam    Amsteldam
Amsterdam    Amstelredam
```

This is called the 'pairview' output, since each line shows 1 pair of NEs. If you rather want the matcher to list all variants of a single NE per line, use the groupview flag in your properties file:

```
groupview=true
```

This will print:

```
Amsteldam       Amsterdam       Amstelredam
Amstelredam     Amsteldam       Amsterdam
Amsterdam       Amsteldam       Amstelredam
```

The flag ‘showScores’ can be used to let the NE matcher also print the matching scores for each variant. ‘showScores=true’ in the properties file gives:

```
Leeuwarden      Leewarden (100)    Gemeente Leeuwarden (100)    Lieuwarden (76)
```

The flag ‘showPhoneticTranscription’ can be used to have the NE matcher print the actual phonetic transcription used in the matching process. For example:

```
        Braddock [bbrraddokk]    Bradock [bbrraddokk] Braddocke [bbrraddokk]
```

By default, the NE matcher shows all matches with a score higher than or equal to 50. Generally, scores lower than 70-75 will contain many false positives, so you can alter the minimal score by using *minScore* in the properties file:

```
minScore=75
```

Note that it might be a good idea to use a minimal score that is not *too* high, since it is harder to filter out false positives than to figure out the false negatives, that is, the matches it has overlooked. The matcher’s score can be used to quickly track the false positives.

You can also tell the matcher to only print the *N* best scores. For this, use the following flag:

```
nBest=5
```

The matcher looks at both the settings of minScore and nBest. Say we have a word with 8 matches with scores 100, 100, 80, 80, 80, 75, 75 and 50. With minScore = 50 and nBest = 2, we only see the first 2 results. With minScore = 80 and nBest = 8, we only see the first 4 results, because scores lower than 80 are not considered.

- use minScore = 0 and any nBest > 0 to always show the *N* best results, regardless of their score
- use nBest = -1 to limit the matches to any minimal score

The option ‘showDoubles=false’ can be used to have the Matcher only print out unique NE’s and their matches.

Types

The matcher can also handle NE-types (e.g. LOC, ORG, PER). For this, it needs its input in the following way,

```
LOC Amsterdam
LOC Leeuwarden
PER Piet Jansen
```

with NE-type and NE separated by a whitespace. You need to tell the matcher that you’re having types in your input file(s) by stating the following line in your properties file:

```
hasType=true
```

Note that this only tells the matcher *how* to read the input files. The matcher will still match all NEs, regardless of their type. If you want the matcher to match only PERs with PERs and LOCs with LOCs, use the following:

```
useType=true
```

By default, the types will disappear in the matcher's output, but you can tell the matcher to print them anyway by adding the following line to the properties file:

```
printTypes=true
```

This will print:

```
LOC Amsterdam	LOC Amsteldam
```

Finally, the *verbose* flag can be used for some more general output to STDERR. The flag *punishDiffInitial* is used to punish the matching scores of NEs that do not start with the same character. Its value is subtracted from the final score. The default value is 10. The flag *perFilter* (default value: *true*) sets the use of the PERfilter, which tries to handle person names more intelligently (see explanation above).

Phonetic transcription rules

As mentioned earlier, the matcher uses default rules to convert each NE to a 'phonetic transcription'. These rules can be overridden by supplying the matcher with a file with other rules, and be putting the path to this file in the properties file:

```
phonTrans=/myfiles/phonTransRules.txt
```

The rules are simple rewrite rules which Java applies to each NE one by one with a single 'replaceAll' method. For example, look at the following two rules:

```
ch=>g		# replace any /ch/ with /g/
d\b=>t		# replace any /d/ at a word boundary with /t/
```

Before the matcher applies these rules, the string is converted to lowercase. For example, if the above rules are applied, the NE *Cattenburch* becomes */cattenburg/* and *Feijenoord* becomes */feijenoort/* .

Since the matcher goes over the applied rules one by one, it is important to take the order of the rules into account. Consider for example:

```
z=>s
tz=>s
```

The latter of the two rules will never be used, since all z's are already turned into /s/ because of the first rule. The rules can also be used to simply remove characters or whole substrings from the NE, e.g.:

```
gemeente=>	#	replaces 'gemeente' with '' (void)
/W=>		#	replaces all non-word characters with '' (void)
```

NERT comes with an example file with the phonetic transcription rules for Dutch in the `matcher` directory. Note that these rules do not have to be passed to the matcher because they are the default rules.

Dealing with person names

With the exception of those strings that the matcher is told to ignore (with the phonetic transcription rules), it uses the entire NE for matching. For person names, this might easily lead to false negatives for names such as *Kurt Vonnegut*, *Vonnegut* and *Kurt Vonnegut, jr.*, because of the differences in string length.

The matcher has a built-in option to try and do a simple estimation of the structure of person names, and thus, to recognize that *P. de Vries*, *Piet de Vries* and *Pieter Cornelis Sebastianus de Vries* are (possible) variants. This option is set by the following flag:

```
perFilter=true
```

This is done by letting the matcher look for possible clues for surnames. In the given example, the word *de* is such a clue, and the matcher will consider all names preceding *de* as given names and all names following *de* as surnames. The given names are abbreviated and only the initial(s) is/are used in matching. Thus, the three examples above are reduced to *P de Vries*, *P de Vries* and *PCS de Vries*. The matcher will try to match the names by their surname first. If it finds a match, it will then look at the initials. If these match as well, it will assume that we are dealing with a variant. In this strategy, *P de Vries* and *PCJ de Vries* match, but *P de Vries* and *J de Vries* do not, while *de Vries* matches with any of the above mentioned NEs by lack of an initial.

A list of these signalling words can be added in a file and given to the matcher:

```
surnameIdentifiers=FILE
```

With the file containing a simple list of words, one on each line. An example file for Dutch in the `matcher` directory. If the matcher cannot find any clue as to which is the surname, it will only consider the last word of the NE and use this for matching. This is also the case when the perFilter is used but no file is specified (e.g. 'perFilter=true' and 'surnameIdentifiers=' or without the entire latter line).

The perFilter gets intro trouble with person names such as *Johannes X* or *Frederik de 2e*, since the matcher will only use *X* and *2e* as its matching strings because of the word *de*. For this reason, the matcher checks the NE for use of roman numbers first. If it finds any, it will consider the first name instead of the last.

Note that *Frederik de 2e* and *Frederik de Tweede* should also be considered this way. For this reason, the user can provide the matcher with a file containing rewrite rules for words and their roman counterparts, such as tweede=>II:

```
string2roman=FILE
```

As for the surname identifiers, an example file for Dutch in the `matcher` directory. If string2roman is not specified or left empty, the matcher will still find roman numbers but not the ones that are spelled out.

3.9 License and IPR protection

The Stanford tool is licenced under the GNU GPL v2 or later.

References:

Finkel, Jenny Rose, Trond Grenager, and Christopher Manning. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL 2005), pp. 363-370. http://nlp.stanford.edu/~manning/papers/gibbscrf3.pdf

APPENDIX 1: Requirement Specification for the Full-Text Transcription of historical documents from the Dutch Institute for Lexicology

General Information

A full-text transcription ("ground-truthing") and XML-tagging of various works from the KB - National Library of the Netherlands digital collection, composed of Dutch printed documents from the 18th - 20th century is needed to gather highly accurate text for the evaluation of the Dutch lexica as well as the language tools developed by INL (Dutch Institute for Lexicology) [33] in the context of the European Research Project IMPACT - Improving Access to Text.[34]

Source Material

The corpus that is to be rekeyed consists of six collections that can be treated separately or together with regard to their different characters, but the requirements that are defined herein are identically applied for all the collections.

The ground truth data set consists of two subsets:

- "Gold Standard" datasets - This is text data which INL is using to identify and mark named entities, in order to train Named Entity recognition software.
- Random datasets

See the two tables below for an overview of both subsets.

Gold Standard Data

Name	*#chars*	*# words*	*# pages*	*# articles*	*Time period*	*Size*
Staten Generaal I + II	2695554	500k	400		1815-1900; 1929-1946	1,92 GB
Newspapers I + II + III	3503320	719k	842	750	1762 - 1814, 1939, 1940	12,6 GB
Newspapers IV (ads)	175945	33278	112	210	1754 - 1806	354 MB
Books*)	286902	58100	352		1789	35 MB
Total	*6661721*		*1706*			

*) Was originally 362 pages; 10 blank pages removed

[33] http://www.inl.nl/

[34] http://www.impact-project.eu/

Random sets

Name	*#chars*	*# words*	*# pages*	*Time period*	*Size*	*Comments*
Staten Generaal *)	2985000		440	19th century, > 1815	4,01 GB	
Newspapers **)	3149000		757	300 pages <1815 ; 457 pages > 1815	1,77 GB	
Books	344129	66592	270	1795, 1784, 1786, ?, 1796	2,58 MB	dpo_63, dpo_65, dpo_93, dpo_45, dpo_113 - actually approx. 35 pages are blank - dpo_45 contains a mix of different fonts - dpo_113 contains small fragments of Latin and French
Total	*6478129*		*1467*			

**) Number of chars for Staten Generaal estimated on the GoldStandard Set: (443 / 400 pages) * 2695554 = about 2985000 chars*

***) Number of chars for Newspapers estimated on the GoldStandard Set (Newspapers I+II+III):*

*(757 / 842 pages) * 3503320 = about 3149000 chars*

Total number of pages: 1716 + 1470 = 3186 (minus approx. 35 blank pages)

Locations

Locations of the files described above on ftp://imp_tmp@dea-ftp.uibk.ac.at:

Gold Standard Staten Generaal data	INL/GS_StatenGeneraal
Gold Standard Newspaper data	INL/GS_Newspapers1 (=Newspapers I, II & III), and INL/GS_Newspapers2 (= Newspapers IV)
Gold Standard Book data	INL/GS_Book_verbessert
Random set Staten Generaal data	INL/RandomSet_StatenGeneraal
Random set Newspaper data	INL/RandomSet_Newspapers
Random set Book data	INL/RandomSet_Book

Delivery of images

The following image data will be provided via ftp-download from the homepage of the DEA, UIBK, Innsbruck, altogether upon the acceptance of the proposal:

- GS Staten Generaal: TIF, 8 bits, 300 dpi
- GS Newspapers 1: jpeg2000, 24 bits, 8,77MPixels
- GS Newspapers 2: jpeg2000, 24 bits, 3,74MPixels
- GS-Book: TIF, 400 dpi
- Randomset Staten Generaal: TIF, 8 bits, 300 dpi
- Randomset Newspapers: jpeg2000
- Randomset Books: PDF

Accuracy

The desired accuracy is 99,95% (5 falsely keyed or not keyed character amongst 10.000 characters). Please include all relevant costs (e.g. including additional setup or other) in your proposal!

Order of processing

Gold Standard Newspapers IV
Gold Standard Book
Gold Standard Newspapers I + II + III
Gold Standard Staten Generaal I + II
Random set Staten Generaal data
Random set Newspaper data
Random set Book data

Workflow

General workflow

1. open the image in the Irfanview (free program in http://www.irfanview.com/)
2. mark the block with the left mouse button, above there is infomation about the position (coordinates) of the block (for example: Selection: x,y,w,h)
3. write down in ONE xml file all the information: blocktype, coordinates of the block (if required), tags and rekeyed text

The general schema for IMPACT ground truth data must be applied:

http://www.uibk.ac.at/ulb/dea/schemas/impactgt.xsd

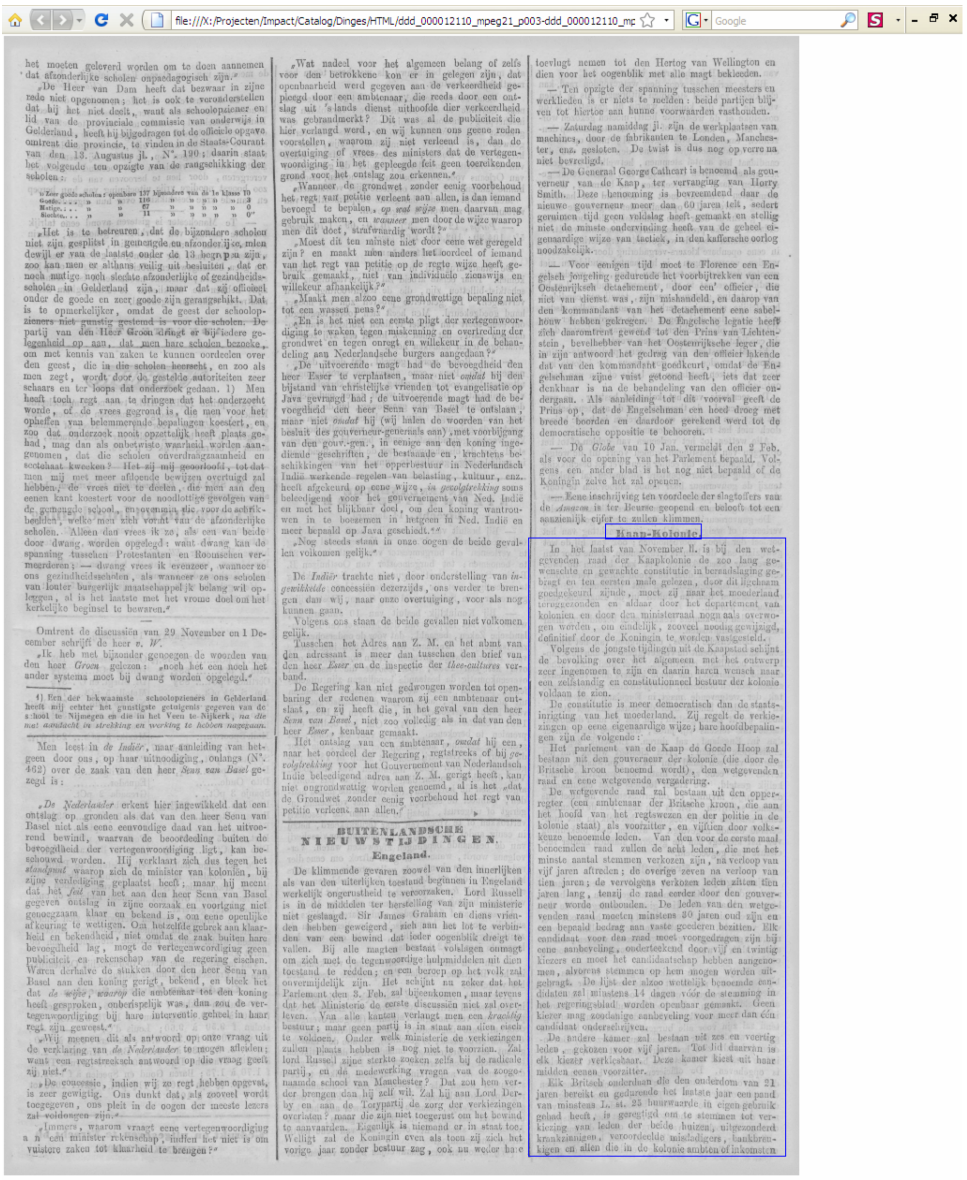
This workflow must be applied to all INL data except form the Gold Standard Newspaper data

Workflow for the Gold Standard Newspaper data

The schema 2 must be applied:

http://www.uibk.ac.at/ulb/dea/schemas/inlclipping.xsd

Here, the workflow is different from the other collections because only part of the page must be keyed. UIBK will deliver images in which the part of a newspaper page which belongs to a selected article is clipped out. These images will be sent together with the image of the whole newspaper page (for the reference purpose), but only the selected article blocks (clipped images containing parts of the page image belonging to a certain article) should be keyed, not the whole page. The name of these images contains the article id and block number which will enable INL to assemble the complete article. EXAMPLE (article id ddd:000012110:mpeg21:a0003):


file:///X:/Projecten/Impact/Catalog/Dinges/HTML/ddd_000012110_mpeg21_p003-ddd_000012110_mp

Google

het moeten geleverd worden om te doen aannemen dat afzonderlijke scholen onpaedagogisch zijn."

„De Heer van Dam heeft dat bezwaar in zijne rede niet opgenomen; het is ook te veronderstellen dat hij het niet deelt, want als schoolopziener en lid van de provinciale commissie van onderwijs in Gelderland, heeft hij bijgedragen tot de officiele opgave omtrent die provincie, te vinden in de Staats-Courant van den 13. Augustus jl., N°. 190; daarin staat het volgende ten opzigte van de rangschikking der scholen:

„Zeer goede scholen:	openbare	137	bijzondere	van	de	1e	klasse	10
Goede. . . .	„	„	116	„	„	„	„	„	3
Matige. . . .	„	„	67	„	„	„	„	„	0
Slechte. . . .	„	„	11	„	„	„	„	„	0"

„Het is te betreuren, dat de bijzondere scholen niet zijn gesplitst in gemengde en afzonderlijke, mlen dewijl er van de laatste onder de 13 begrepen zijn, zoo kan men er althans veilig uit besluiten, dat er noch matige noch slechte afzonderlijke of gezindheidsscholen in Gelderland zijn, maar dat zij officieel onder de goede en zeer goede zijn gerangschikt. Dat is te opmerkelijker, omdat de geest der schoolopzieners niet gunstig gestemd is voor die scholen. De partij van den Heer Groen dringt er bij iedere gelegenheid op aan, dat men hare scholen bezoeke, om met kennis van zaken te kunnen oordeelen over den geest, die in die scholen heerscht, en zoo als men zegt, wordt door de gestelde autoriteiten zeer schaars en ter loops dat onderzoek gedaan. 1) Men heeft toch regt aan te dringen dat het onderzocht worde, of de vrees gegrond is, die men voor het opheffen van belemmerende bepalingen koestert, en zoo dat onderzoek nooit opzettelijk heeft plaats gehad, mag dan als onbetwiste waarheid worden aangenomen, dat die scholen onverdraagzaamheid en sectehaat kweeken? Het zij mij geoorloofd, tot dat men mij met meer afdoende bewijzen overtuigd zal hebben, de vrees niet te deelen, die men aan den eenen kant koestert voor de noodlottige gevolgen van de gemengde school, en evenmin die voor de schrikbeelden, welke men zich vormt van de afzonderlijke scholen. Alleen dan vrees ik ze, als een van beide door dwang worden opgelegd: want dwang kan de spanning tusschen Protestanten en Roomschen vermeerderen; — dwang vrees ik evenzeer, wanneer ze ons gezindheidsscholen, als wanneer ze ons scholen van louter burgerlijk maatschappelijk belang wil opleggen, al is het laatste met het vrome doel om het kerkelijke beginsel te bewaren."

Omtrent de discussiën van 29 November en 1 December schrijft de heer *v. W.*

„Ik heb met bijzonder genoegen de woorden van den heer *Groen* gelezen: „noch het een noch het ander systema moet bij dwang worden opgelegd."

1) Een der bekwaamste schoolopzieners in Gelderland heeft mij echter het gunstigste getuigenis gegeven van de school te Nijmegen en die in het Veen te Nijkerk, *na die met aandacht in strekking en werking te hebben nagegaan.*

Men leest in *de Indiër*, naar aanleiding van hetgeen door ons, op haar uitnoodiging, onlangs (N°. 462) over de zaak van den heer *Senn van Basel* gezegd is:

„*De Nederlander* erkent hier ingewikkeld dat een ontslag op gronden als dat van den heer Senn van Basel niet als eene eenvoudige daad van het uitvoerend bewind, waarvan de beoordeeling buiten de bevoegdheid der vertegenwoordiging ligt, kan beschouwd worden. Hij verklaart zich dus tegen het *standpunt* waarop zich de minister van koloniën, bij zijne verdediging geplaatst heeft; maar hij meent dat het *feit* van het aan den heer Senn van Basel gegeven ontslag in zijne oorzaak en voortgang niet genoegzaam klaar en bekend is, om eene openlijke afkeuring te wettigen. Om hetzelfde gebrek aan klaarheid en bekendheid, niet omdat de zaak buiten hare bevoegdheid lag, mogt de vertegenwoordiging geen publiciteit en rekenschap van de regering eischen. Waren derhalve de stukken door den heer Senn van Basel aan den koning gerigt, bekend, en bleek het dat *de wijze, waarop* die ambtenaar tot den koning heeft gesproken, onberispelijk was, dan zou de vertegenwoordiging bij hare interventie geheel in haar regt zijn geweest."

„Wij meenen dit als antwoord op onze vraag uit de verklaring van *de Nederlander* te mogen afleiden; want een regtstreeksch antwoord op die vraag geeft zij niet."

„De concessie, indien wij ze regt hebben opgevat, is zeer gewigtig. Ons dunkt dat, als zooveel wordt toegegeven, ons pleit in de oogen der meeste lezers zal voldongen zijn."

„Immers, waarom vraagt eene vertegenwoordiging aan een minister rekenschap, indien het niet is om duistere zaken tot klaarheid te brengen?"

„Wat nadeel voor het algemeen belang of zelfs voor den betrokkene kon er in gelegen zijn, dat openbaarheid werd gegeven aan de verkeerdheid gepleegd door een ambtenaar, die reeds door een ontslag uit 's lands dienst uithoofde dier verkeerdheid was gebrandmerkt? Dit was al de publiciteit die hier verlangd werd, en wij kunnen ons geene reden voorstellen, waarom zij niet verleend is, dan de overtuiging of vrees des ministers dat de vertegenwoordiging in het gepleegde feit geen toereikenden grond voor het ontslag zou erkennen."

„Wanneer de grondwet zonder eenig voorbehoud het regt van petitie verleent aan allen, is dan iemand bevoegd te bepalen, *op wat wijze* men daarvan mag gebruik maken, en *wanneer* men door de wijze waarop men dit doet, strafwaardig wordt?"

„Moest dit ten minste niet door eene wet geregeld zijn? en maakt men anders het oordeel of iemand van het regt van petitie op de regte wijze heeft gebruik gemaakt, niet van individuële zienswijs en willekeur afhankelijk?"

„Maakt men alzoo eene grondwettige bepaling niet tot een wassen neus?"

„En is het niet een eerste pligt der vertegenwoordiging te waken tegen miskenning en overtreding der grondwet en tegen onregt en willekeur in de behandeling aan Nederlandsche burgers aangedaan?"

„De uitvoerende magt had de bevoegdheid den heer Esser te verplaatsen, maar niet *omdat* hij den bijstand van christelijke vrienden tot evangelisatie op Java gevraagd had; de uitvoerende magt had de bevoegdheid den heer Senn van Basel te ontslaan, maar niet *omdat* hij (wij halen de woorden van het besluit des gouverneur-generaals aan) „met voorbijgang van den gouv.-gen., in eenige aan den koning ingediende geschriften, de bestaande en, krachtens beschikkingen van het opperbestuur in Nederlandsch Indië werkende regelen van belasting, kultuur, enz. heeft afgekeurd op eene wijze, *in gevolgtrekking* soms beleedigend voor het gouvernement van Ned. Indië en met het blijkbaar doel, om den koning wantrouwen in te boezemen in hetgeen in Ned. Indië en meer bepaald op Java geschiedt.""

„Nog steeds staan in onze oogen de beide gevallen volkomen gelijk."

De *Indiër* trachte niet, door onderstelling van *ingewikkelde* concessiën dezerzijds, ons verder te brengen dan wij, naar onze overtuiging, voor als nog kunnen gaan.

Volgens ons staan de beide gevallen niet volkomen gelijk.

Tusschen het Adres aan Z. M. en het ambt van den adressant is meer dan tusschen den brief van den heer *Esser* en de inspectie der *thee-cultures* verband.

De Regering kan niet gedwongen worden tot openbaring der redenen waarom zij een ambtenaar ontslaat, en zij heeft die, in het geval van den heer *Senn van Basel*, niet zoo volledig als in dat van den heer *Esser*, kenbaar gemaakt.

Het ontslag van een ambtenaar, *omdat* hij een, naar het oordeel der Regering, regtstreeks of bij *gevolgtrekking* voor het Gouvernement van Nederlandsch Indie beleedigend adres aan Z. M. gerigt heeft, kan niet ongrondwettig worden genoemd, al is het „dat de Grondwet zonder eenig voorbehoud het regt van petitie verleent aan allen."

BUITENLANDSCHE NIEUWSTIJDINGEN.

Engeland.

De klimmende gevaren zoowel van den innerlijken als van den uiterlijken toestand beginnen in Engeland werkelijk ongerustheid te veroorzaken. Lord Russell is in de middelen ter herstelling van zijn ministerie niet geslaagd. Sir James Graham en diens vrienden hebben geweigerd, zich aan het lot te verbinden van een bewind dat ieder oogenblik dreigt te vallen. Bij alle magten bestaat volslagen onmagt om zich met de tegenwoordige hulpmiddelen uit dien toestand te redden; en een beroep op het volk zal onvermijdelijk zijn. Het schijnt nu zeker dat het Parlement den 3. Feb. zal bijeenkomen, maar tevens dat het Ministerie de eerste discussiën niet zal overleven. Van alle kanten verlangt men een *krachtig* bestuur; maar geen partij is in staat aan dien eisch te voldoen. Onder welk ministerie de verkiezingen zullen plaats hebben is nog niet te voorzien. Zal lord Russell zijne sterkte zoeken zelfs bij de radicale partij, en de medewerking vragen van de zoogenaamde school van Manchester? Dat zou hem verder brengen dan hij zelf wil. Zal hij aan Lord Derby en aan de Torypartij de zorg der verkiezingen overlaten? maar die zijn niet toegerust om het bewind te aanvaarden. Eigenlijk is niemand er in staat toe. Welligt zal de Koningin even als toen zij zich het vorige jaar zonder bestuur zag, ook nu weder hare toevlugt nemen tot den Hertog van Wellington en dien voor het oogenblik met alle magt bekleeden.

— Ten opzigte der spanning tusschen meesters en werklieden is er niets te melden: beide partijen blijven tot hiertoe aan hunne voorwaarden vasthouden.

— Zaturdag namiddag jl. zijn de werkplaatsen van machines, door de fabrikanten te Londen, Manchester, enz. gesloten. De twist is dus nog op verre na niet bevredigd.

— De Generaal George Cathcart is benoemd als gouverneur van de Kaap, ter vervanging van Horry Smith. Deze benoeming is bevreemdend daar de nieuwe gouverneur meer dan 60 jaren telt, sedert geruimen tijd geen veldslag heeft gemaakt en stellig niet de minste ondervinding heeft van de geheel eigenaardige wijze van tactiek, in den kaffersche oorlog noodzakelijk.

— Voor eenigen tijd moet te Florence een Engelsch jongeling gedurende het voorbijtrekken van een Oostenrijksch detachement, door een' officier, die niet van dienst was, zijn mishandeld, en daarop van den kommandant van het detachement eene sabelhouw hebben gekregen. De Engelsche legatie heeft zich daaromtrent gewend tot den Prins van Lichtenstein, bevelhebber van het Oostenrijksche leger, die in zijn antwoord het gedrag van den officier lakende dat van den kommandant goedkeurt, omdat de Engelschman zijne vuist getoond heeft, iets dat zeer denkbaar is na de behandeling van den officier ondergaan. Als aanleiding tot dit voorval geeft de Prins op, dat de Engelschman een hoed droeg met breede boorden en daardoor gerekend werd tot de democratische oppositie te behooren.

— De *Globe* van 10 Jan. vermeldt den 2 Feb. als voor de opening van het Parlement bepaald. Volgens een ander blad is het nog niet bepaald of de Koningin zelve het zal openen.

— Eene inschrijving ten voordeele der slagtoffers van de *Amazon* is ter Beurse geopend en belooft tot een aanzienlijk cijfer te zullen klimmen.

Kaap-Kolonie.

In het laatst van November ll. is bij den wetgevenden raad der Kaapkolonie de zoo lang gewenschte en gewachte constitutie in beraadslaging gebragt en ten eersten male gelezen, door dit ligchaam goedgekeurd zijnde, moet zij naar het moederland teruggezonden en aldaar door het departement van kolonien en door den ministerraad nogmaals overwogen worden, om eindelijk, zooveel noodig gewijzigd, definitief door de Koningin te worden vastgesteld.

Volgens de jongste tijdingen uit de Kaapstad schijnt de bevolking over het algemeen met het ontwerp zeer ingenomen te zijn en daarin haren wensch naar een zelfstandig en constitutionneel bestuur der kolonie voldaan te zien.

De constitutie is meer democratisch dan de staatsinrigting van het moederland. Zij regelt de verkiezingen op eene eigenaardige wijze; hare hoofdbepalingen zijn de volgende:

Het parlement van de Kaap de Goede Hoop zal bestaan uit den gouverneur der kolonie (die door de Britsche kroon benoemd wordt), den wetgevenden raad en eene wetgevende vergadering.

De wetgevende raad zal bestaan uit den opperregter (een ambtenaar der Britsche kroon, die aan het hoofd van het regtswezen en der politie in de kolonie staat) als voorzitter, en vijftien door volkskeuze benoemde leden. Van den voor de eerste maal benoemden raad zullen de acht leden, die met het minste aantal stemmen verkozen zijn, na verloop van vijf jaren aftreden; de overige zeven na verloop van tien jaren; de vervolgens verkozen leden zitten tien jaren lang, tenzij de raad eerder door den gouverneur worde ontbonden. De leden van den wetgevenden raad moeten minstens 30 jaren oud zijn en een bepaald bedrag aan vaste goederen bezitten. Elk candidaat voor den raad moet voorgedragen zijn bij eene aanbeveling, onderteekend door vijf en twintig kiezers en moet het candidaatschap hebben aangenomen, alvorens stemmen op hem mogen worden uitgebragt. De lijst der alzoo wettelijk benoemde candidaten zal minstens 14 dagen vóór de stemming in het regeringsblad worden openbaar gemaakt. Geen kiezer mag zoodanige aanbeveling voor meer dan één candidaat onderschrijven.

De andere kamer zal bestaan uit zes en veertig leden, gekozen voor vijf jaren. Tot lid daarvan is elk kiezer verkiesbaar. Deze kamer kiest uit haar midden eenen voorzitter.

Elk Britsch onderdaan die den ouderdom van 21 jaren bereikt en gedurende het laatste jaar een pand van minstens L. st. 25 huurwaarde in eigen gebruik gehad heeft, is geregtigd om te stemmen tot verkiezing van leden der beide huizen, uitgezonderd krankzinnigen, veroordeelde misdadigers, bankbreukigen en allen die in de kolonie ambten of inkomsten

```
<IMPACT_INL_GS_Clipping>
      <File filename="DDD_000012110_004_access_a0003_01.jp2">
      <Article id="ddd:000012110:mpeg21:a0003">
            <TextBlock number=”1”>
                  <Headline>Kaap-Kolonie.</Headline>
            </TextBlock>
      </Article>
      </Filename>
</IMPACT_INL_GS_CLIPPING>
```

The XML-filename for the file above would be: DDD_000012110_004_access_a0003_01.xml

```
<IMPACT_INL_GS_Clipping>
      <File filename="DDD_000012110_004_access_a0003_02.jp2">
      <Article id="ddd:000012110:mpeg21:a0003">
            <TextBlock  number=”1”><Paragraph>In  het  laatst  van  November
      l&apos;l.  is  bij  den  wetnderf  raad  der  Kaapkolonie  de  zoo  lang
      geilde  en  gewachte  constitutie  in  beraadslaging  gebragt  en  ten
      eersten male gelezen, door dit lig ….. (etc)
      </Paragraph>
      <Paragraph>.........</Paragraph>
            </TextBlock>
      </Article>
      </Filename>
</IMPACT_INL_GS_CLIPPING>
```

The XML-filename for the file above would be: DDD_000012110_004_access_a0003_02.xml

Delivery

Output

For each image/file one xml file with information written in according to the corresponding schema.

There are two way to name the XML file:

1. If one file contains only one page or a part of the page than one XML file per image shall be created with the file name reflecting the Image-ID, for example:

Image file name:	*160014-0017R.tif*
xml file name:	*160014-0017R.xml*

2. If one file contains more than one page, than one XML file per page, having the book filename and page number in the XML filename, for example:
dpo_035_0123.xml

Delivery

Delivery of the fully transcribed and tagged XML-files is expected to happen in chunks of approx. 500.000 characters per day. A timeframe of max. 2,5 months after the signing of the contract for the complete delivery of all six collections and possible corrections must not be exceeded!

Two weeks after the signing of contract is setup time. In this time the following works should be done by the service provider:

1. Prepare for the process: recruit data processing personal, training, write formula….
2. In the first week: process 5% of the first 3 collections, send the output to the INL.
3. In the second week: process 5% of the rest 3 collections, send the output to the INL; correct the output of the first 3 collections and send it back to the INL

So that after these two setup weeks the process can be done quickly.

The six collections will be processed accordingly, one collection after another. Moreover, the result should be sent every week on Monday, at 9a.m. (European time) so that a checking process can be carried out in parallel. The output files will be checked by the INL staff and within one week time they will be either accepted or the need for corrections and the according issue will be expressed. The contractor should then provide the corrected version within the next week.

Legal Obligations

The contractor is hereby obliged to save and use the attached sample images exclusively as a means to determine the proposal. Further on, the local storage of the files is only granted for backup reasons. This backup, upon completion of the work, is to be deleted immediately. Additional backups and use, in particular for purposes of the contractor or other parties, are not permitted, as is the transfer of files to other parties.

Any violation of this agreement will lead to an immediate cancellation of the contract and to a claim for compensation. This agreement shall be governed by German law; the court of Munich is the exclusive place of jurisdiction.

Structuring

The two schema shall be used. The following tags are hereby defined to be used in the full-text files:

<page>

<page_number>

If a page number is printed on the page, it shall be encoded via this tag. All character inside the page number block must be keyed.

<block>

- Where the text comprises of two or more blocks, the text in each of them shall be enclosed by a <block>-tag. The order of the blocks is their order in the xml file.
- Block types which should be marked:
  - textblock
  - illustrationblock
  - tableblock
  - unknownblock

- The main criteria to trigger a block are „change of layout“. So: If the layout (e.g. column number, font size, font type, type face, distance between lines, etc.) changes and the change expands to the whole line, than a new block will start.
- Blocks are also triggered if:
  - Change in text direction (vertical/horizontal text)
  - Graphical separators
  - the layout of lines changes: centred, left-, right-aligned; justified;
- Blocks are NOT triggered if:
  - bold, italic, larger words, etc. appear WITHIN a line and not expand to the whole line
  - if there is a capital (Versalie) on the beginning of a paragraph which spans two or more lines
  - If there is no change in line distances, font type, font size and the following lines are only bold or italic
- Rules to define the order of blocks
  - Always from top to bottom, horizontal text than vertical text.
  - If more than one block at the same height: from left to right.
  - Always finish one column before going to the next. The column finishes at the end of the page or when a graphical separator/frame runs through the whole page width.
  - Different blocks belong to one column only when a straight line can be drawn in both sides (right and left) of the blocks so that they contain only these blocks (in whole) and not a part from another/other blocks.
  - If a perfect order of blocks cannot be created (happens mainly with newspapers) a lower accuracy rate should be accepted.

<paragraph>

- They appear only in running text. Whenever a new paragraph starts, the <paragraph>-tag must be used.
- One block may contain several paragraphs but no paragraph spans a block.
- They are triggered if:
  - there is an indent of the first line (positive or negative); or
  - there is a clear distance to the following lines

<cell>

- They appear only in tableblock.
- The order of cell in one tableblock is defined according to the rules applied to the order of blocks

<running_title>

If applicable, the running title (repeated title shown on the top of the page) shall be tagged.

<headline>

Where headlines occur, they shall be encoded via the <headline>-tag.

<footnote>

Footnotes shall be rekeyed and enclosed in a <footnote>-tag at the bottom of the running text.

<marginalia>

If marginalia exist on the page, special conditions for the keying of these shall be applied: the complete text for each marginalia shall be keyed and enclosed by the <marginalia>-tag.

<unknownElement>

If some characters or parts of the text can by no means be safely rekeyed, then the <unknownElement>-tag shall be used. It can also contain a number of question marks ("?") equal to the number of characters that are illegible.

<textstype>

Cursive, bold, and superscript must be tagged. So we'll have: italic, bold, superscripts, unkown.

General

The encoding for all full-text files must be UTF-8.

Special Characters

Initials:

Initials are large capital letters, mostly upon start of a new paragraph or chapter that are very illustratively shaped, sometimes even hand-drawn. Often they take the space of several lines. In some cases the next character of the word is also set as a capital letter. Initials shall be treated like regular characters and no special tagging needs to be applied.

Coloured text:

Coloured text shall be treated like regular text and no special tagging needs to be applied.

Line breaks in the full- text file must correspond to the line breaks in the source material. For hyphenations at the end of line, the "-"-symbol (negation symbol) must be used.

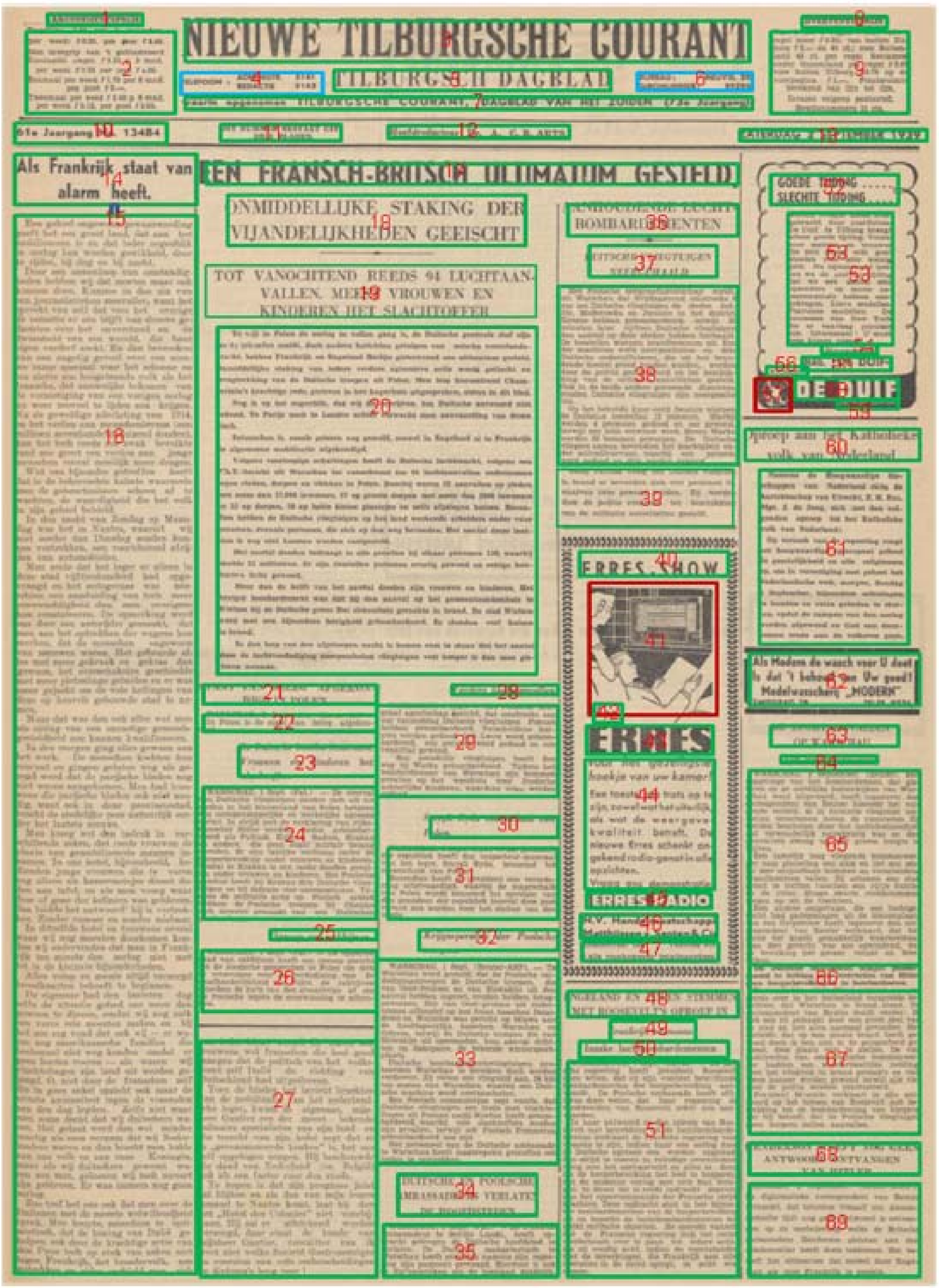
Example

On the following pages, several graphical examples are given to illustrate the intended use case for each of the tags defined above.

Note: the following images show ONLY the basic block segmentation, according to the criteria from above:

- textblock – green
- illustrationblock - red
- tableblock – blue

Otherwise the frames are drawn not very exactly; the purpose is only to show how the blocks and blocks order should be detected. Only in the first pages all the blocks are marked. In the following pages only the important blocks are marked.


NIEUWE TILBURGSCHE COURANT

TILBURGSCH DAGBLAD

61e Jaargang No. 13484

Als Frankrijk staat van alarm heeft.

EEN FRANSCH-BRITSCH ULTIMATUM GESTELD

ONMIDDELLIJKE STAKING DER VIJANDELIJKHEDEN GEEISCHT

TOT VANOCHTEND REEDS 94 LUCHTAANVALLEN. MEER VROUWEN EN KINDEREN HET SLACHTOFFER

BOMBARDEMENTEN

ERRES SHOW

ERRES

ERRES RADIO

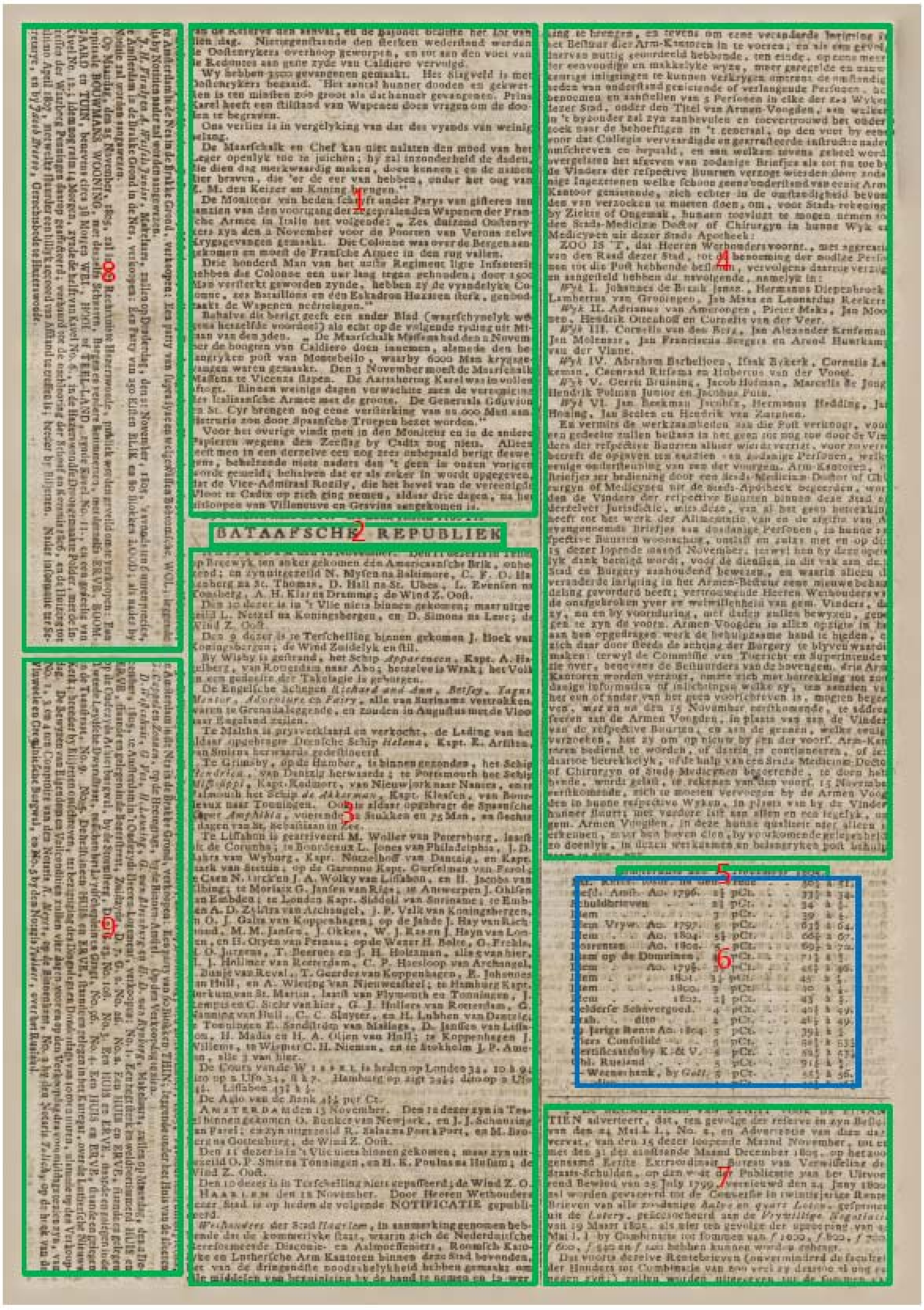
GOEDE TIDING ... SLECHTE TIDING

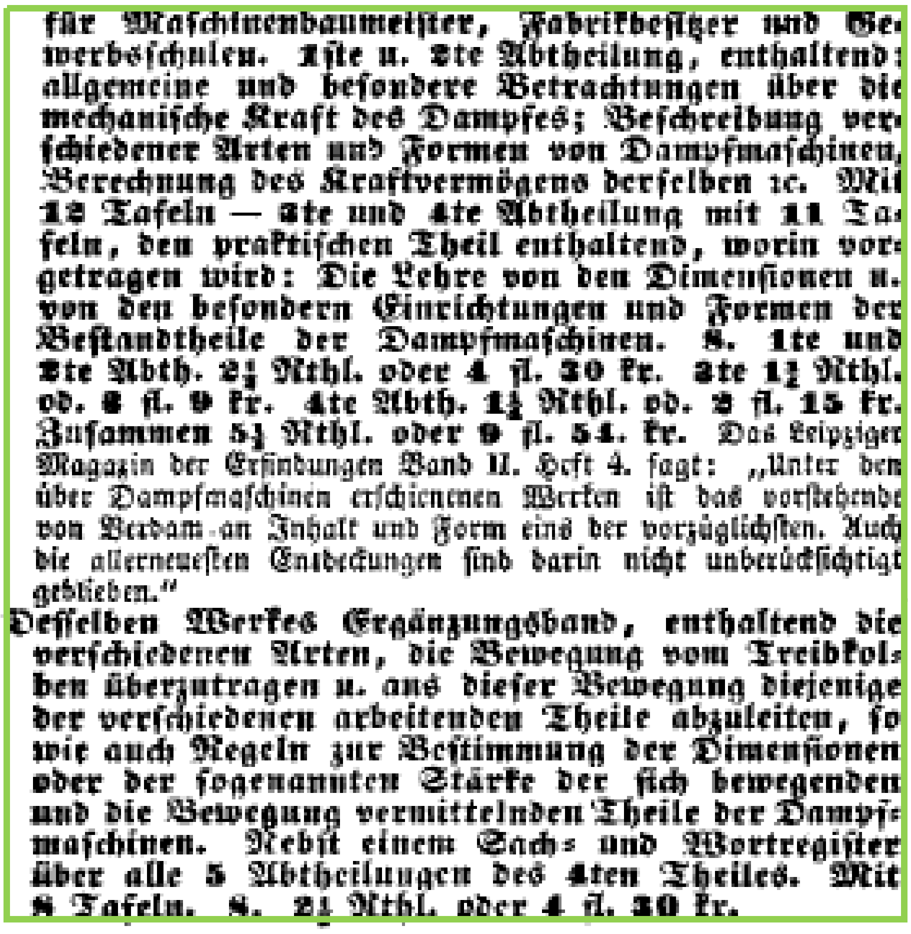
DE DUIF

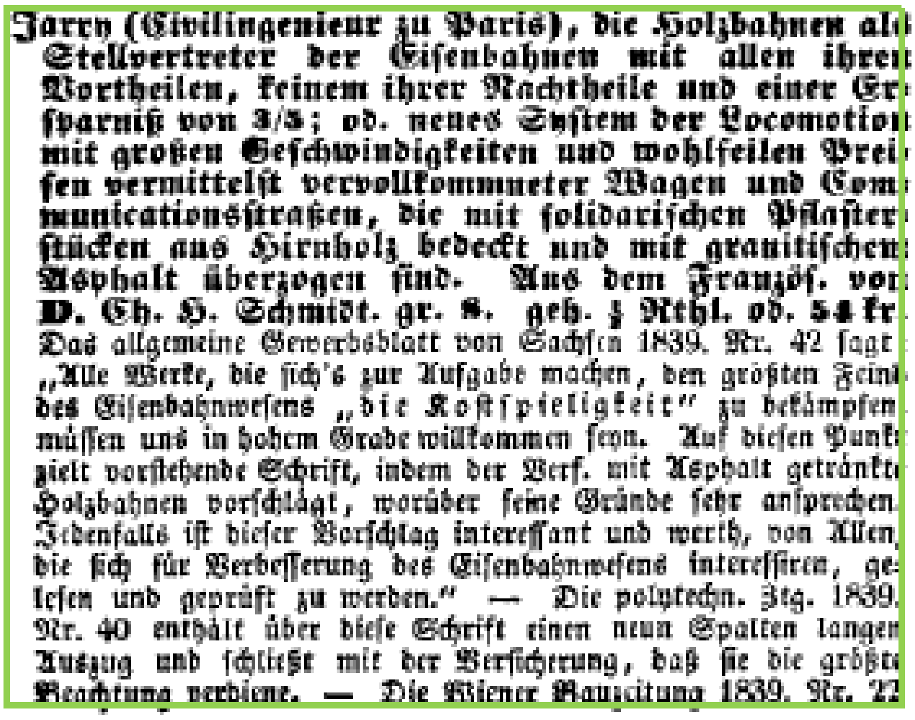
Als Modern de wasch voor U doet is dat 't beloop van Uw goed! Modelwasscherij „MODERN"

DUITSCH EN POOLSCHE AMBASSADEURS VERLATEN

[illegible]


BATAAFSCHE REPUBLIEK
1
2
3
4
5
6
7
8
9


für Maſchinenbaumeiſter, Fabrikbeſitzer und Ge-
werbsſchulen. 1ſte u. 2te Abtheilung, enthaltend:
allgemeine und beſondere Betrachtungen über die
mechaniſche Kraft des Dampfes; Beſchreibung ver-
ſchiedener Arten und Formen von Dampfmaſchinen,
Berechnung des Kraftvermögens derſelben ꝛc. Mit
12 Tafeln — 3te und 4te Abtheilung mit 11 Ta-
feln, den praktiſchen Theil enthaltend, worin vor-
getragen wird: Die Lehre von den Dimenſionen u.
von den beſondern Einrichtungen und Formen der
Beſtandtheile der Dampfmaſchinen. 8. 1te und
2te Abth. 2½ Rthl. oder 4 fl. 30 kr. 3te 1¾ Rthl.
od. 3 fl. 9 kr. 4te Abth. 1¼ Rthl. od. 2 fl. 15 kr.
Zuſammen 5½ Rthl. oder 9 fl. 54. kr. Das Leipziger
Magazin der Erfindungen Band II. Heft 4. ſagt: „Unter den
über Dampfmaſchinen erſchienenen Werken iſt das vorſtehende
von Verdam an Inhalt und Form eins der vorzüglichſten. Auch
die allerneueſten Entdeckungen ſind darin nicht unberückſichtigt
geblieben.“
Deſſelben Werkes Ergänzungsband, enthaltend die
verſchiedenen Arten, die Bewegung vom Treibkol-
ben überzutragen u. aus dieſer Bewegung diejenige
der verſchiedenen arbeitenden Theile abzuleiten, ſo
wie auch Regeln zur Beſtimmung der Dimenſionen
oder der ſogenannten Stärke der ſich bewegenden
und die Bewegung vermittelnden Theile der Dampf-
maſchinen. Nebſt einem Sach- und Wortregiſter
über alle 5 Abtheilungen des 4ten Theiles. Mit
8 Tafeln. 8. 2½ Rthl. oder 4 fl. 30 kr.


Jarry (Civilingenieur zu Paris), die Holzbahnen als
Stellvertreter der Eiſenbahnen mit allen ihren
Vortheilen, keinem ihrer Nachtheile und einer Er-
ſparniß von 3/5; od. neues Syſtem der Locomotion
mit großen Geſchwindigkeiten und wohlfeilen Prei-
ſen vermittelſt vervollkommneter Wagen und Com-
municationsſtraßen, die mit ſolidariſchen Pflaſter-
ſtücken aus Hirnholz bedeckt und mit granitiſchem
Asphalt überzogen ſind. Aus dem Franzöſ. von
D. Ch. H. Schmidt. gr. 8. geh. ½ Rthl. od. 54 kr.
Das allgemeine Gewerbsblatt von Sachſen 1839. Nr. 42 ſagt:
„Alle Werke, die ſich's zur Aufgabe machen, den größten Feind
des Eiſenbahnweſens „die Koſtſpieligkeit“ zu bekämpfen,
müſſen uns in hohem Grade willkommen ſeyn. Auf dieſen Punkt
zielt vorſtehende Schrift, indem der Verf. mit Asphalt getränkte
Holzbahnen vorſchlägt, worüber ſeine Gründe ſehr anſprechen.
Jedenfalls iſt dieſer Vorſchlag intereſſant und werth, von Allen,
die ſich für Verbeſſerung des Eiſenbahnweſens intereſſiren, ge-
leſen und geprüft zu werden.“ — Die polytechn. Ztg. 1839.
Nr. 40 enthält über dieſe Schrift einen neun Spalten langen
Auszug und ſchließt mit der Verſicherung, daß ſie die größte
Beachtung verdiene. — Die Wiener Bauzeitung 1839. Nr. 22

Appendix II: TAGGING NAMED ENTITIES: applying the 1999 NER Task Definition (NERTD)[35] to Dutch historical newspapers, parliamentary proceedings, 18/19th-c. fiction

Background

This document is an addition to the NER Task Definition (NERTD), which provides extensive rules for the tagging of named entities. It focuses on specific examples and issues found in the Dutch historical data shown below. Most examples are therefore in Dutch.

- newspapers (18th and 19th century)
- parliamentary proceedings (19th and 20th century)
- a collection of prose, poetry, fiction and non-fiction (18th and 19th century)

General

The tagging should not anticipate automatized procedures; one and the same form may be a name in one context but not in another.

categories:

- PERS
- LOC
- ORG
- NOT KNOWN: is NAME but undecided which type (e.g. PERS or LOC in Greek mythology). Not to be used as 'miscellaneous')

Direction for buttons when using the IMPACT NE Attestation Tool (DD EE2):

w-button: dubious: not sure if NAME, but if so, it is clear which type to be used sparsely / temporarily, to be discussed; avoid "OPTIONAL" Names& forms containing something extra

`Jantje-lief`
`Rijnarm`
`Reschid-Pacha`
`Nederland-België`

s-button: scanning & printing errors
(also if uncertain; better used too often than that erroneous variants are included without special marking)
(we also use it with words that were hyphen-ated at end of line, or where hyphen was wrongly removed in OCR)

e-button: elliptical part of multi-name expression: e`Noord`$_1$ `en` `Zuid`$_1$ `Amerika`$_1$

Ignore any PERS/LOC/ORG

- within a title (of a book, journal, opera, etc.): *NERTD § 5.1.7*& see below for delimitation and interaction with ORGS. (we don't consider headings etc. to be titles)

[35] Nancy Chinchor, Erica Brown, Lisa Ferro and Patty Robinson, *1999 Named Entity Recognition Task Definition*, MITRE, 1999. http://www.nist.gov/speech/tests/ie-er/er_99/doc/ne99_taskdef_v1_4.pdf

- within names of treaties, laws, meetings, etc.
- referring to ships, statues, etc.
- in text that is in a language other than Dutch (cq German).

Tag NAMES in other languages (e.g. `Belgium foederatum`) if in Dutch (German) context.

The use of capitals is not decisive; historical sources are similar to speech transcripts in this respect (see below for ORGs).

Genitive/possessive **-s & ’s** is included; the *NERTD*-rule that possessive **’s** should be excluded is ignored in the cases that an apostrophe occurs – relatively rare in Dutch and German.

Do not include article

`De <ORG>Gebroeders`$_1$ `Murray`$_1$`</ORG>`

only if commonly associated with an entity name

`The Hague` *(NERTD: p. 9)*

`de Vries`

`The big Apple` for “New York”. *(NERTD: p. 8)*

Nicknames/aliasses should be tagged if “established” *(NERTD: p. 12)*

Isolated part of multi-name expression:

- if not a name by itself: tag, group, & mark with **e** button (‘elliptical part’):

  `<LOC>`^{e}`Noord`$_1$`</LOC> en <LOC>Zuid`$_1$ `Amerika`$_1$`</LOC>`

  Note that this deviates from *NERTD § 4.1.2*, according to which the part-without-head should be tagged just like that.

- if name by itself: just tag, without **e**, and not grouped; only the full expression forms group

  `<PERS>Jan</PERS> en <PERS>Marie`$_1$ `de`$_1$ `Groot`$_1$`</PERS>`

- complication (rare): plural head with ORGs: way out is to exclude the headword from tag

  `Departementen van Verkeer`$_1$ `en`$_1$ `Waterstaat`$_1$ `en van Economische`$_2$ `Zaken`$_2$

  with LOCs – no proper solution, too bad, is very rare anyway:

  ^{e}`Oost-`$_3$ `en Zwarte`$_3$ `Zeeën`$_3$

Persons (PERS)

PERSON: named person, family, or certain designated non-human individuals

no titles/functions etc

NERTD p.11ff

Titles such as “Mr.” and role names such as “President” are not considered part of a person name. However, appositives such as “Jr.,” “Sr.” and “III” are considered part of a person name also: `Zn`

not: `Hertog, oom/tante, sint/St., H.` etc.

exception: if fixed part of 'name', e.g. **`meester`** `Prikkebeen,` **`Uncle`** `Sam`

tag <u>in entirety, group:</u>

Jacob$_1$ Eduard$_1$ de$_1$ Witte$_1$, junior$_1$ Jacob Eduard de Witte Jun

Jacob Henriques de Barrios Jr.

including initials, prefixes, maiden name:

k.f. de groot-de vries

if maiden name occurs separately: discont. group:

S. Verlegen$_1$ Geboren Virago$_1$

tag <u>separately</u>

<PERS>Jacob$_1$ den$_1$ Tweeden$_1$</PERS> van <**LOC**>Engeland</LOC>

Locations vs Persons

LOC vs. PERS remains arbitrary to a certain extent, there will be inconsistencies in this area In cases like the following, assume that `Leeuwaerden` belongs to the name (rather than LOC)

Justus$_1$ van$_1$ Leeuwaerden$_1$ (for Dutch, cf. DBNL)

Hermingard van de Eikenterpen

vs. Hertog van <LOC>Brabant</LOC>

(Hertog van) Alva: PERS ('exception' if..)

Jan$_{\mathbf{PERS}}$ Graaf van **Holland**$_{\mathbf{LOC}}$

if PERS than including article:

G.W.W.C. Baron **van Hoëvell**	PERS
De heer **de Voltaire**	PERS

Dutch article with French person may suggests location rather than LOC:

Prinses **de Lamballe**	PERS
Prins van **Joinville**	LOC

`God`, **`Jezus`**, **`Christus`**, **`Satan`**: <u>tag</u> unless occurring as curse/exclamation/interjection.

not: `messias`, `devil`, etc, as these aren't proper names

Organizations (ORG)

ORGANIZATION: named corporate, governmental, or other organizational entity

Proper names that are to be tagged as ORGANIZATION include stock exchanges, multinational organizations, businesses, TV or radio stations, political parties, religions or religious groups, orchestras, bands, or musical groups, unions, non-generic governmental entity names such as "Congress" or "Chamber of Deputies," sports teams and armies (unless designated only by country names, which are tagged as LOCATION), as well as fictional organizations (to ensure consistency with marking other fictional entities).

(NERTD p.18)

NB also churches, museums, etc.!

Proper names referring to meeting places or places where organizational activities occur (e.g., churches, embassies, factories, hospitals, hotels, museums, universities) will be tagged as ORGANIZATION. *(NERTD p.20)*

Tag news sources (newspapers, radio and TV stations, and news journals) as ORGANIZATION even when they function as artifacts. *(NERTD p.21)*

Whenever capitalization information is not available or is unreliable, as in the case of speech transcripts (see Appendix A), then organization-designating common or proper nouns are considered part of the name.

"chrysler division" *(NERTD p.6)*

`Het` **`Leesgezelschap , Het Lezen strekt tot nut Vermaak , bevordert en verfijnt den Smaak:`** incl. "Leesgezelschap"

We tend to have a rather broad interpretation of what may count as ORG. Not only official ORGs, and not only official name variants:

A human annotator can sometimes determine the meaning by mentally inserting the appropriate head noun to see if the meaning is changed in the given context (e.g., "Republicans" vs. "Republican Party")

(NERTD p.24)

Some terms are sometimes NE, sometimes not – depending on context (incl. implicit context)

e.g. `vergadering` normally not tagged in Dutch when meaning "meeting",

but tagged if referring to French Assemblée nationale

not if used in generic sense

To be tagged if it concerns a specific ORG that is acting as an ORG: '... decided...', 'members of ...' etc. (whether or not it is written with a capital is irrelevant)

not, because too vague/general:

```
bestuur
Oranjehuis
```

not, ambiguous, would be LOC if specified

```
provincie
land
gemeente
```

`gemeente` **`Den Haag`**: only **`Den Haag`** = LOC

occasionally:

`Rijk, Staat`

exceptionally:

`Gemeenebest, Republiek` (both are normally LOCs)

to be tagged (unless used in generic sense):

```
Staten-Generaal
regering
gouvernement
Rekenkamer
Eerste/Tweede Kamer
Ministerie van Oorlog etc.
Department of Something
Hooggerechtshof
criminele rechtbank
Openbaar Ministerie      so also  Publiek ministerie
Staatscommissie van redactie
Overijselse Staten
Surinaamsche Geregtshof
Nieuwe Kerk
Schutters[]doelen
```

Vader van 't **`Kathuizers Klooster`**
een hs. uit het klooster **`Rodendael`** of **`Rode Cluse`**
in het **`Guillemiten-klooster`**
De Abdis van het klooster **`Santa Maria del Nova`**

For Dutch, see also:

- http://www.inghist.nl/Onderzoek/Projecten/Socialezekerheid/instellingen_en_personen/list_instellingen
- http://www.inghist.nl/Onderzoek/Projecten/Repertorium/app/instellingen/index.html?instelling.naam=&instelling.apply.used=&order_by=naam&start=0&instelling.actions.zoeken=zoeken
- http://www.inghist.nl/Onderzoek/Projecten/Socialezekerheid/lokale_instellingen

"truncated" – cf `((Boston) (Red)` **`Soxs`** *NERTD: p.6*

Problematic: where to draw the line......

we decided to tag truncated forms rather generously - as long as it's clear that they have a specific referent, e.g.:

she works at the UNIVERSITY in this town
the BLABLA SOCIETY ..... this SOCIETY...
the REPUBLIC ....[if referring to e.g. Dutch Republic]

tagged also if not explicitly specified but evident that a particular ORG is referred to, e.g.

`Staten`	if it stands for `Staten-Generaal / Staten van Overijssel...`
`parket`	if it stands for `OM`
`Justitie`	if it stands for `Ministerie van Justitie`
`Ministerie`	if it stands for `Ministerie van Iets`, or 'cabinet'
`volksvertegenwoordiging`	if it stands for `Staten-Generaal / 1ste & 2de Kamer`
`(Staats) Bewind (1802)`	
`Universiteits-Bibliotheek / UB`	
`Provinciale Staten`	(e.g. van Zuid Holland)
`Gedeputeerde Staten`	
`Staten`	`i`f clearly – Generaal, of Provincale –, or… is meant. (& if not referring to member(s)) e.g. Hoogmogende Heeren `Staaten Generaal Der Vereenigde Nederlanden`: Here the 'Staaten' are persons, so `Staaten Generaal Der V. N.` is not to be tagged; just `Vereenigde Nederlanden = LOC`
`Raad`	if clearly Hooge (or...) Raad is meant (& if not referring to person/member(s))
`Kamer`	if clearly the 1ste of 2de Kamer is meant
`Ministerie`	if clearly non-generic, e.g. "Min. van Justitie", based on (implicit) context (verg. `Justitie` if clearly "Ministerie van –" is meant)
`Staatscommissie`	(idem...)

(interesting to know what lemmata from the normal lexicon can occur as NE)

`Stadschouwburg`	
`Schouwburg, Stedelijk`	if known under that name
`Maatschappij` ('Gesellschaft') (van...)	
`Commissie`	

Beyound a certain point, not truncated because form is too general

`printer` even if referring to ""Printer Jansen"

fuzzy...

not:	`minister van` **`Marine`**
tagged:	`ambtgenoot van` Marine
	`Atjehsche kust van de Indische` marine

Problematic: when to include the adjective because it can be considered part of the name?

cf *NERTD: p.17* "russian air force"

`russian <B_ENAMEX TYPE="ORGANIZATION">`**`air force`**`<E_ENAMEX>`

p. 21 `general of the <ORG>`**`army`**`<>`

vs. official name in entirety:

`<B_ENAMEX TYPE="ORGANIZATION">`**`louisiana state police`**`<E_ENAMEX> (p.22)`

no Adj:

het Indische **leger**
de Amsterdamse **politie**
de Engelse regering
Nijmeegse **Stadsarchief**

Adj included ?

Overijselse Staten

Adj included:

Fransche Akademie = Académie française

We allow for alternative names including translations

Commercial institutions/companies: tag as ORG

Hotel X
Restaurant Y
(café) het Zwaantje

'Gedrukt bij', "printed at/by"

Jan Döll	= ORG
P. van der Eyck en D. Vijgh	= group ORG

ORGs within a 'title' (*cf. NERTD § 5.1.7*):

We do tag ORGs in cases of Proceedings, Papers, Series..., where it can be difficult to draw the line between a genuine Title and a description, and where the ORG seems to be what matters anyway:

Handelingen der **Staten-Generaal**
Handelingen der **Maetschappye** (= Maatschappy der Nederlandsche Letterkunde)
Werken der **Maatschappy**
Verhandelingen van het **Oudheidkundig Genootschap**
Baltimore defeated the **Yankees**: Baltimore=LOC en Yankees ORG!

Locations (LOC)

LOCATION: name of politically or geographically defined location (cities, provinces, countries, international regions, bodies of water, mountains, etc.) and astronomical locations.

Germany invaded Poland: both LOC and not ORG

Baltimore defeated the Yankees: Baltimore=LOC en Yankees ORG! *(NERTD: p.8)*

Buildings, gates not LOC unless

- "primarily built as monument"
- functioning as address

can be ORG though: churches .... commercial...

The following combinations are to be tagged as LOC:

```
Russische Rijk
Romeinsche Rijk
Oostersche Roomsche Rijk
Heilige Roomsche Rijk
Alpische gebergte
Zeeuwsche Eilanden
Waddeneilanden
Brabantsche Hertogdom:
Frans- Vlaanderen
Europisch Turkeyen
holy Land
```

also: `Koninkrijk der Nederlanden` also `Koninkrijk` if clearly...
`Republiek der Verenigde Nederlanden` also `Republiek` if clearly...
`Oostenrykscheʼ Nederlanden` (political entity)
idem `Zuidelijke Nederlanden` in certain context

?? `Zuidelijke/Noordelijke Provinciën`

> There is a gray area here, it seems to depend what is known (also by us) as a political/gegraphical entity

tag PARTLY as LOC:

Oostelijk **Saxen**: just Saxen loc
Haven van **Alexandryen**: just Alexandrien
Zeeland bewesten **Schelde**: tag seperately without bewesten
Hertogdom **Nassau**: just Nassau
Koningryk **Polen** : just Polen

NOT to be tagged as LOC: (because not "built primarily as monument") (though possibly to be tagged as ORG if functioning as one, or as LOC if it concerns an address)

Burcht-zaal
Aalmoesseniersweeshuis ORG
't Klooster van de Regulieren ORG
Conversationshaus
St. Jacobs parochiekerk ORG
Stadhuis ORG
Villa Borghese ORG
Anthoniespoort
Gemeenelandshuis Etc etc

NOR:
Hel / paradijs / hemel
Overzeesche bezittingen
Provins
Distrikt
Westlijken Rijkszetel